Contents

[01-01 Introduction.txt 2](#_Toc499300569)

[01-02 What is Computational Photography.txt 5](#_Toc499300570)

[01-03 Dual Photography.txt 8](#_Toc499300571)

[01-04 Panorama.txt 10](#_Toc499300572)

[02-01 Digital image.txt 13](#_Toc499300573)

[02-02 Point processes.txt 19](#_Toc499300574)

[02-03 Blending modes.txt 21](#_Toc499300575)

[02-04 Smoothing.txt 24](#_Toc499300576)

[02-05 Convolution and cross-correlation.txt 28](#_Toc499300577)

[02-06 Gradients.txt 32](#_Toc499300578)

[02-07 Edges.txt 37](#_Toc499300579)

[03-01 Cameras.txt 42](#_Toc499300580)

[03-02 Lenses.txt 47](#_Toc499300581)

[03-03 Exposure.txt 51](#_Toc499300582)

[03-04 Sensor.txt 55](#_Toc499300583)

[04-01 Fourier Transform.txt 59](#_Toc499300584)

[04-02 Blending.txt 63](#_Toc499300585)

[04-03 Pyramids.txt 67](#_Toc499300586)

[04-04 Cuts.txt 70](#_Toc499300587)

[04-05 Features.txt 73](#_Toc499300588)

[04-06 Feature Detection and Matching.txt 78](#_Toc499300589)

[05-01 Image Transformation.txt 84](#_Toc499300590)

[05-02 Image Morphing.txt 92](#_Toc499300591)

[05-03 Panorama.txt 96](#_Toc499300592)

[05-04 High Dynamic Range.txt 100](#_Toc499300593)

[05-05 Stereo.txt 106](#_Toc499300594)

[05-06 Photosynth.txt 113](#_Toc499300595)

[05-07 Extrinsic Camera Parameters (opt).txt 119](#_Toc499300596)

[05-08 Intrinsic Camera Parameters (opt).txt 123](#_Toc499300597)

[05-09 Calibrating Cameras (opt).txt 126](#_Toc499300598)

[06-01 Video Processing.txt 131](#_Toc499300599)

[06-02 Video Textures.txt 133](#_Toc499300600)

[06-03 Video Stabilization.txt 138](#_Toc499300601)

[06-04 Panoramic Video Textures.txt 144](#_Toc499300602)

[07-01 Light Field.txt 147](#_Toc499300603)

[07-02 Projector-Camera Systems.txt 153](#_Toc499300604)

[07-03 Coded Photography.txt 158](#_Toc499300605)

# 01-01 Introduction.txt

01 - Intro.srt

Hello. Welcome to the class on computational photography. This is an exciting class and I'm actually very pleased to be bringing it to you in this format. And what I'd like to do throughout this whole semester is introduce to you this amazing concept of computation and photography, which is about the computational pipeline. With the relationship to the photography discipline which has a lot of art and aesthetics associated with it. Again I want to remind you, this is a class on the computational and the technical pipeline of photography. And throughout the semester we're going to look at a variety of things associated with how competition has changed the whole aspect of how light is captured to generate images and photographs.

02 - About Me.srt

So let me take this opportunity to first introduce myself. My name is Irfan Essa. One of the first things anybody who has ever taken a class with me gets to know is how to pronounce my name. Again, it's Irfan Essa. I've been a professor here in the School of Interactive Computing at Georgia Tech for about 18 years now. Prior to that I actually did a master's and PhD at the MIT in the media lab there, and actually stayed on there for a few years as a research scientist. And my whole 20 odd or more than years of work has been in the areas of images and videos, where I've actually been doing work on analysis and enhancement of images and videos. And including disciplines like computer vision, computer graphics, machine learning, and robotics. And , now computational photography is also a discipline of its own, and I've been teaching courses on it for many years.

03 - Overview.srt

So now let me take a moment to tell you a little bit about the class. This class is about computational photography, which is actually nowadays also considered a discipline. This discipline is about building the computational technology, and the tools to figure out how light from the environment is used to generate images and photographs. We will be concentrating on the computational aspect of this pipeline, how computation impacts photography. It's an interesting discipline because if you really hard think about it, the computation is impacting an artistic and aesthetic aspect of what photography is. But again, I emphasize, we'll be spending time on trying to understand the technology that impacts this whole pipeline of how light is used to generate images and photographs.

04 - Course Structure.srt

So now let me tell you a little bit about the overview of this class and how we have structured it for you. We're going to have these video lectures where I'm going to introduce to you variety of technical concepts associated with the field of computational photography. Within these lectures there'll be once in a while a few quizzes, and each one of them is aimed to get you more engaged with the material, and learn more about different aspects of this specific discipline we're talking about. In addition to the lectures, we've also created a variety of assignments. There will be one assignment due almost every week throughout the whole term, and in these assignments we're going to play, have you interact with material in different ways. Sometimes we'll actually have you do coding to be able to generate computational photography artifacts. But also how we produce artifacts that can engage with each other, in a peer feedback, peer review manner. There will be an exam, somewhat near the end of the term, which will cover and review the concepts throughout the whole term of what we've looked in in computational photography. Then, we will actually allow you to do a final project in an area of your own choosing. I will provide you with variety of different ideas on what you can do, but again this is something in my many years of experience in computational photography and teaching these concepts is the students enjoy finding something free form that they can do on their own. We have a great team of people working with me on supporting this class. There will be head TA who will be interacting with you, as will I, on the different types of fora made available to this class. And we look forward to interacting with you throughout the whole term, engaging with you in different variety of ways. I'm excited about this class, I'm excited bringing this whole concept to you. I hope you're excited in learning about it, and hopefully you'll engage with me throughout the term. The tagline I always tell everybody about computational photography, have fun computing with photographs.

05 - Requirements.srt

So one question I usually get is, what are the requirements for this class, and what is expected, and specifically, what is expected of students who are enrolling in this class. Let me try to outline some of those things now. So just to re-emphasize, this is a computational photography class, and we're interested in trying to understand the computational pipeline of how an image is formed, how a camera captures an image, the rays of light. And also, how do we actually process this information? To do this, you will need to know things like linear algebra, calculus, and probability. Linear algebra, because as you will learn, we will represent a whole lot of things as matrix, matrices, and vectors. Calculus because we'll be looking at differential and integration types of processes to understand the math. Some of the concepts of what happens within an image, and probability because at the end of the day there's a lot of sampling associated with it. Look at these images and we need to be able to look at from the statistics of this and probability distribution function of these. Another thing that would be important is the comp, the computational release associated and what happens when an image is formed. To support this we're going to provide you with tools one an experimental one within browser for you to track images, but then to really do this kind of stuff in real, you'll have to play around with a toolkit called or, or an API called openCV. CV stands for computer vision. Python and C++ are the languages associated with that. We'll give you examples on how to get started on these types of things. In fact, some of the same things will be used within a browser. The good thing with this is you will be able to put this on your own workstations and actually do the assignments. And more importantly play around with the concepts we will be discussing in class. Another thing that is available to you would be things like Matlab and Octave, where you can actually also do these types of things. That are, actually we going to be covering in class. One question I always get is do I need a high-end camera? Well, having any kind of camera would be useful because you'll be taking a few pictures and playing around with this on your own. But really there's no need for high-end SLR camera or something like that. , if you feel like buying one. Use this class as an excuse. Feel free to. We will also provide you with images of various types of you know, for example, different exposure levels to do HDR and stuff like that, that'll help you do your assignments and all the work that is needed. But remember, the biggest goal of this class is start opening black boxes on the kinds of things that you've been using with images and photography all the time. How does a camera work, what are the processes that go within a camera? Your favorite software that you use is Photoshop. In this class, we learn how to open the black boxes and how some of the tools that you've been using regularly. What do they really do to images? So that's really more about the technical aspects of photography that we're going to be looking at.

06 - Module 1.srt

Now, I'm going to just try to outline the different things we will cover in this class. First module of this class is introduction. We will talk about, what is computational photography? I'll just give you an example of a kind of a, a newer type of a camera that has been an artifact that's been produced, as we have started diving deeper into the whole concept of computational photography. We will learn more about what are different artifacts of computational photography, and what I'll do in this module is introduce you to two simple concepts. Well, actually, they are not that simple, but some concepts that actually would be valuable for you to kind of remember throughout the course and these will be the things that I'll kind of use as examples throughout, will help us understand how we are going to do computational photography. I'm going to provide some sort of a context on the scope of what is computation and how does it really relate to all the other disciplines. As I've stated, computational photography is a discipline of it's own now, has its own conference and publications and stuff like that in the academic manner. But what we're interested in is learning what other disciplines impacted. In this one, in the first, when we get started, initial weeks, we'll have a couple of assignments for you. One to get started doing simple processing with images and stuff like that.

07 - Module 2.srt

The next module is when we start getting into more analysis of the image itself. Here we'll actually learn more about the digital image representations, how you represent images to make them computable objects. We will learn about pixel and point processes of images. Again, doing some mathematics on these images or computation on them. We will learn things like, how do you smooth and filter images? learn the basic principles, the intuition behind these. We will learn more about how we can extract different types of information from images for analysis. And in this timeframe, we'll actually do experiments where you'll actually learn more about image filtering and also feature detection.

08 - Module 3.srt

Third module, we're actually going to look inside the camera. We will look at the basic foundations of a camera using a pinhole camera concept. We will learn more about the importance of optics, how does and you know, various types of lenses, what kind of a role they play in photography. And what computation has to do with them. And also the basics of how does a camera work. We learn more about sensors within a camera, again, these are the represent, these are the systems that will take the information that a camera has and converts them to digital information that we want to do some things with. And in this one, we learn about concepts of epsilon photography and also a fun experiment that you will do is try to build your own pin hole or a camera obscura.

09 - Module 4.srt

And the fourth module will get deeper into the images again, where we learn more about how we take multiple images and blend them together, to generate weird images like this, or also kind of find information within images that allow us to do analysis. So this will lead us to thinking about images and the concept of what are the sampling rates of images, what are the frequencies, and how do you look at images on the frequency spectra. We will learn about things like blending images together to generate novel, perhaps crazy images. We will learn about features, again these are points here that actually would be used to combine these two images together if that's the goal. And you'll learn more about how to do image blending, again with an exercise.

10 - Module 5.srt

Once we get into that kind of concept, we will now move to more advanced ideas of how actually we do computational photography. So for example, learning how to build panoramas. This is a panorama I took of the Georgia Tech basketball team, the Yellow Jackets at their arena a few years ago. Or also how do you play around with high dynamic range imagery like this one ,where you can actually see the snow, and also the insides quite clearly. We'll learn more about how to do image editing, but more importantly, you'll get to play around with all of this, by doing some simple coding and programming will lead you to things like HDRs and panoramas.

11 - Module 6.srt

We will also learn how to extend a bunch of these concepts that we're learning about in this class to video. Now, to be very honest, video is the real area that I work on as a researcher and it's an academic discipline that I've been working on for oh, two decades now. In this class, I will introduce some of the basic ideas, some of them actually related to the work, we ourselves have done, of capturing video, video as you know is becoming even a more popular medium now, ,. Sports video analysis becomes very common, and, as we start getting into fancier, newer types of cameras like the quadcopter cameras that are capturing video. In this context we will learn about video textures and also learn about video stabilization, two of the techniques that my group has also worked on, and actually more importantly, we'll introduce you to the whole area of video with a simple and fun experiment and assignment to let you do video textures.

12 - Module 7.srt

Then, we will step back further diving deeper into the technology and learn about computational cameras. These are newer forms of cameras that are showing up, which have additional features and additional abilities, again, you beyond the norm of what traditional photography is, these are commercially available cameras from Lytro, which are light field cameras, more and more standard cell phone cameras eventually will have an array of cameras attached to it, which is shown here, and also newer research efforts like this one, where a camera and a projector system can be merged together, to be able to then show, you know, a different way of looking at information in a scene, so again, we will look at light filled cameras, we will look at multi-view cameras, projector camera systems, and, you know, this will be another fun thing we will do.

13 - Module 8.srt

In module Eight I'm going to introduce to even special and really kind of from the, recent research efforts some of the exciting new topics in computational photography. For example, we'll learn about newer camera technologies. How we can actually even control lighting in cameras, to be able to generate newer forms of images. How does concept of blur and deblur in images, and how do new forms of computational cameras can be used to deal with those, types of artifacts? More importantly, we'll also deal with the fact that now, cameras are everywhere, and this has led to a whole new form of social and crowd photography. Again we'll discuss those types of examples, more importantly I do want everybody to start thinking we will do a final project. Or more appropriately you will do a final project of your own choosing, on a topic that we covered in class, or something that you want to do on your own, and we'd like to provide you with various types of supports to get you there. But more importantly, we want you to play around with newer concept on your own. So that's an important part I want everybody to think about as you will select topic, I will provide some examples of the kinds of things you could do, and you will make it work for real in the last few weeks of class.

14 - What to Expect.srt

So let me reiterate here again what to expect from this class, why you should be taking it, and what I am expecting out of this class. And well actually, more importantly, the topics I'm covering and the goals of this class. So one thing of the offset, I want to make sure it's clear to everyone. This is not a photography class. What that means is I'm not going to be teaching you the basics of how to take pictures or also to take good pictures. I, personally, am a photographer. Hopefully, I'll share with you all of the kinds of different things I have done in the area of photography. But, I have an amazing job where I can actually also work in the field that I actually am also a hobbyist in. I take photography for, as a hobby, but I really want to kind of spend time also on the technology related questions associated with photography. So this class is really about the technology related content. You will learn about the back end technology. We open the black boxes of cameras and other types of image analysis software. Again take a very computation driven effort through it. Now, I do argue that learning about what's behind the tools that you use may actually help you become a better photographer. But, that's a, not a direct, you know, prerequisite of this class. If you are taking pictures already you may enjoy this class a lot more because you'll learn more about the technology behind it. But, more importantly if you don't know anything about photography, this won't teach you much about photography. But, it'll teach you about the cameras that you use, and the tools you use, and you know, some may argue that might make you a better photographer. This is also a hands-on class. So you will learn by doing things yourselves. So, that's important too. Many people have asked questions like, oh, I'm not really comfortable with a lot of programming. Well, this class will get into a little bit of programming. Or quite a bit. And you will need to know some of the tools that require you to actually manipulate images using simple programs. So I do want everybody to be aware of that. At one point, you will learn by doing by yourself, and with the class you will share things. As I said, we have homework assignments, and peer feedback assignments, to just kind of keep things interesting, and also learning from each other, while learning from the content and the material shared with you. So, to summarize this class is going to be an interesting class where we're going to learn about the technology aspects of photography. Computational photography's are a discipline of its own. We're going to cover our ideal ways to introduce to you the concepts of computational photography. You will learn by doing things and you will hopefully have fun and the tag line that I love to share with this class all the time is you will learn to have fun computing with photographs. So that's the goal and hopefully you'll have fun doing it. Thank you.

# 01-02 What is Computational Photography.txt

01 - Intro.srt

Welcome back. In the previous lecture, I gave you an idea of what this class is about in terms of the discipline of computational photography. In this lecture, I want to tell you a little about, more about what computational photography is. We will actually now, tease apart the whole process of how light is used to generate pictures. And we will look at each and every aspect of how computation can impact it. This will kind of create a framework, a foundation of what we now will leverage throughout the entire term. To understand computational photography at it's core. So we will actually look at different aspects and frameworks associated with what makes computational photography a discipline. And we will actually keep on looking at it throughout the whole term. I look forward to working with you on it.

02 - Lesson Objectives.srt

The objectives of this specific lesson are for you to learn the basic concept of, what is computation photography? And I will introduce you to the basic fundamental elements of what makes computational photography. Let's get started.

03 - What is Photography.srt

So what is competition photography? Well, before we get that let's ask the question, what is photography? And the best way to do this these days, if you believe that, is to go search for it on the internet, and I will. Just take a chance and see what the Wikipedia page on photography says about it. Couple of things too we should look at here. One, photography is a science and art. It's a practice of creating images by recording light. It could even be some other forms of things like electromatic, electromagnetic radiation, but let's just stick to light. Either electronically or by means of an image sensor. Or chemically. So that's film. Electronic sensor is digital. And to generate an artifact, an image. So in essence, if you really think hard about it is, photography is a process of taking light from a natural environment and storing it in a form of an image. The literal definition of the term photography is drawing with light, all right? I mean, that's the exact literal definition of the word photography. In essence, what we are interested in is taking the light from a natural environment most of the time and creating an image to store that information. And of course, what we I do this it depending on the application, could be scientific, could be art, aesthetics, depends on the user at the end. to do this, we have a variety of things that we can use, there is a lens that we can play around with which then has variety different types of things. A camera has a lens and we look at all those types of things again. And again this part gets into the whole process of how an image would be created from that kind of stuff and then what we do with it at the end, right. So we can share it, we can print it. So, basically the whole process of drawing with light and using that to share the image, for a variety of purposes, either art, science, is an important part of what we do with photography. it's related to many disciplines in science and manufacturing and all that kinds of stuff and many different applications come in. That's what makes photography very interesting and exciting.

04 - What is Computational Photography.srt

So now in relation to this, what is computational photography? Well, one thing you may actually want to start thinking about now is, that these days cameras are pretty complex devices. So if you buy any kind of camera, doesn't have to be a high-end based DSLR, that is the, you know, digital version of the older SLR cameras, single lens reflex. But there is a lot of computation involved with it. But we want to actually start thinking it, a little bit more. Imagine that there is a relationship now, much more symbiotic one, between a camera and a computer. What can happen when I really bring a much more powerful computer and attach it to a camera? Well that's the kind of things computational photography's attempted to do. And a example of this is a camera 2.0. Sometimes also referred to as a Frankencamera which is a research attempt done a while ago. Building a camera like this with a lot of computation so you can actually change things like put multiple flash, that's multiple light sources. Also control various aspects of how the light is captured. That's one example of what computational photography would be. But more and more, if you look in your pockets you have smartphone cameras. These phone cameras have computation built in where the camera is, so computational photography can impact even devices like the ones you have in your hands, or even the computers these days all come with cameras attached to them, perhaps for things like you know, teleconferencing applications. But if you noticed, camera and, and computers are now becoming integral parts of each other, which basically means that now, we can actually take computation and impact the entire workflow of how light is captured, to generate an image or a video. And that's what computational photography really is about. The whole entire workflow of how light is captured to generate an image, photograph or video. How can computation actually impact that whole workflow all the way to sharing the images or the videos that you have at the end. Well that's what computational photography in the simplistic matter is.

05 - Computational Photography Combines.srt

So building on that concept let's, define, what computational photography is, and what it combines, basically it combines computing, the ability to convert information into a digital representation, also using digital sensors, which they basically provide us with digital information, this makes it all computable, that is processing, and all that kind of stuff can be done with it. We can actually use, a more advanced optics with this cameras do require optics, we'll cover this in detail also. Actuators, because now I can actually put an actuators to actuate, for example, each and every optic, how the sensor does a variety of things, then smart lights, I can actually also put in,lights that have some sort of computation associated with them. So this computation is involved with the sen, the sensor, the optics, the actuators, and also the smart lights, so in essence, what that basically means is, it combines all of them to generate newer forms of cameras, sensors, newer forms of artifacts that actually let you do a lot more with photography. So in essence, computational photography aims to do a lot more than traditional film or even traditional digital cameras do. So it basically now gives you a newer paintbrush to draw with light, all right, not just the traditional one where the passive process of exposing a film, or even a sensor to the light in an environment capturing and preserving it or sharing it afterwards, while that is actually a great pipeline, we all love it, but now, using computational photography, perhaps we can do more, and perhaps extend beyond the limitations, assuming we consider those to be limitations, I personally think there's nothing wrong with traditional photography, I love to do it myself, but this now gives us a newer medium.

06 - Limitations of Traditional Film Cameras.srt

So just to kind of think about it. Lets look deeper into what film camera's all about. And again I want to emphasis that I don't think anything's wrong with them and I'm happy to use them all the time too. But there are some concerns people have expressed about film cameras. One, you need chemicals and darkrooms. Well very nice creative work can be done with them. But, there are still limitations to those types of things. you could only take 12, 24, 36 pictures per roll. These days of course, all of you with your smartphone cameras most probably taking hundreds if not thousands of pictures, whenever you feel like. again, perhaps somebody may argue, this is a great way of actually making the right photographs happen. Having constraints, like being able to take 12 pictures, would actually be great because we'll only take 12 pictures at a time, and not hundreds. But the other thing, and this is perhaps the biggest reason for getting into digital photography and computational photography, is in the traditional film. By the time you take a picture and then you develop it and it takes a long time. These days with displays you get pretty much instant gratification. You can see the results right there. Another part of it has been the sensitivity of film. We'll cover the sensitivity of film versus digital sensors in a future lecture. But some may argue, and actually I would agree, that film is a pretty sensitive medium. It actually can cover a lot more dynamic range, than most sensors can, but sensors are getting better, and now actually there is a race on comparisons between the two. But again, purists may love to use this and I still, support that argument.

07 - Computational Photography Enables Imaging.srt

Now let's look at, in what ways computational photography enables imaging perhaps to generate newer or better types of imaging. One, you know, computational photography may allow you to get unbounded dynamic range, this might be hard for you to see. But in these types of images being able to get a very good viewpoint of the outside, with a little bit of information about the inside, something our eyes can do very well. Cameras have a tough time doing. But at the same time, we can vary a lot of different types of things on a camera, if we bring computation close to it. For example, we can vary things like focus, depth of field. Here basically shown by these two examples. We can play around with resolution, control it as we need to while the picture is taken, at that time of taking the picture. We can control lighting. But basically, we can control a lot of things. Also color, reflectances of the environment, and all of that wonderful stuff. Again, understanding how to do this is valuable, and something we will look it from a technology perspective. But after you master the technology prospective, how you leverage it to do better photography, well I'll leave that up to you. So in essence this supports a creation of a newer medium of photography, which could be enhanced by these technologies. And of course, worse comes to worse, you can always just revert back to the traditional form of digital photography and take pictures as you've done. Question now is, if we bring in more computation, can we do more? I'm being very careful and I'm not using the word better, because I don't want to get into that argument. But the bottom line is computation and photography combined together, will let you do different, and perhaps more types of things with cameras.

08 - Elements of Computational Photography.srt

So now, let's dive in and ask the question. What are the basic elements of computational photography? We start off with things like a 3-D scene. Here, I'm basically showing a nice beach scene here, and of course, what we're interested in is taking a picture of this nice 3-D scene. So, of course, what makes this 3-D scene, is light is emitting from everywhere. Right. Rays of light. Again that's what I've talked about. And each one of them a different color, showing you the details and all that kind of stuff. Well, these rays of light are coming in. Of course, they're illuminated by a source of light. So for example, in this case, the sun is illuminating the whole scene. The rays of light are illuminating this scene. I have now the most basic element of the scene. The word itself is illumination. In essence what's happening is light transport. Light from this thing is coming in, rays of light. And now this is what we refer to as illumination of the scene. Another part of it is, what we want to do with this is, we want to use optics. To be able to now capture, assist in capturing the light transport, the rays of light in the environment. To actually capture the information in the digital world, we need a sensor. So we need to have some sort of a digital sensor. in traditional photography the sensor was a chemical process by film. Which captured the light and ordered the light, and fused the light with chemistry to save it, the information, by doing some sort of photo uptake. Our photo electric was actually changed into energy and saved. In this case, of course, we do this with digital where light sensors are basically saving the light information. Then of course, what we can do is process the image. This would actually be an optional step, but it is an important step also now the current pipeline where all of the sensor information is now processed to give it actually, much more of the information which creates an image. And you can notice in the computational pipeline, this could happen much more actively. In a digital film pipeline, once you get the information, processing and stuff happens later. Another part of it is, after I processed it, display is a general word here I use for sharing information. That is, how do you kind of create an image or print or something like that. So, it could be just a computer display, or it could be the printing medium. So, now we have taken this 3-D scene, taken it through all of these processes, to be able to share the output. And, another important ingredient in all of this, is the user, because the user wants to interact with. It's not just a unidirectional when going user can actually do things like share these images and perhaps also control of it. Now, what you notice is. The user is now much more closer to all of this. they were closer to it with darkroom photography and stuff, but it was a passive process. And this is more of an active process. The bottom-line is, computation can be embedded in all aspects of these elements to support photography. I can use a computer here to illuminate the scene differently. I can use a computer here to control the variety of aspects what optics does to generate and capture light. computation is very close to the sensor, and imagine I can put a multi-, multi-array of sensors to capture things differently. Processing, that's the more obvious one, right? I can just do processing of the images in the computer. Display, well, if this is a special type of an image, I need to have a special way of displaying it. So imagine, for example, concepts of augmented reality and, you know, even holograms require you to have computation right here. And the user, of course, is just to kind of there to use all of this. To then of course, do things like share more images and stuff like that. So these are the main elements that I will be referring to throughout this class. And we will talk about how we going to control, and computationally impact each and every of them in this class.

09 - Elements of Comp Photo.srt

So just to have you folks think for a little bit, as we just covered this, I'd like for you to fill in the six different regions here. Write down exactly what you think of the six elements, that we just discussed, the main elements of computational photography that converts basically a scene with light into an image that somebody can use.

10 - Elements of Comp Photo.srt

Well these are the answers, the main step, first step is the illumination where we control the light. Optics where we take the light, the rays of light, into a manageable medium. Sensor which converts it into digital information. Processing where we take this digital information and do something with it. Display is to how we share the information and the user can do a variety of things with it. So the six elements, and again we'll be referring to that extensively in this class.

11 - Rays to Pixels.srt

So one thing I wanted to emphasize again in this pipeline is what computational photography really is about the whole pipeline. How we take rays of light from a real 3D scene and convert them into pixels. Do a variety of things with. Of course, these pixels are the ones that we use for display. So, just to think about it, a scene, again, is a little, has light coming out of it. And these are the rays of light that we want to capture. Of course, this scene could be illuminated by a variety of light sources. Here, I've just put a bunch of different light sources. Of course, in the actual scene, it's the one grand old sun that illuminates our entire scene. But now let's think about it, that I could actually take this thing and put an aperture, we'll get into definitions of an aperture later, and optics. Basically how much light is getting into the optics and could be controlled. And that now could be used to basically illuminate this thing in a controlled manner. So that basically leads to rays of light hitting the scene but through a process of generalized optics to create a novel illumination source. So in essence, this is a natural scene, lit by the sun. But imagine if this was not an actual scene, but a scene that I could control the lighting for. So I could create a whole computational artifact like this, which controls the light, the amount of light that passes through the optics that control this light. And how that's focused on the scene to generate newer rays of light here. Of course, when these rays of light illuminate, they come out again and that's the one we want to capture. This, we can actually create a novel camera, which basically, again, has some concept of generalized optics and aperture here. But now, now the reverse is going on. See? Light was coming in here. Now light is going in through here, through the aperture into the sensor where we can process it. So this is a novel camera. And novel cameras like this can be used to generate newer forms of images. And again in some instances they will still be pixels to create a display, a newer form of display that we can actually use that to you know, view the scene. Now of course, pixels here is still perhaps not the right term here because and it's assuming it's a 2-D image and output. This could be a variety of things. So we can basically say the output of this may not be a pixel. It could be a variety of different things coming in from this camera. Could be holographic imaging or something like that. So, but the bottom line if you see through this, is now rays of light are the primary rays. And using this computational framework and this computational framework, still using the standard scene. We can now go through this whole pipeline of novel illumination, novel camera to novel, novel forms of imagery. So we basically are creating three different types of novel ideas here. Novel illumination, novel cameras using that to create novel forms of imagery.

12 - Summary.srt

So in quick summary, for the things we covered so far. What I discussed was Computational Photography, the basics of it and how it computationalizes the entire workflow photography. Basically we talked about rays of light being the crucial element. And of course, we're still trying to get pixels. But pixels could have different types of meanings now. In essence, we're generalizing the control. How elements of different elements of photography, elimination optics, apertures, sensor, processing, display and sharing can be actuated by computation. And that's the key. Of course, the goal is all of this would enhance the photographic process to let you generate more novel images. In the next class I'm going to provide a specific example of computational photography, in fact I'm going to give two different examples. These will allow us to do a deeper dive into what computationalizing the different elements of photography are all about. And help us create both a vocabulary and kind of an understanding of what are the aspects of computational photography. The two examples we used extensively, the rest of the term to kind of situate different technologies we're going to be studying. And just a quick thanks to people like the for example the camera 2.0 paper and I also borrowed some ideas from other people who have been working in the area of computation photography.

# 01-03 Dual Photography.txt

1 - Intro.srt

So in this lecture, we will now actually try to make concrete sum of the frameworks we've looked at previously. Remembering again, that using light and using that light to create images. In this one what I'm going to do is showcase a specific example, the example we will look at is referred to as dual photography. Whereby using computer control of a light source and a computer control of a camera, we will be able to extract a newer form of imagery that was not available to us using traditional photography as a medium.

2 - Lesson Objectives.srt

The specific lesson objectives for this lesson are, to introduce to you the concept of computational photography through a specific example. The example I'm going to introduce today is that of dual photography. It's actually based on a recent work by some people up at Stanford and are the collaborators, who kind of suggest that you can now swap a camera and a light source to actually now generate a newer form of imagery. And the whole pipeline of how to do this impacts all the elements of photography that we have covered so far. And the reason for doing this introduction of this concept right now is to kind of have you think about these concepts as we go further into analysis of different methods and different technologies that are underlying both a camera and the digital processing pipeline of images

3 - Recall Rays to Pixels.srt

We already looked at this before but , now I want you to recall the whole pipeline of how rays of light are converted into novel forms of images. have a 3D scene, which is emitting lights. , now we can put our own light sources in that scene. We can control it with generalized optics and aperture. So this now scene is lit by lights that we control. And these are rays of light coming in and hitting the scene now. We refer to this as a newer form of illumination. Course these rays of light are then being captured by a Novel camera. This Novel camera already has, also has optics, aperture, and a sensor. And out comes various forms of information to generate a novel image, a standard one is pixels, but it could be any different types of things like a light field or other types of information to regenerate new images for somebody to see. So this is the pipeline. Now, in this example we're going to look at how we can do both this novel camera and novel illumination. And both of these things would be much different in how they are thought of than traditional photography pipeline would be.

4 - Novel Illumination.srt

So let's first dive into Novel Illumination. What I'm going to do is, I'm going to introduce this concept very simply first. and then we're going to look at that example in much more detail and, and those examples some of these details will be repeated again. So we just have a 3D scene. What I can do is, I can put a photo cell, to be able to capture all the rays of light that are going into this photocell. And that is going to generate an image. what I could do to illuminate this scene, now it's already naturally illuminated. But I could actually do something different. That is I can put a projector, and what this projector is doing is, illuminating the scene, and what we can do is this can be considered a controllable light source, but what we can also do in front of it, is create some sort of a modulator. Which is in essence a controllable aperture. Which would mean is, it would open different regions of this modulated region, depending on what kind of computer code we will send it. So, in essence what I can do is, I can illuminate just this part and close all of it, and therefore the light would come from here, and through here and hit the scene, just that ray of light, as shown here. So, only this ray of light from this projector is illuminating the scene. I can computer control and move this light source within this region. This is in essence what it means by an aperture, is I'm opening only the regions and I know exactly which region I'm opening. And I can move this in a predefined manner depending on what I want to do. We'll see this again in a bit, but I just want to setup the situation. So this is what novel illumination means.

5 - Novel Camera.srt

, what I am going to talk about is not just about illumination, but a novel camera. So now, let's replace this passive photocell with a camera. We can do some similar interesting things. Remember in our generalized novel camera, we had an aperture in front of it. So again, the same modulating aperture we can actually put a similar modulating aperture, which we can put in front of the camera. Both of them can be coded and controlled differently. So I can modulate it by opening the apertures here, and that illuminates the scene. that goes into and the camera captures this

6 - Dual Photography.srt

So similar to before now, just like I opened this aperture, I can open a specific aperture region of specific control, that would only get light through, I mean, in a controlled manner to my camera. So far I've talked about what kind of control to put in here. We'll talk about that in a bit. But imagine that we could do this, and now a novel scene would be generated for this camera. , I can move this, and generate new sources of light, and that would allow me to produce a whole new way of doing cameras. , all of this suggests is now we have a novel form of photography. This is referred to as dual photography, and you can imagine that both the camera and the projector, the light source and the sensor, are swappable. You can actually swap them around, and that's what they mean by dual photography. We'll explain that in a bit. In this instance what I would do now is, I would open a specific aperture and relate it back to what light source was illuminating it. So that way now I can actually start seeing more controllable, what illumination results in what changes in the image. By just controlling these two aspects together, we could learn a lot more. At least, learn more about the scene and how it's illuminating and how to capture it. Question is, in doing so, can we come up with a newer form of image representation that says more about this image than it did before?

7 - Reflective Properties of Ray of Light.srt

Before we go on, let me just show a simple example of what's happening in some of these things. Here, we need to think about, again, rays of light, which were the unique, the thing that we want to actually measure in photography. But we want to also think about that rays of light have various properties, one of them being the reflective property. So for example, if I have a light source that's illuminating the scene right here, and this is a some sort of a surface of an object in the scene. , light hits the surface, and , if it's reflective, it's going to reflect in an interesting manner. So if there is my light sensor, in this case, an eye, it would reflect back into the eye. But we have, we know one thing about surfaces, , that not all light gets reflected from a surface if it has different properties. So for example, this ray of light may also reflect back in this direction. And this, in essence, this kind of thinking about rays of light is referred to as light transport. And we can actually start thinking about what happens with light transport like this. So in this case, reflects off here, comes in, reflects off here, and then come back to the eye also. So now, the same eye is getting this surface up here to it from direct and also reflecting this way. So, reflection of light depends on the kind of surface, what kind of surface it is, could be a specular surface like a mirror, or a diffused surface like a matte surface. So this kind of says is that whenever a light hits a surface, it could actually get to the sensor many different ways. Question is, can we control it? And then, , can we observe this controlled change?

8 - Stanford Dual Photography.srt

So, this was an idea behind this very nice paper on dual photography that was presented at the Siggraph Conference in, 2005, and I'm going to now just talk about it. As I said, this was led by a bunch of people at Stanford with a bunch of collaborators also at different places. It's a very nice piece of work, and for those of you interested more, please go to this link here, and you can see more details about this paper. Just to showcase this example I'm going to play the video from this. >> Our technique allows us to interchange cameras and projectors, thereby enabling us to take images from the point of view of a projector. Suppose for example we had the following experimental setup. Here we have a scene that is imaged by a camera on the left and illuminated by a projector on the right. This is the image taken by the camera, which shows the scene flood illuminated by the projector. In our paper, we refer to this as the primal image. After measuring the light transport between the projector and the camera, we show that the flow of the light can be effectively reversed using Helmholtz reciprocity. This means that we're able to generate an image from the point of view of the projector as shown here. This dual image shows the scene from the perspective of the projector, while the lumination is coming from the position of the camera. Note that this is the image synthesized by our technique. Dual photography is the process of measuring the light transport to generate the dual image. A simple example will help us understand how dual photography works. This scene is illuminated by a projector and the outgoing light will be measured by a photo sensor. Suppose we light up a single pixel at a time in the projector and store the value measured by the photo sensor as a function of pixel location. We do this for all the pixels in the projector. Helmholtz reciprocity specifies that the light transfer will be the same along a light path, regardless of the direction of the flow of light. This means that the same value would be measured whether the light starts off at the projector pixel and goes to the photosensor, or if it starts from the photosensor and arrives at that projector pixel. The transfer of energy from one to the other will be the same in either direction. Thus we can transform our projector into a virtual camera and the photo sensor into a virtual light source. By putting back the measured values into the correct position so the camera image array, we can form the picture that would have been taken by the virtual camera. The resolution of this image will be that of the projector. Replacing the photo sensor with a camera, allows us to capture the four dimensional transport between the pixels of the projector and the pixels in the camera. However, scanning the projector pixel by pixel is very slow, since there are millions of pixels in a standard projector. To accelerate this process, we must identify pixels whose contributions unto the scene can be later separated and illuminate them in parallel. Our adaptive algorithm subdivides the projector image recursively to determine which pixels can be lit simultaneously. This allows us to capture the transport between a projector and camera significantly faster than with the brute force scan. On the left, we show the projector pattern and on the right, we show the image captured by the camera. So now you noticed how this whole pipeline unfolds, and how what the projector is illuminating, and how camera captures the whole concept of dual photography results in a new work type of an image. Let's look at one of the more interesting applications of this, again there are other applications that are mentioned by the authors in that paper that I encourage you to look at. Finally, we perform an experiment to demonstrate that we can capture subtle diffuse interaction. The projector's set up in front of a standard playing card, while the camera's placed so that it can see the back of the card and the diffuse page of a book. In this case, the light going from the projector to the camera had to undergo a diffuse bounce at the card, and another at the book. The image on the right is what the camera can see under ordinary room lighting. It seems impossible that we could ever identify the card by simply changing the incident illumination. However, our framework shows that this is indeed possible. By scanning the pixels of the projector, we can generate the dual image. Here we see that the color of the book changes depending on the point on the card we're illuminating. After the complete transport has been acquired, we can generate an image of the card from the perspective of the projector. So now you see we can actually use this approach to, well, see things that were not directly visible by a camera, but again by looking at how the light transport impacts the surfaces around it and just observing those. So this is one of those instances where both the illumination from the projector and the capture device, the camera, both are being controlled by a computer, and used that to now actually generate a newer form of an image. That's an important part that I want you to kind of understand, because that will lead to the basics of computational photography.

9 - Summary.srt

So, in summary, what I showcased today was the example, of how, novel illumination, novel cameras, and generalized optics and stuff like that, are foundational in the whole principle of computational photography. We looked at one's example, dual photography, that actually showcases how this be done to generate newer images, you are able to see a card, which was not visible directly from where the camera was, but by just doing control illumination and control capture, we can actually still indirectly measure things that were not visible before to generate a novel image. So in the next class, I'm going to introduce the concept of computational photography with another example, specifically we will look at panoramas. The dual photography concept is really interesting and kind of makes you want to think more about the extent of what computation photography can do, more in the future and kind of things it's empowering. Panorama, on the other hand, is much more real, it's something that's available to you already, so for example, if you have a mobile smart phone, it already has software to do panoramas, but the goal for us now would be, is to study, how, images are stitched together to generate larger image and pretty much open the black box of what panorama software does. We'll relate back to the principles of computational photography, and use that to kind of understand more about what we can do with computational photography. Again, I encourage you to look at the dual photography paper mentioned here, and, , thanks to again people that I've been leveraging off their work to kind of explain these concepts.

# 01-04 Panorama.txt

01 - Intro.srt

So my goal with this lecture is for it to give you another concrete example of what computational photography is. We've looked at an example in the past, where we've controlled lighting and cameras to generate a novel image. Here, we will actually make it in the same category, look at another way of generating something called a panorama. Something, again, you've seen enough examples of or may have actually generated yourself. Here we'll open the black box and look inside how those panoramas are built. But more importantly, this will be just an overview, an index of what we're going to be looking at in the future lectures. This will actually allow you to start thinking about what we're going to do next in the next few weeks. because it's going to connect back to what we're going to cover today in this example, and how it relates to different frameworks of computational photography. Again, pay attention to each and every aspect of it and remember that this will connect back to what we will learn in the future.

02 - Lesson Objectives.srt

The specific lesson objectives for this lesson are, to learn about the specific steps required to make a panoramic image, and then identify the elements of computational photography that are used in making a panorama.

03 - Dual Photography.srt

So here is the example of dual photography we looked at before where, illuminated a scene with the computer controlled projector that opened the aperture in different ways, and then we got a modulated aperture in front of the camera to figure out how which light source was related to what parts of the image will be updated. Now remember again, these are the elements of photography, we have a 3D scene and a user, and these are the five phases in between, where we have Illumination, optics, sensor, processing, and display. In the case of dual photography, what we know that we control the illumination, so that's what computational is involved, optics are controlled a bit here and how we control the aperture, both sides, but more on the camera side here. Sensor is being used as is right now, but you can control that also, and then image that is actually computed from a series of things as process and then displayed to the limit.

04 - Panorama Pipeline.srt

Let's look at this pipeline again or these elements again in the context of a panorama, we are going to be able to look at the optics, the sensor, processing, the display, to a user, to get a pretty panorama. So for example, here are the images, I'm showcasing seven images, of a cricket ground, the famous Lord's Cricket Ground in London, England. And if you notice, this was taken by a camera as the camera was being panned from left to right. So we have seven images, one, two, three, four, five, six, and seven. The goal of panorama building is to take this image and generate an image like this, which is a seamless image that shows more detail and space than an individual image there. Just looking at the numbers, these are seven pictures, each picture was about seven megapixels, so this is 7.1 megapixel each image, okay? This is the panorama, this is 1,000x2,000 pixels, so that's 31 megapixel, and more appropriately, the field of view is 151 degrees wide, and 24 degrees high. Individually , you can imagine that, the vertical field of view for each and every one of these images is the same, approximately as that, but, this shows a lot more space because it's got a much wider field of view. We'll talk about the concept of field of view a little later, but, right now, think of it as this showing more, angular space, then each and every individual one which is most really just this much, going spreading out this way.

05 - Taking Pictures.srt

So, first step in trying to do any kind of panorama building is to take pictures. In the previous example I showed you, that , the pictures were taken from left to right. , they could be right to left also. So what it means, is that I take these pictures by rotating the camera. , it's usually much better to actually rotate this camera about the center axis. This is just for demonstration purposes. Let's look at it again. One more time. So an images are captured this way, and want to be able to, do things like, identify the matches. So if you noticed this image is offset quite a bit. We want to align them all together. All of the things here. And then what we want to do is. Merge and blend these images together. We want to do s, align these images. We noticed the structure here is offset. Also these structures seem to have offsets. You want to align these images together, and then blend and merge them. Many different ways could be done. This was me showcasing this by a handheld camera. Which is how I actually took this image. But these days you can use robotic cameras like this GigaPan camera, which is on a robotic mount. And it takes a picture, and moves in a controlled manner. Takes a picture and moves in a controlled manner. So this in essence shows how you can control the camera movement. As part of the photography pipeline. Or the computation photograph pipeline.

06 - Matching to Warping.srt

So let me show you another example of exactly how this kind of taking pictures would be done. And this time around with a handheld camera, which is on a iPhone, iPad or an Android tablet, whatnot. Here we going to use a piece of software from Cloudburst Research, which lets you do this by autostaging. So here you see the video as the person is rotating the camera. Panning the camera in this instance, new picture is taken, they are merged together, and again feature matches and all that kind of stuff is done in real time. So using this we can generate panoramas, just took five pictures. And you know, it's doing the alignment and generating a panorama right there. , a whole lot of processing was captured and you know, controlled to generate this. Another example would be another system again in a handheld system like this here, moving the camera, or moving the handheld device you're generating a panorama. And in this case it's pretty much doing it live also. And stitching things together and aligning them. Again, the reason for showing this panorama as an example, this is an easily accessible example and you have tools already doing it. now let's look at exactly the steps going on in capturing to matching to warping to generate a panorama. Here I'm going to show you the steps as used by this specific software, but these are general steps that would be applied across the board for any method. So, if you notice what we've done in this case is put all of these images. And as I click on each and every one of the image, see that image highlighted so there are you know, six and seven images here. The same images I showed you. And now all of them are a little bit warped, registered, aligned to generate this panorama.

07 - Which Image.srt

Before we go on let me just ask a simple question here. I have these two sets of images, pairs of images. Pair A and pair B. Which of these two image pairs would be used to make an image panorama? Pair A which is shown here? Or Pair B? Please choose the one, highlight the one that you think is the right one.

08 - Which Image.srt

Well, if you had actually highlighted it now, the answer really is that both these images would require some sort of common set of coverage. , meaning is, this image in here have something similar. Similarly, this board here is similar. So, there have some matches between these two that are essential for us to use. However, this one and this one are too far apart. So, these were next to each other. This was one side of the side. The other one on the other side. And yes while there is a green things of common. If you really look at it there is no comparative matches that would happen here. So these two things, if they were to actually match and align they would be wrong. These would be a better pair so, the answer would be , A.

09 - Detection and Matching.srt

So let's look at that one now in a little more detail. The step two is detecting and matching. Here I'm just showing you two different examples. And here we need to do is find features that are common between both. Let's zoom in a little bit so we can actually look at this detail a little bit more carefully. So I'm zoomed in. Now this should give you an interesting sense. These are two different images. Right, because as the game was progressing, even when the camera was being panned from left to right, this megatron the kind of the display of the stadium changed, so it went from lighter to darker. The rest of the scene, , is still the same, but we want to find matches between this and that for doing various types of alignment. Looking at that a little bit more carefully, let's look at these two examples again. What I want to do is now find features. Now, you see a bunch of numbers in both of these images. So, the software I used already found those measures. Now I'm going to find and connect, , specific examples of this. So look at these circles. I just put up five of them. They need to match to other ones in the image, the corresponding next image. So the same five ones and there and what you see is this feature and this feature match, this feature and this feature match, this feature and this feature are a match. All five of these highlighted features have found a match from one to the other. This is an important part of the matching of features from one to the other, because that will be important to align these images together, and then, , merge them. We will cover the techniques of how to do this much later in the term, but I just want to use this as an example of why you would want to do this.

10 - Warping.srt

So, next part is, warping. If you noticed again, in the earlier example I had shown you, not only are these images aligned, they need to be warped a little bit to register them on top of each other, so then actually when certain regions are the same, the images will also be kind of, appear to have similar, detail. So, here kind of shows is that images are warped, so these were, , rectangular images, before aligning them to various types of things, I need to warp this image, this image, this image, and this image, so then they'll register on top of each other. So, let's look at this warping example, I am showing one frame. In the next one, if I was to put them top of each other, while some parts may align, if you notice there's still lot of blurriness going on, none of the crowd is visible, so, to achieve this, I would have to take one of the images and kind of put a warp on it to register to the first image. So in this case, I need to find, what is the between points here to there, so, if you notice this person's head is here on the other example, so I need to figure out how to warp it, similarly, this person here, this person here, you can see the same thing for this building here. They're all kind of misaligned here, and I need to find these warp fields, this corner here with the same, this corner here, this building is also in between two images. So I need to kind, a, and complete warp field between these two and use that to align the two images together, and here I show you how we did this. After I do the alignment, if you notice there's no more ghosting, that's what we want to do. Let me show them to you again. This is before warp. Post warp. And that's the final result after we've done all the warping between the two images, everything is registered.

11 - Fade Blend Cut.srt

Another process between them is once I have these multiple images that actually have overlaps and everything registered, I need to figure out how to blend them, crossfade across them, or cut them. Again, something we will cover in extensive detail in a future lecture. But here I just want to set the context. So again, I have all of these images. Seven images that we've seen before. Just kind of put them on top of each other and kind of shown them to have a little bit of transparency so we can see the overlap amongst all of them. Warping has been done, but now you notice some of the lighting conditions are different. This can be done by figuring out which pixels from which images should be visible. What percentage of the mixture of these two pixels be visible? Or should I just choose one pixel or set of pixels from one image to the other? The fade and blend are merging those two images together. Cut is choosing which one of the original image pixels do I want to choose. This shows the step, now I can merge them and then slowly, see notice that all of the boundaries are gone. And this shows a perfectly clean and smooth panorama. Another simple part, the step 5, is cropping panorama just to kind of create a rectangular image again. Again these are optional. Sometimes I actually like to see panoramas without the cropping because it kind of shows a nice transition across different viewpoints. But this is now a much bigger image. Shows the space much nicer.

12 - Five Steps to Make a Panorama.srt

So the five steps in making a panorama are capture images, detecting and matching the features amongst images, warping the images to align the features, blending, fading or cutting to get the right pixels, and merging all of those images together and then, , cropping. This allows me to generate a panorama like this one. Again, each and every aspect of this we will cover in detail. The reason for this example here is to showcase this whole pipeline and how it relates to the computation of photography pipeline. So then we can come back and refer to it as we develop the techniques later.

13 - Which of these are Panoramas.srt

Before we go on, let me ask a couple of quick questions. Here is an example of variety of Panoramas that I found on the Internet. You can see much more details on these type of things. Question for you is, which of this is a Panorama? And if not, which one is not? Just click on the ones that you think are appropriate

14 - Which of these are Panoramas.srt

Well if you have, you'll notice that this is a panorama. It's referred to as a planar rect, rectilinear panorama, where all the lines are still straight. This is a rotational, spherical, or a cylindrical panorama. As you notice, it seems to be like if you were at one point, the whole panorama is rotating around it. This is referred to as a path, route or a multiview panorama, and this is somebody driving in a car and actually now stitching slices together. And , it looks like something you'll have best views as a move from one place to the other. This is a vertical panorama. In this case, the panorama is moved up and down, or at least the camera is moved up and down to capture height of something much larger. This is not a panorama, it’s a single shot.

15 - Summary.srt

So today what I've done is I've introduced the basic concept of a panorama and presented a variety of steps to make a panorama. And I've related back to the steps of how a panorama making is related to the whole concept of computational photography and of different elements of computational photography. Next I'm going to now, again, push on the whole concept of why are we trying to study computational photography. I'm going to show you different examples of it. And again what I'm going to talk about is why is computation photography and interesting discipline and how it relates to traditional photography. Something I've been covering a little bit in previous lectures but I'm going to try to unify them together in the next one. Just to conclude here the some of the software that I used to generate the panoramas that you saw in this lecture. There are many other differ.

# 02-01 Digital image.txt

01 - Representing an Image.srt

Welcome. So far whenever we think about photography, we think about a process where it takes a camera like this, and captures a image that can be done, used as a photograph, could be printed, or you could do different types of things with. For computational photography, we need to actually add a level where we can take this image and create a digital representation of this image that would make this a computable entity. So in this series of lectures, what we're going to do is take a digital image, and start looking at what we can do with an image and different types of processing approaches on it.

02 - Recall the Comp Photo Pipeline.srt

Just as a refresher, let's recall the basic elements of computational photography. But first, let's imagine what is our scenario. We want to take a picture of a 3D scene, as demonstrated here, and what we want to convert into is a picture. So in essence, what we have done is we have taken the rays of light that are illuminating the scene, and converted it into a two-dimensional picture. Now recall that the main aspects of the computational photography started with illuminating the scene, the optics that were used to then take the light information and get it onto a sensor which was then converted into an image. And then we can do various types of image processing on it. That would be displayed into a picture like this. As you may, may remember, our goal was to computationalize each and every aspect of this pipeline. Today , in this lesson, what I'm interested in having you start thinking about how the digital information, the sensor, can be actually used to generate a representation that we can actually do various types of image processing on. So then we can display an image. So the three main aspects that we're interested in here is how the digital information from the sensor is put on to generate an image that we can do various types of processing on it to generate a picture. We will discuss more aspects of how a sensor does generate this kind of a signal in a later lecture. Today the goal is for us to start thinking about how we can actually start doing image processing and computer vision types of techniques on images. So, the goal that we have in this lesson is to be able to make an image a computable entity. That is, we can run various, various types of computational processes on it, and actually allow us to start thinking of what is a digital image, and what's a representation of a digital image, to convert and create images and pictures that we can use for processing.

03 - Lesson Objectives.srt

Now, one thing I'll be doing throughout all my lessons, is I'm going to start off each lesson with a list of objectives for that lesson. And then we'll try to see if we can match and have you learn about those specific things in that specific lesson. The four specific objectives for this lesson are, is one again, for you to start thinking about a digital image and look at representations of an image with, in terms of pixels, and at the resolution of that image. I will cover basics of how we would actually represent that image as in a discrete form using a matrix, or in a continuous form as a function. We will be looking at the other foundations of grayscale and color images. And I'll also be touching on how digital images and the file formats and stuff like that are used. What is actually in those files that you've gotten used to seeing on your cameras and your computers?

04 - Digital Image - I.srt

To help me make the point about what a digital image is, here I've taken a sample image of the Georgia Tech mascot Buzz, in his wonderful, colorful glory here. What we interested in now actually start looking at and easing apart various aspect of an image, for example, a square image like this one. Just to help us along, I've actually converted this image into a grayscale image, which means that the whole image is now represented not in color, but just the different aspects of the gray values of an image, the , the range from a white pixel or white color, to a black color. And this now is showing a black and white or a gray-scale image. Now, mostly we will be looking at these images and trying to represent them in a coordinate space here. I'm referring to the x and y coordinate space, which allows us to be able to traverse this image in this axis which is x, and in this axis which is y. And the interest in doing this is because now I'll be able to kind of count down in some instances. If you start thinking of this as different columns and different rows of the image. So I'd go down each and every column, until I come here, wrap around and I would actually start looking for other columns going down row by row. So this is one of the representations we want to start looking at, and how do we best kind of represent and image in a form that allows us to, in a raster scan format, traverse each and every element of that image. So now that we know the axes of this image, let's look at the dimensions of this image. So here, I show you the whole box that represents this image. And the easiest way to reference this box would be through width and the height of the image. Width represents the number of columns that would exist in this image, and the height represents the number of rows that represent this image. This is a square image, a classic image, and each of the basic width of this im, image is 512 pixels and the height is 512 pixels. And if you just do the simple math of trying to take 512 width and 500 high, 512 pixels height you get an image that has about 262,000 pixels. And this converts it into a 0.26 megapixel image. And this is an approximate number of pixels, megapixels this image has. So, this is one of the ways of, the simplest ways of representing an image where we need to know the width and the height and how we'd be traversing it.

05 - Digital Image - II.srt

So now, we have the buzz image. We know it has a width and height. we want to use this to give it a numeric representation of this image in two dimensions. So we can actually be looking for information in it in the x and the y directions. The best way to refer to this would be a now a function I xy which is a continuous function. Or if we have a discrete, we'd actually be able to refer to this as I of ij. So i, j here would be discrete indices, which could be the number of the rows and the column that we're trying to traverse this thing. If this was, , a continuous function, we would be looking for variables x and y to be able to now extract the information out of this image. And , one more thing to add here is then we tried to take the resolution of the image, which is represented in terms of width and height as I showed in the last slide. Or just showing here, very quickly, as to an image as the scope of these images, defined by the width of the number of columns and the height of the number of rows of this image.

06 - Pixel.srt

So far I've used the term pixel. Let's try to understand what that pixel is in the context of an image. A pixel, or a picture element, is the element that contains the light intensity at some location within that image. In this case i and j of that image. So for example, what we're really interested in is now we have the scope and the boundary of this image here, which is known by its width and height. But I also want to know within it at any location here what would be various intensities within that image? And I want to be able to do this for all aspects or all parts of this image. So for example at the ith and the jth index, this is the val, this is the pixel. What would be the intensity value at that location? In essence, a computational aspect of all of this is, that I want to be able to create either a discrete representation or a continuous representation that I can now, by giving it values of i and j, give it some numeric value of what this image would be. Because once I get this numerical value, I could use it for various types of computational steps.

07 - How many pixels in an image.srt

Here I've just a very simple quiz for you to start thinking about. If somebody gave you an image, and they said, oh, the width of this image was 1280, and the height was half of the width. What would it be, in terms of the resolution of the image? That is, what would be the size of the whole image? Please put the answer in the box there.

08 - How many pixels in an image.srt

So the answer for this one is rather simple. We have a width of an image which is 1280. The half of the width is 1280 divided by 2. So we need to take 1280 and multiply it by 1280 divided by 2. Which is a result here of 819,200 pixels. That's the whole scope of resolution of that image.

09 - Megapixels.srt

Here is another simple question. You have been you have a camera and it claims to be about 8 megapixels. What is its likely resolution? Which one of those make an appropriately the right answer here?

10 - Megapixels.srt

in this instance what I was referring to was the image from my camera here. So for example, this is a picture I took on my iPhone and an 8 mega pixel is the resolution of a iPhone 5S camera. And if you look at the resolution of this one, this is the resolution of this camera, which is about, you know, 3200 wide and 2400 high makes about an 8 megapixel camera. I should add here that actually if you were to do this multiplication, you would find that the number of total pixels in this image are about 7,990,272 pixels. So 8 megapixel is an approximation of this camera resolution.

11 - Characteristics of a Digital Image.srt

So now, I want us to start looking deeper. And start looking at these images and start seeing what are the numeric values at each and every pixel. And what in different may, ways, what representation do they have? And how can you actually start doing these simple types of computations on it? Here I'm showing a simple image of the mandrill image, the original image here. And to help us kind of look into it, I'm going to actually start zooming into it very closely. And start showing you exactly the details that actually would be seen if you were actually that close up to that image, but also start showing you the numerical values associated with an image like this. So here you note, I've actually put a small red box inside this image. This is a small red box about, eight pixels by eight pixels or something like that. again, I'm showing you pictorial representations of it. Look at it more of an exercise of looking at how these types of images, and what kind of information that we have in these images, less of the exact values. This is just now me showing a zoomed in version with the red region is. The next one I'll show you exactly what the red region looks like, namely this one. Now if you notice here there is a lot of white area and a little bit of a grayish area and kind of a, a dividing line between them, which is kind of where you would see the details of this. And you can see this in much more close-up here now. , what you also should be seeing if the your display supports it, a little bit of blockiness in generated pixel. Again, because these are pixels that we have zoomed in and gotten really close to. If i had some magic, and I do when I actually look at it computationally. I would been able to look at this image and not actually be able to look at this image but the values that exist in each and every pixel. Let me zoom into exactly the level of the red pixel region that we have. And now if you notice, you should be seeing some numbers. And you notice the numbers are like 159, 168 and then lower numbers like 131, 132 here. And approximately there is a little bit of a you know, the same kind of thing here, where I have white values and a little bit of more gray values. Now again, this is just simply a pictorial way of looking at it. These numbers are not exact, even though I've attempted to match it as closely as possible to this sub image here. A couple of things we can start doing now. And that is, start looking at this image a little bit more carefully and see what we can learn about different intensities and different directions. So here what I've done is, I've actually now drawn a line. what I've done is remembering these are our different columns, and in this there are different rows. I found one row, I've drawn this, and I've actually drawn a plot, of the values at that slice. And if you now traverse it, so I'm going to start kind of moving it from this way, and if you notice, there are lots of darker values, , there is a lot of change going on, and all of a sudden, when you come up here, there's a lot more higher values. Remember, that lower values kind of designate more of the darker shades of gray or black, and the brighter values are much more lighter shades of gray towards white. So white comes in, and here again you see a little bit of change, but now this actually starts telling us a lot more about this image, just for that one slice. I also wanted to show that the same slicing can be done in the y axis. So, here, for example, we have looked at the one row, just one row. And if you look at it again, the same kind of stuff if you come in from the top there's a lot of kind of various dark values. All of a sudden we see a lot more white values. coming down to a lot more gray values. So this now starts saying we can start looking at these numbers. Even though the image is shown to us in this form or this form, there are, underneath it, these values. And these are extremely important values for us, and we can actually kind of start doing various types of visualizations of this, and these are just looking at the slices. So looking at this now, you should be seeing that we've kind of looked at an image, been able to visualize it x and y slices. This should start suggesting, and especially looking at this this is . This is a two-dimensional representation where each and every point is a pixel, which has its own intensity value. So the best way to represent this would be a matrix. This already kind of looks like a matrix. It's a two dimensional array. And each and every element of this matrix is a pixel and the value of that is the intensity value there. So an image can be therefore represented as a matrix. And the variance of all the vary, the values vary from zero, which is for black, to 255 which is white. So this is a scale that goes from 255 to 0, 0 being pure black, that is no pixel is actually now showing any intensity whatsoever. And white, where the entire pixel is showing the maximum intensity that is pure white. , anywhere in between would be different shades of gray

12 - Representing pixel values.srt

So here is another simple question for all of you to ponder about. And this might be something, you know, computer science people that some of you are might actually have a very good sense of. How many bits do we need to represent a pixel, if the intensity values range from 0 to 255 that we've talked about? Please choose one of the right answers here.

13 - Representing pixel values.srt

, the correct answer here is 8 bits. And the way to think through this in the binary world is 2 raised to the n, where n is equal to 8. Would give you a value of 256, which will allow us to have values from 0 to 255. Different types of images do have different ways or different number of bytes or bits of information we would use. We could actually imagine a binary image which bit allow only looking at values of 0's and 1's, and that would be a binary image or a black and white image. , we can imagine also having images 2 raised to the 24, where a lot more information is stored beyond just 0 to 255, and I’ll actually be showing an example of that a little later.

14 - Digital Image is a Function.srt

Now, let's actually start looking at how we would represent a digital image as a function. We've already looked at, that I've given all images an axis kind of traversing, and in continuous variables x and y. Or, indices in the discreet space, like a matrix, with i or j. We've also discussed that I want to start representing this image as a matrix, so here is a six-by-six matrix. Now I've filled this matrix in with some values. Again, there are values that vary from 0 to 255. The intensity values of each and every pixel of this six-by-six just a sample image that we want to look at. Now what we want to try to see is, okay, what would this kind of a function or a matrix have in terms of functional characteristics and how would we actually look at that signal? One way of looking at this would be in the continuous store form. And remember when I actually previously showed you this example of just doing a slice and looking at the variables recorded. Actually generate a plot very similar to this. So if I was to just look at the, this axes of six values. And this, , is just a pictorial of a much bigger image. It would start showing your continuous signal varying in different intensity values. And the way we can actually represent that continuous signal, there are various ways that is possible and we will be looking at that in a future lecture. Discrete signal on the other hand would just be these discrete functions at each and every index value with no continuation going on between them. But again, it captures a whole lot of detail in a discrete form for an image like this. Also another way to look at the same matrix, or same kina for information would be looking at it with a height map on an image. So again, on this instance what I do is I look at the y axis and the x, the x axis. I just turned it around just to help us visualize this better. And on the third dimension, the top one, I would be showing the intensities. That's what this image represents. If you look at the same mandrill image, now remember, the intensity values of one or higher zero would be black. So here, for everywhere the blacks are, you see kind of a valley. And then you start seeing peaks and a lot of detail where the whites are. And actually you can see kind of the ridges forming. So in essence, this starts making it look like small hills, and mountains, and ranges, and stuff like that. We'll be looking at it more carefully, and I'll actually be even showing you animated versions of this in a bit. That'll start kind of explaining what are the values of these types of, or what is the value and what are the advantages of this kind of an image.

15 - Sampling and Quantization.srt

So let's look at these two representations again a little carefully. Now, typically to do any kinds of processing we would actually require us to be able to create a discrete representation of a matrix like this. So something like this can be converted into a matrix, as I've shown before. I'm just showing you a six by six part of this. But this entire thing would be represented as a matrix. And what we would be doing is, we're trying to sample the two dimensional space into a regular grid. In this, the regular grid are columns and rows. And that allows us to be able to now look for values that we can actually start looking at as we traverse through. So I can start counting down in raster scan format, going down the axes this way, and wrapping around. And , in essence, what it means is now we have a matrix, just like I showed before, of integer values ranging from 0 to 255. Now my goal is to give you a bit of an intuition of what this image looks like. And what are the kinds of things within it. Again, let's take of this black and white or grayscale image. And I'm going to now both again zoom into right box here, but also kind of show you this image in few different ways. So the first video I'm going to show here is the height map of this image. If you look at it as it's being rotated around, you can actually see the pixel values that I was referring to earlier. And these peaks and valleys kind of show again that the intensity values of the image have various types of statistics associated with it, and by playing around with those statistics we can actually do a lot of interesting things to images. Now I'm going to show the same thing, for this sub-image, again zooming in. Now here again you notice the, gray and the white regions which we have seen before. And now in this video which you see , as I rotate around that see some of the details. And also the, the ridge, that shows up between the, the hill and as we go towards the value which is again the lower values of gray here.

16 - Digital Image Statistics.srt

So in the last slide I kind of talked about that images have a lot of numerical values that we can actually be traversing around on a matrix and a regular grid. Now let's look at what we can do with that kind of stuff. One of the things I want us to think about is the whole concept of an image histogram. An image histogram measures the statistics of the image. In terms of all of the gray values that exist in that image. So imagine if I could. As I have created a bin which has values of 0 to 255. And every time I will scan through this image. And every time I'll say, come down to the first one. I see the value of intensity is let's say 120, I will put one there. And then I will go find another one maybe 125. I might move around and, and it starts mentioning how many pixels have a value of let's say 200? How many pixels have a value of 100? And that will start of kind of creating a histogram of that image and kind of start giving you statics of that image. And this is what this image's histogram looks like. With a peak here, comes down, and closes this way. Now interesting thing to note is there're not a lot of full 100% white values here, and if you look at this image, you might actually see that to be true. There's a few zeros, dark values here. But most of the information is right in the middle. Now , we don't have to do this for the entire image. Now the things to note is that we can actually do statistics on the whole image. We can compute the average for the entire image. You can compute the median, or any other kind of statistical, statistical information for that image. The scope of this need not be the entire image. I could just say is I want to do it for this box. Or I want to actually, even if I could figure out how to come up with this, just this region, what would be the statistics of this, or just the, you know, pixels associated with that eye. , I would have to find the scope, the range of that region to be able to do those computations in. But that entirely depends on us. This histogram is for the entire image. But , in the, histograms could also be for specific ranges. I could do one for this region, this region, this region. And again it starts giving me information about what is the range of information, of the pixel values, the intensities for that specific subpart. It could be region-based, and it could be channel-based. Now I haven't actually introduced concept of channels yet, but that's coming soon.

17 - Color Digital Image An Example.srt

So far, I've only kind of looked at grey-scale images. Let's step back and now actually go back to color images. So, this is the original color image, the mandrill image, that we've looked at before. What we want to do now is start thinking about is, what makes this a color image? Well, , what we need to now start thinking about is, what is a red channel? What is a green channel and a blue channel? In essence, I'm going to claim this, this image is actually divided into, three different color channels. So now I'm going to slowly and slowly show you the three different channels, starting with the blue channel. , each color image is of three channels with their own intensities. This is the blue channel. If you notice here the regions in this image which has a lot more blue, is, , more white because that's where the maximum intensity is. In the green case, if you notice, these parts have a lot more white colors. And in the case of red channel, this part, , is the brightest. So, in essence, these are the three different channels, and these merge together. Blended together create our color image. So this now makes us want to add one aspect addition to the resolution of the image. That is we have a width, we have a height and we have number of channels. So in essence if this was a 512 by 512 image, we would also have three channels. And therefore, actually, our resolution would be 512 by 512 times three. And following from that, each pixel therefore has three intensities. Each and every pixel in this color image has a red color value, a green value and a blue value, and composite of this gives us a color image. I'll recall from last time that each grey scale image is represented by a, a range of grey values from 0 to 255. That is also true for each and everyone of the color channels, 0 to 255. And remembering that we said that we needed eight pixel, eight bit image. To be able to get a range from zero to 255, because that's what we needed. Well, now, if I wanted to have three of them, you can imagine the range is going to be 24 bits to be able to get a much bigger range than we have right now. And that's what we mean by an eight-bit grayscale image or a 24-bit color image.

18 - Digital Image Formats.srt

So now, let's talk a little bit about digital image formats. again, raster image formats, again, the ones that as I said on a matrix, representation scan, row, column lies down to rows, well each and every one of those elements is a pixel. And in each and every image is nothing else but a se, series of colored dots with intensity values. In the last slide, I kind of started getting into this whole concept of number of bits for each pixel represents the depth of color. Well, again, one bit per pixel would just be a two color image where black or white would be represented. Four bits would be 16 colors, eight bits would be 256 different colors. And again if you keep on scaling this, remember, the equation that we looked at was where n. n would be 8 for a regular 0 to 255 image. And just reviewing from last one again, images can be 16, 24 bit, and 32 bits-per-pixel. A 24-bit pixel is usually the one that has 8 bits per color. At the highest levels, the pixels themselves can carry up to 16 million different types of colors depending on again how, what bits of pixels which we're ta, trying to use. Common raster image formats and this is some things which again you've seen from your cameras a lot, is allowing you to be able to look at images in various formats. GIF or GIF is one more prominent one. JPG and PPM, TIF, bpm, BMP, many variety of things exist. I'm not today discussing the camera RAW format. We will talk about that later because that's a special format that's coming in. That actually captures the information directly from the sensor and stores that into a file for later pre-processing kinds of stuff or post-processing kinds of stuff.

19 - Exercises to do on your own.srt

So one of the most crucial things that you have to do in this class is you have to get hands-on experience with playing around with images. And for that purpose, I would like to now introduce to you several tools we will be using in this class. Again, it's essential that you learn how to interact with images, because we are going to be doing a lot of processing and computation on images to take us towards path of doing competition photography. To facilitate this, first thing we will pair on with is something referred to as OpenCV. An interface to OpenCV that's available through Python. Now, or Daniel, the head TA, is the instructional designer for this class. We'll provide you the various suggestions on how to get setup on this kind of stuff either on and stuff like that. But the goal of this just brief introduction here is to introduce to you these tools. OpenCV has become a predominant standard of doing any kind of computer vision or computational photography. It's a toolkit that actually started off from Intel many, many years ago, but it's become an open standard. And its available to anybody who can download it and write code in C++. And now with availability for Python wrap around it, even in Python. Again, please look at these sites and look at how we can actually interact with this. What we are going to provide in this class is both an interactive browser, and I'll show you that in a bit, and also ability for you to download and actually set this up to do various types of processing of images for computational photography. Several of the assignments are going to rely on you to do this kind of coding in this environment. And share with us as part of different metrics of success of how we can actually accomplish these goals in this class. The next toolkit and again is a much more widely available as Matlab. This is a predominant tool for doing processing of matrices, but and you know images can be represented as matrices and it becomes actually a very widely used tool for image processing. So if you get Matlab you can actually also get the image processing toolkit, and allows you to do a lot of different types of things. I'll showcase that in a bit. For educational purposes it's available at a student discount. again, you can get that from the MathWorks website. Again, I am not getting any financial rewards for recommending this. It's up to you to do so. there is a public version that is somewhat similar to Matlab also available called Octave. my hope is that you will interact and play around with these tools to kind of help you create the building blocks that create the machinery for doing computational photography. Another toolkit that's become widely used these days is called Processing. We will not be using much of this in this class, but I just wanted to introduce this to you a little bit. And so again a Java based setup that lets you actually play around with images and manipulate images and videos and stuff like that. Let's look at all three of these very briefly first. So this just showcases an example of what we can do in a browser using OpenCV. But you can use this extensively on your own workstations after you've downloaded both OpenCV and Python on your workstations and got them installed correctly. Again just by doing simple coding like this we were able to run and do processing on images. I've just showed you, you know, smoothing an image, grayscale, edges, and cropping an image. And all that kind of stuff after I did the test run. Showed all of the examples on the browser. When you run this the first time it will be a little slow, and you'll get, get better as you use it, but really doing more complicated things that we do in this class will be much better if you do it on your own computer, not on this browser-based stuff. And you'll be able to actually, you know, save your work and all that kind of stuff, and in track with the code that you develop. So this will become a, you know, tool that I'd like for all of us to use in this class. And this the primary tool that you'll be doing your assignments and stuff. I'm here now showing you just a screenshot from MathWorks Matlab. Again this is a tool that you can also use for doing a variety of things and it becomes a very interactive tool for being able to kind of just load images and manipulate images. For example. So here I'm typing in that I want to assign to the variable buzz image content from the file buzz3.jpg. Once it's there I can actually show this image. Which is right here. So just by interactively doing these types of things I can now generate or see images and then I can run various types of processing on these images. One of the beautiful things about having it in Matlab is that you also can do variety of things by just you know, using the mouse on it and stuff like that. Very handy tool. Again, Octave provides very similar facilities and I encourage you to look at both the MathWorks site and the Octave sites. Again, I do want you to make sure that you're picking up on these concepts on your own. We will not be getting into a lot of these details in the class. again, these are introductory materials which, and there's lots of material out there on the web for you to get familiar with these things. Finally, this is processing. I, I'm just showing you again the main console window for this one. I've just inputted in a simple code. Most of the time you'll have two different types of things. One, do us a little bit of a setup. Here are says the size of the file. I'm actually loading in this file, and then displaying this image. And we can just run this and it displays the image. Again, we can now do a variety of regular processes on this image because what, in essence, we know is how to display an image which also mean now we have the entire image in our code to do a variety of things with. And once we have code we can interact with this code to do a variety of things. For example, here I am just going to now change this code, run it again. And if you notice what I have done, put half of the image of buzz here at this point here, so. Simple processing like this can be done, in this one we'd be actually leveraging Java style code. Three simple tools for you to play around with images. Again, I want to emphasize that we will be using one of them more extensively. So I just briefly showcased three different tools. again, I want to emphasize, this is a tool that you can use in the browser, and also interact with on your own computer by installing the tools. Please look at the OpenCV and the Python sites and get familiar with these things. We will provide with various types of recommendations how you can get set up on this. Welcome to use and get access to MatLab and Octave if as, as much as you want. Those of you interested in doing these kind of things with processing, you can feel free to do this. I do want to remind everybody that most of the assignments and stuff that we will be providing would be actually in this domain and we'll also be providing some sample code to get you started.

20 - Read and write images.srt

Couple of things I want all of you to start playing around with if you can, and it would be great for you to start doing this on your own if you have the old Python-OpenCV installation. we will be providing a browser-based way of doing some of this stuff, the simpler stuff. So, you can actually try to do testing of loading images and stuff like that on your own, within a browser. But again, we do encourage you to set up a whole platform for yourselves on your computers so you can start doing some of this on your own. I would encourage you to play around with these types of things on your own. On Python , or OpenCV you can import the OpenCV framework by just typing in import cv2. And you can read input images of any form and also write them out. And play around as I said, different aspects of doing these types of things. In Matlab the same thing is possible. On what you can just read images and you can also write them out. There are other functions you can play around in Matlab including, for example, showing the images. But I encourage you again, to play around and get used to this. I'm not talking about processing in this lesson here. But again, there is a lot of documentation available for processing online. Just a couple other pointers here, you can actually start looking at the sizes of images, color channels, bits per pixel information. Again, both in Matlab and in OpenCV Python, there are ways of being able to do this and seeing what kind of details you can come up with.

21 - Understand image formats.srt

One thing I want all of you to do is start looking and understanding image formats a little bit. Look at how the color channels are distributed. And also what information we can extract about image compression. And also these days, , most camera formats have a EXIF, or Exchangeable Image File Format. Which has information including , the width and the height of the image. But also things like how it is coded and what are the compressions associated with it. But more importantly these things also has information like the date the image was captured geotags, and other information that's added in as an additional data structure with an image data cell.

22 - Summary.srt

So the end of this lesson, I just want to quickly summarize what we have covered. I've talked about what is a digital image and how can you make it into a computational object. We've talked about different types of formats of images, black and white, color. And I've talked about histograms and other ways of extracting simple statistics of an image. we have not talked about what they would be good for, but that's coming up soon. So, in the next lesson we will actually get into is now, once we know what we know about images, how can we do simple processing and filtering. And actually that's where some of the interesting things that I want you to play around with actually becoming really important. So far in all of the lessons we've looked at right now we're just looking at the pixel itself. We'll also be looking at what happens in the neighborhood of the pixels and how do you actually do combinations of information one pixel to the other.

# 02-02 Point processes.txt

01 - Intro.srt

So, in the previous lecture, we started looking at, how are we going to represent images as digital representations? Now, let's start looking at what we can do with these images. I'm just showing you again, a color and a black and white image. Now, what we want to do is we want to start looking at individual elements, pixels, of each and every one of these images, and start looking at what we can do by simply doing point-process arithmetic on it. That would allow us to start combining one image to the other.

02 - Lesson Objectives.srt

So the objectives of this specific lesson is that I'm going to introduce you to the concept of point processes. That is, how do you use pixel values themselves and use them in computations? We will use this to do addition and subtraction of images but it can be used to doing any other kinds of mathematical operations like multiplication. I'm going to use this also concept to introduce the concept of alpha values and alpha blending. And how can we use some different types of simple image processing methods. And while we have talked about image histograms in the previous lecture, in this lecture I'm going, I'm going to give you a much better handle on how we can use image histograms to look at the statistics of images and how we can use them in different methods of computer vision and image processing

03 - Recall Digital Image is A Function.srt

So recall, in the previous, lesson, we talked about using a digital image as a function, or representing a digital image as a function. Now what want to try to do is, I want to try to, bring in the concept of how we can use that, for simple forms of image processing. Recall in the last lecture, we talked about that an image is a function. Which is represented by various ways of scanning through it and x and y and also looking at the intensity values which would be the height map of an image like this. Using this we constructed and stated that now we can actually create a simple grid, like this, to be able to represent an image, and now we would be able to pick up the values from each and every element of this matrix. So, just to review this, what we're trying to do, is we're interested in extracting discrete values of an image. So what we want to do is sample the whole image, in the two dimensions, in this case this matrix here. And look for the values off of this matrix to be able to do any kinds of computational processes on. this is approximation because it's quantizing each thing into a districtized sample and what we now have integer values which we will be then looking for to be able to extract the kinds of values from each image, which will then be using in different computations

04 - Point Process.srt

So now let me actually go into the details of how we're going to do point processing with an example. Here, I've come up with a, an illustrative example using a small image, a small matrix here. , it's a six by six. And if you notice, it's filled in with various grey values or values from zero to 255. 120 intensity value, 121, 122, 125, 126, and so on. Again, it's a small image. We can now to do point arithmetic. Let's also consider another image, the second image which is shown here. So now I have two different images, image one and image two. What I'm interested in is now doing a simple addition of the intensity values of these two images. So for example take the first one and the second one and add it together. Here is what the result would look like. And again just to kind of look at it we can go through the math here. 120 plus 120 results in 240. 121 plus 121 results in 242, 244, 250, 252. You know, , again these things are repeated, so they're just multiplied by 2 here. And you can see the same kind of impact here on this row. Let's actually look at this example. 140, and 140. Well, when you do these types of additions, the answer is 280. And similarly, 142 and, and 142 results in 284. And 143 and 143 times results in 286. So if you notice, there's an interesting problem here with these three pixels. All of their values are greater than 255. So we need to now actually do some other types of processes to help with this. Because one, I cannot, as we've learned about it, image can only have values from 0 to 255. So we need to start thinking about what to do with images when you start getting values like this, and part of the secret is to be able to kind of scale this up to these values become 255, i.e., white, and the rest of them get reduced appropriately. And the scale from whatever is the lowest value to the highest one, to be able to capture the change of the ramp of the image. Same example, but now let's look at it another direction. So again, I have the same two images, except now I'm going to subtract one from the other. So in this case, , if you notice, 120 subtracted by 120, so this whole. Our column of pixels is 0. And similarly since these two pixels are exactly the same, here again we get 0's. Except now, you get an interesting problem here. So the values here are 11, 10, 13. Here are 151, 152 and 153. Which means these are negative values. Again, remember from the example we looked at before. We can only have values from 0 to 255. I cannot have negative values. So these are black, which means that they are no information, so I want to replace these with 0's, but I will also have to do an adjustment on all of the other ones. To go from 0 to 255. So in this case the range would be, this is 0, this is actually a very high number. So, , either we could lose the information or still capture it by doing some sort internal scaling of these images.

05 - Pixel Point Arithmetic.srt

Now let's build on an example using the simple point arithmetic. Let's say I get this image, image one. Here I have Einstein, I was coming, visiting the office, and he's sitting in front of, on the desk here. And now what I actually want to also do is kind in tract with this image. But Einstein, actually, before he visited, I was able to take a picture of the same location before Einstein came in. So now I have two images, Image 2 and Image 1, here there's no Einstein. There is Einstein. So now the question is, by just doing simple point arithmetic, or pixel additions and subtractions and such at each and every pixel. Again these two images are the same size, can I actually now find interesting information between these two images? So by just doing subtraction of this first image from the second image what will we get? We will now be getting all of the pixels that actually have changed from first to the second. So this actually becomes an example of almost doing something which is referred to either as change detection or you on background subtraction. So using this I should be able to figure out all of the pixels that are in this image, that are also in this image by subtracting it. But then all the pixels I should find are the ones that have changed from one to the other. , in this case, I've taken the exact same images, the location of each and every part of this image is the same as this one. That is, you can imagine this camera was on a tripod, did not move at all, and was a controlled situation. So , they're exactly the same background, and therefore I can do background segmentation or background separation. So by just doing a simple point arithmetic of Image 1 by Image 2, I get, just all of the pixels. Again, I haven't, I mean, this is an illustrative example. I haven't actually done clean-up or anything else like that. But you'll notice that all of the pixels that get highlighted are the ones where Einstein is. So now you start seeing the pixels over here, and these are the ones I wanted. I wanted to find all of these but I did not want to find these. So, now I want to find all the red parts, but not the green ones. , this gives me that result, gives me all the pixels, and , now I can do, start doing various types of processing on this one. Here's an example. One, I can just look for the ones and zeroes. That is, everywhere there is a value of certain things, I say, okay, make it be black. So all of the smaller values become black. All of the bigger values that are closer to much more you know, the white of things, become white. So this becomes 1, this is 0, becomes a binary image. And I can run this process, between, the subtraction that I have, between the two images. And this allows me to now find pixels, which are where Einstein is. Now, I know those of you paying attention to this, you should have gotten the head and all that kind of stuff correctly. It kind of implies that in some of my computations I didn't pay attention. Again, I'm showing this as an illustrative example more than anything else.

06 - Pixel Operations.srt

Let's look at another example of pixel operations. Here I actually want to now up the ante a little bit and look for some fun examples, and see what we can learn about how we can actually make progress by just combining a bunch of images of the same scene. Again, taking my example, in my office one day, you know, Charles Darwin shows up. Another time, Albert Einstein is there, and another day, Leonardo Da Vinci is present. Now what I'm interested in, is I want to actually have these three giants of science actually be together. Well how would I go about doing that? Well one simple way would be is, again these three pictures are all , of the same size same information, same colors and everything else. That is the same color range and everything. What I could just do is simply add them. Well, let's see what happens if that's what we do. This is simple addition of all three images. Now if you notice, and now you must have guessed this will happen, if I just do simple addition, all of a sudden a whole lot of values, and I do this again, at three different layers of RGB and their intensity values, and I combine it together, a whole lot of values go over 255. When a whole lot of values go over 255, and I've not done any rescaling here majority of the image, , becomes white. What's the best way of analyzing an image like this and seeing what the dynamic range of an image looks like? Remember, we have values that go from zero to 255. In this case we can imagine the range is between zero and 255, and most of the pixel intensity, pixel bucket, some around middle. Same thing is true here, same thing is true here. But for this one, there are a lot more whites. , the best way to analyze this would be to look for a histogram. The original histogram of this image shows that if zero, this is zero, this is 255, most of the information is in the middle. Similarly, Albert Einstein, same case here. Alot of the information, intensities are in the middle. In this case the same thing is true. Alot of the intensities are in the middle. But what happens when you look at this? If you notice, a lot of the values are here, but the peak is right here. And majority by a long shot are the pixels are, , white. That tells us yes, this addition would have required us to do some sort of scaling. Now I could do the scaling now, but most of the information would have been lost if, unless I do it at the right time. One of the ways of doing this, , would be to figure out how to do the combination of these three images a bit differently. One proposal would be to be able to combine the three images, CD, AE and LD, with a different weighing function in front of it. So in this one, I'd do a multiplication of all of the elements individually with 0.34. The other one with 0.34, and also the third one with 0.34. And this is what I get. An interesting thing to note here now,is the three giants of science are, , now transparent a bit. You know, maybe more appropriately also have a little bit of a ghostly effect, to showcase that they are now together, but they are only partially visible. So how, how can we actually understand what happened here? So what really happened here was, that by combining these and giving it different ratios and the same ratio of 0.34 here, we now merge these things and giving it a little bit of mixtures of these intensities. Now let's look at the histogram of this image. The histogram looks, , much similar to the three images that we used to construct this. Most of the information is in the middle. There's a little bit more white showing up, but otherwise, it's you know, a decent image. , the artifact does remain. The three personalities here are a little transparent. That's the word I was looking for here. They're a little transparent, because you can see through them. So that's the thing that we wanted to get out of this. By doing these pixel operations, and by combining them with a number like 0.34 before I do any of the additions, I've added a little bit of transparency.

07 - Alpha blending.srt

So this transparency is what I want to use to kind of introduce the term of alpha-blending. Let's see what that means. So by doing a multiplication that I have done here with 0.34 before each and every image, what I've done is added kind of a do a mixing of different types of things. So what I've done is I've made the original image, compared to the next one, transparent by 34%. So in essence transparency is what an alpha is referred to as. So in essence what actually that means is I've actually converted and given each one of them and made each of them transparent by 34% from the original image. This is referred to as an alpha and it usually varies from 0 to 1 whereas 0 is completely invisible. Remember if I multiplied by 0 here you would not have seen Leonardo Da Vinci at all. If I multiply by 1 you would have seen him completely visible. it would have had changes to your dynamic range of the pixel values here, but this guy, Mr. Da Vinci, would have been perfectly visible. And similarly I could play around with these numbers for all of them. So alpha varies from 1, 0 to 1, where 0 is invisible and 1 is fully visible. So, in essence, one of the ways we could do this is now add another layer. So RGB up to three layers of the image or three channels. An alpha could be another channel, which would have values from 0 to 1. And RGB would have, , then since the values from 0 to 255. This additional number, alpha value, could be also be computed and put inside those channels or kept separately and actually dealt with as a different mask. And this is an interesting way of being able to now create a separate mask, which we will then be able to play around with for different applications. Again, something we will come back to as we look at many examples.

08 - What is the result.srt

Just to keep us going, here is a simple quiz. Here is that, you know, two images. And I just want you to simply play around and do the computation to be able to generate another image that actually has these types of processes. So what that would mean is, first I would, to get the value here, multiply the number 36 by 2 then subtract 36. In this case , the answer would be 36. So imagine doing this across the board for all of them. 24 times 2, that would be 48 minus 78. That starts kind of getting interesting again. So, one of the things I want you to do when you do this kind of stuff is also do the truncation, so you keep the values from 0 to 255.

09 - What is the result.srt

Here is the full answer for this, solution. You have, or I have, for some things I have truncated the value. Again, if you noticed, the 0 is there because 2 times 28, 24 is 48 minus 78 would have been a negative number and I have placed a 0 there. Similarly, the zeros are to kind of account for those negative numbers. In this case none of the pixels went to full, to 55

10 - Summary.srt

So in quick summary, in this lesson I covered the basic concepts of how we can take point processes, how computations at the pixel values, and use that to add images, subtract images. I showed you how to use this to be able to do simple processes on images. Showed an example that by just doing this we actually can learn more about transparency the alpha blending, which is used commonly in image processing types of techniques. Alpha blending is also widely used to be able to represent the transparency layer which is actually another processing thing that can be used for masking images. I showed an example of how we would use an image histogram and all of the statistics that come with it to help but look at images a little bit more carefully. In next class, what we will do is now we will look at not just pixels. So, you know, we were so far concentrating on images where each index is known. We had an image. And I looked at a pixel. , so far in this case we took another image that had a pixel. But now we want to also start looking at is what happens in the neighborhood around the pixels. And that will let us introduce us to the concept of convolution and correlation. Look forward to it.

# 02-03 Blending modes.txt

01 - Intro.srt

So in this lesson, I just want to add a simple concept to, what we've already discussed when we actually covered the whole concept of point processes and simple arithmetic operations in the pixel values between multiple images. I'm also going to showcase a real example of something that you'll witness and perhaps are bothered by, as, and something that you know, hopefully I can explain. So another title for this lesson could easily be, What is with all those weird artifacts in the lecture videos? So we are going to cover, and understand those. First let me show you the videos that you have been seeing. Right, when you see these videos you are seeing, various kinds of artifacts showing up in my hand, goes over it and I am writing things down, on diffrent types of things and you can see all of the effects but, you know, this is because. These images of these videos that you've seen are actually a result of combining things from multiple different types of images, and compositing or blending them together to showcase things. So again, I would like to explain to you why you are seeing some of those artifacts that you are seeing in these videos. Again, they relate exactly to topic at hand, of doing pixel arithmetic operations.

02 - Lesson Objectives.srt

So, the objectives for this lesson are for you to learn about what is happening with those videos. And more importantly, also introduce to you the concept of blending modes that are in wide use with all kinds of applications. If you use Photoshop, you, you'll see them in the layer mode for example.

03 - A bit about the setup.srt

So, let me actually show you first, the set up that we use for this class. So, , I'm not trying to get any sympathy points from you, but in producing this class I spend a lot of time in a dark room like this working with you know, a tablet that's attached to a computer. And what's happening, there's a camera above. part of the content is on this computer. Which is then displayed on this tablet. And using a pen like this, I write on top of this tablet. And then a camera is capturing the images of also what's going on. So in essence, there are two different images being captured. One from the camera, which has whatever I'm doing with my hands. And then another one, which captures all of the writing I do here. So in essence this screen, that's actually shown here, is also captured. On this computer, and then this camera captures everything from the top down of what's going on. And the final artifact is the combination of these two images.

04 - The final output is a pixel blend.srt

So what Aaron, who's the video editor for this class who's been working with me and doing an amazing job so far. What he does is takes these two videos. One from the camera that actually has the hand, and the other one which is the screen capture of the screen itself, which has also my hand writing on it. These two are merged to generate an image like this which is a combination, a blend of the screen capture and the camera. And it's because of this you kind of see artifacts like this where the hand seems to be sometimes in front and sometimes in behind that image which actually is on the tablet itself. the reason to kind of have this is we do like to show you the hands moving around interacting with different aspects of the image. So, this hopefully is not confusing you so since you've seen the hand and the pen several times. So now let's try to kind of build on this concept and switch to what is going on

05 - Blending Pixels.srt

So what we're interested in is taking two images, these two images are perfectly the, aligned, they're the same size, and so in essence this means that each and every pixel in between these two images is the same, except one gets the hand, the one that is getting from the camera itself. And the other one gets all of the stuff that's on the screen. Since they're exactly the same size, and same aspect ratio and aligned. What we , not, need to do do is take the pixels from this one, and merge it with that one. So this means is if I refer to this. As image a, and this one as image b, which means all pixels in this one would be referred to a and therefore what means is I have a new function now which takes all the elements of a and b and does various types of mathematical operations on it. For example, it could be just a simple average, where I take a pixel from a, take a pixel from b. And divide it by 2, and give me a new value. And this is what the output of that one would be. So this one would be also where you kind of see transparency artifacts. That you already kind of know about. Because in essence by doing this, you've kind of half the. You know, creating half alpha, that is 0.5 alpha if we generate pixels from this image, 0.5 of pixels from this image, added them together to generate this, , that's why some things look transparent. Here, , you see my hand is much lighter color, and the, and the pen also looks a little lighter. , a normal form blend really would be just take the base layer, b, and not actually part of anything from a. But , you can now image this starts giving us lots of additional tools how to do this.

06 - Arithmetic Blend Modes.srt

Let me show you some simple examples of Arithmetic Blend Modes. For example, I can do a divide, which lets me brighten photographs. I can take two pixels of information from each one of them and do some sort of division and then scale them up again. I can do addition, and we've seen examples of this in our lecture where I've taken two images and just added them together. In this case, rather than looking at images from the values of zero to 255, let's assume they're from zero to one. And if I have two images over pixels with more than value one, they'll all become white. And in this simple addition we'll let that be the case so it, , makes the image have too many whites. Subtract is again something we've looked at. And in this case again for the range of zero to one, and we subtract, we get a lot of values below zero. And we just replace them by zero, and this image shows a lot of blacks. A most important type, and again we see an example of this is when we do a difference. But we do subtract but whenever we get to zero we kind of do a scaling up to make sure all of the range of values between zero and one are covered. The more wildly used simple blend mode is that of darken which is shown by this equation here. Which takes the minimum of the pixel value from the top and the bottom layers for each and every color channel, R, G, and B. In essence, what darken does, it creates a new novel image that creates represents each and every pixel that retains the smallest component of the foreground and the background pixels. Smallest meaning, , the darker. And therefore it's referred to as darken. It actually starts giving you more of the darker pixels that's a combination of both and finds the lowest value between both of them. On the other hand lighten does the opposite. Where it takes the maximum value. it attempts to select the maximum of both the top and the bottom layers in RGB and showcases that and actually allows you to see a much more lighter image. Again, I encourage you to try these types of things out on your own.

07 - Advanced Modes.srt

Other advanced modes that exist are, multiply, which takes the pixels from both a and b and just multiplies them. The output of this kind of an image is much darker, again. Another popular method is screen, which actually is represented by this equation. And if you look at it, what it is doing is that screen blend mode. The values of the pixels in the two layers are first inverted. Inverted by again as I said in this case I'm showcasing only values that are from 0 to 1 and in this case, inverted because 1 minus a would invert it and similarly inverting it for the bottom layer, 1 minus b. And then finally again multiplied, and then inverted again. So the, in essence, screen is the opposite effect of multiply. By doing the inversion, in this case the inversion is done by subtracting from y, 1. Again, these are images. I emphasize that I'm showcasing only images that are from 0 to 1, not 0 to 55. As we talked about can take an image from 0 to 255 and scale it and space from 0 to 1 also. the output of this is a brighter image. A very popular method is overlay, which what it does is combines a little bit of both multiply and screen. So in this case, the parts of the top layer where the base layer the bottom layer, its light becomes lighter. And the parts where the base layer is darker, becomes darker. So, in essence this kind of combines both those values and actually gives you much, much nicer way of looking at images again. all of them are kind of trying to give you a novel pixel.

08 - Dodge and Burn.srt

if you're a photographer you already know terms like dodge and burn. Dodge and burn are techniques that actually are inherited from traditional photography done in the dark room. Dodge builds on the screen mode concept we just covered and is much more aimed at lightening an image. While burn builds on the multiply and is much more aimed at darkening an image. And actually, in the good old traditional photography mode, it's something equivalent to burning a negative before it's been processed. Again, I'm just giving you a simple set of examples of these types of modes. There are numerous other examples and variations linear dodge, linear burn that are actually not completely driven by screen or multiply. There are many different variations again, if you remember, that all we're really trying to do is come up with the equation f of top layer and bottom. Any processing I can do on a and b can actually allow me to generate a new blend function. And that's the strength of this approach by just getting the pixel values and getting every one of them, you can come up with various ways of blending pixel values from one to the other. And that creates a foundation for a whole lot of stuff that we will be looking at.

09 - Darken.srt

To conclude, I want to make sure you understand that in our work right now. In this example, the kinds of stuff that you're seeing on a screen. You're seeing lots of example of darken, because we want to be able to kind of show you everything on the screen, especially since our base screen is white and we do have lots of text and images. We have chosen or at least Aaron has chosen, to actually apply the darkened process between the two layers, the top layer coming from the camera and the bottom one coming in from the screen capture from the tablet. And this allows us to kind of showcase the best possible image that actually shows all the information that's in combination of these two screens. Yes, in doing so sometimes. Unreal artifacts like this will show up and you can see why that's happening again there is a little bit of a dark blue color here. Which when you go into the RGB space with the skin color of my hand actually does have a little bit of this artifact or getting a little bit of darker values. So again remember, this is done for each and every channel RGB separately. So hopefully, you won't be minding these kinds of artifacts but hopefully, also using that as a learning experience.

10 - Summary.srt

So just to conclude I just introduced a simple concept of pixel layer blending. As actually you've seen now in the context of the videos that you've been seeing. But again, the bottom line is there are many different techniques like this, widely in use in photography. For those of you interested. Just go in to Photoshop, open up the layer mode. And then, in the layer mode, when you actually have multiple layers, there is whole bla, you know, layer blending options that'll allow you to kind of play around and see what the artifacts of each and every one of them, and what effects you get from each and everyone of them. again, showcase why some of those videos that you've been seeing. Are looking up, and they will continue to look up for further information, you can look at Wikipedia site on blend mods or you can actually look at the pipeline of udacity videos producted from the blog site that I have listed. Again,ah, you know start playing around with this kind of stuff using any of the software that we have provided, you should be able to do this kind of simple additions, and simple multiplications and other types of things. Again this was an example of us opening a black box, and that's what we just did for something you've been seeing. Thanks

# 02-04 Smoothing.txt

01 - Intro.srt

Welcome. In the last lecture we looked at point processes of an image that allowed us to do simple arithmetic, addition, subtraction of pixels from one image to the other. Let's now look at the whole concept on how we can take a pixel and look around the neighbor of that pixel and use that information to improve the quality of an image. For example, by looking at and averaging around a pixel, we can blur an image. We can also do other things like, if there is noise in the image, we can look at the neighborhood get rid of the noise in an image. So those kinds of processes let us now look at neighborhoods of pixels within the single image of multiple images, and use that to enhance the quality of images. That's the kind of stuff we're going to look at in this series of lectures.

02 - Lesson Objectives.srt

In this lesson, I hope to be able to cover aspects of how we can actually now start looking at neighborhoods of pixels to help us do, among other things, like smoothing of an image, to be able to extract more information from an image, or just be able to enhance the image quality. The two things I want to be able to cover in this lecture is one, I'm going to talk about how we can smooth an image by using a neighborhood of pixels on an image. And also I'm going to specifically talk about an application of this, which is using median filtering to be able to among other things, remove errors and smooth out an image. Median filtering will be another instance of a method where we would be looking at a neighborhood and running a statistical process on that neighborhood to create a, a newer rendition of an image. this will go towards enhancing as how we can do filtering an image at processing to be able to help us extract content from images.

03 - Digital Image is a Function.srt

Remember that we've actually started talking about digital image is a function. And this is something we've covered in the previous set of sub-lectures. What I have introduced in those lectures is that we can actually talk about an image as a function with we can actually look at an x and y axis with the intensity in the third dimension. That has allowed us to create images, or looking at images like this one, the mandrill image we have looked at before. And the bottom line of all of this has been that we want to be able to represent an image as a matrix. Where we can now traverse through this matrix in discreet Indices i and j to be able to get any kind of value that we would be interested in to then for example extract content. And for example look at, the intensities at specific values, and do some sort of processing with it.

04 - From Single Pixels to Groups of Pixels.srt

Now, so far we have only looked at a specific index within an image matrix and done all kinds of mathematical operations, point processes of doing, for example, additions of this one to another one. So in essence in this one by looking at the index 0 1 2 and 0 1 2. I'm just looking at this specific pixel and looking at the intensity of that. What I want to now introduce in this lecture is that we can actually expand this. Not just looking at the one pixel but it's neighborhood. So here, for example now, is a three by three neighborhood around this pixel. We can look one, in all directions around this central pixel. And , now we have a small sub-matrix which is a three by three sub-matrix. So how do we actually now start using information, not just at that point but around that specific point. So the question now we're interested in is, how do we smooth the signal? Again, an image is represented as a matrix with the intensity signals now inputted into each and every, element, which now I can transverse by looking at the indices i and j. Now we're interested in saying okay, I'm going to take this three by three neighborhood and smooth out the value here with respect to what's happening around it. To help explain this, let's take this simple example of a 1D Signal. This is the original 1D Signal and as we traverse through it, you see simplify this, to be one of the rows off a image. And these are my intensities. And that's just for simplification just we'll take this as a 1D signal for now. Our interest in is that I want to be now able to get rid of some of the noise in this signal. Here I'm showing you a smooth version of the signal. what I've done is now in essence run a process which looks at different aspects of this image or this signal here and says well this is too low. I need to move this up a little bit based on how I can smooth this curve or this whole image that would actually be represented here. Making assumptions like sometimes I want to do this smoothing because this might be an error or actually just want to smooth the signal out. One of the best ways of doing a simple smoothing of a signal could be that we can actually just take the average of the neighboring values. So I could for example look at these four si, four image values and intensity values and figure out the average of four or five of them and replace it in the middle. So here I'm showing a simple example how would I would go about doing an average. So this would be a moving average, what I'd take is I would take five values and , sum them and then divide them by five to get the average of the five, and replace, this element to have that value. And I would do this by moving one by one, as I would go down these indices. Another option would be is not just to do a one by one but actually, or just not doing a summation of equal things but change the weights around. Here is an example of that instance where now I give more weight to the one, the value that I'm actually changing. And less weight to the ones in the neighborhood. So in this instance, I have a flat signal that I'm actually also averaging, equally giving all neighborhood values equal weight. And this one, I'm actually giving it a little bit of a ram. And to sum this back and normalize it. And this one, I had five of them. I divided by five. And here the sum is 16, so I divide by 16 to be able to normalize the instance back to, the neighborhood that it's in. And this allows me to do smoothing in different ways these are moving averages, and move from one pixel to the other, and start doing this. I'm showing this in a simple 1D example.

05 - Smoothing using Average.srt

So now let's look at a 2-D example of this. Here I've shown you a nine by nine, image just as a sample. And we will actually work around with this. And note that I've filled it in with a lot of zeroes and 90s and just to keep the, approach simple. And these are intentionally designed to be values that we can actually do some simple math with. What we want to do now is smooth some aspects of this, to be able to generate a newer, smoother image. Here is the new image that I want to start filing in the values for. Notice again that in this image, I've actually given values of 90s. So you can imagine them to be the most, the peak value, of the image. And again I've left us some holes with the zero here, black point, and also 90 here to kind of give it some diversity, and see how that actually generates itself. To help us do this, let's actually look at a three by three neighborhood. So while this image is nine-by-nine, I want to actually use a three-by-three neighborhood to be able to then, smooth out and an intense, the intention is, that every time I apply this three-by-three neighborhood, I want to generate a new value, at this point, and place it here. Okay? That's the goal. So by looking at this, you can start guessing that if I was to do a simple linear average, that is take the sum of all of this and divide by nine, I would be able to come up with a value that I could place in here. And that value would be zero. Let's keep moving, and now I actually, next time what I'll do is, I want to move this one frame here because we want to actually raster scan and move this all over this image to be able to generate newer versions of the output image. Now doing the summation over all of this neighborhood here, you would see the summation is simply ninety, and I have nine divided by nine, nine elements so you can predict what the next value is going to be, ten. And we can keep doing this one after the other, moving to the next one, one eighty is the sum divided that by. nine, twenty and you can see , how we can start filling up all of the values of this output matrix. Once I'm done with this part here, I've filled in the ten and now I need to rotate around, and start looking at these values here. Move here. Zero again. Three values of ninety, so 270 divided by nine. And using this, I can get all the way to the end here and, fill in all of the values that came out of this process. Now one thing you may notice that because the way we looked at our neighborhood of three by three, and we're replacing the value here which is value here, this whole top row, and the two edged columns, and the bottom row are, , not filled. We'll discuss how to fill that up in a, in a bit. So while this thing is filled up, let's now start looking at what really happened and what we can learn from this. So a couple of interesting things happen. There was a zero here. If you noticed, the zero is now in this image, replaced by a much higher value. Because again, if you notice it in both direction. This pixel would have been of this intensity has been smoothed out. Similarly there was a 90 here and it's been reduced to 10. And again you may argue that actually the whole image now is much smoother than this thinks. Now there were two reasons we could have done this one. Maybe this was some sort of an error. Or maybe we just want to blur some information out. And again both we will look at in careful detail. Blurring or removing noise and error. To help visualize this let me actually show this with a little bit of information that's not just numbers, but shades of gray. To achieve this, what I'm doing now is creating the same image, except now I'm giving white values to all 90. Assume this to be 255 equivalent scale between zero and 90. And all of the blacks are, , still zero. So this is my original image. And now what I do is run the same process and see what the output looks like. So this would be what the output would look like for that image. Again, smoothed out by an average filter that's a three-by-three rubbed over the whole image. And if you notice again, most of the 90s persist here because that's where majority of the information was much more in the neighborhood the same. And the rest of it now is kind of a simple smooth ramp, as opposed in this one where it goes to 90 to zero and 90 to zero here. All that kind of jaggedness has been removed.

06 - Smoothing along Edges.srt

Now I did say that we need to look at what happens at the edges of the image. So let's look at that specific example. Again, the same image except that this time around I've replaced the zeros with some numeric values. Again just to help us see what would happen. Again, my output. And I've filled it out again, the middle part as we've done in the previous frame. So now we have to start thinking about how do we fill the edge cases out? Again, doing my filtering using a three by three, which we will rub over. And we did that. Now, I still don't have any values here. To achieve that what I need to do is pad this image with additional information. So to achieve that what I would do in this instance, as long as I am actually looking at a three by three neighborhood, I need to add one neighborhood pixel, or one column or one row, and make this image bigger by one. , if my kernel or my size of the filter that I'm looking at is bigger that 3, let's say five by five, then I will have to add three different rows and columns to be able to give it more information so I can fill this out. So once I put this up, I need to also fill it out with values. One thing we can do is take these values and copy them over. So, , just copying them over actually gives us more information here. This one can also be then copied over, or some sort of simple numerical method, like an average of this could be used to do replace this value here. So now I've actually created a new image that's a little bigger than my original image. Now I just mirroring and copying this over. And once I do this, I can now run the process which allows me to fill in values here. So now suggests is that I can take this three by three and when I apply it here, the value would be filling out here. And similarly, this way all of this, this, and this would also be filled out. , there will be degradation of information. Most of the pixel values here would not, , be completely correct. Because we're kind of synthetic, synthesizing a buffer to be able to do this type of filtering. There are many strategies usually available how we can actually add this information. So some of the options of doing this is we can wrap the information around, as we kind of did, or we can just copy it from one to the other. Or we can reflect information across making this be the axes and take these values and move them around. Again many different methods can be used. What we're really trying to do is figure out how to increase the edge size so we can do computation of this and create a newer image from it. there is significant error accumulated in doing this because the edges will start loosing information as you can start making bigger and bigger filter kernels. remember again if I were to do five by five what will have to add two layer, two rows and two columns, and make a bigger image to be able to then fill in values. We will see examples of that as we do start doing image processing. That some of the information at the edges does start getting to be suspicious and lost.

07 - Observations.srt

So here I'd like to start making some observations about what we kind of just did. We did it in very kind of a specific step by step way. But we want to come up with some general methods to help us understand how to do this. Well simply put, what we did was we took a small image. So, a small three by three, and we rubbed it over a bigger image. And when we rubbed it, we did some calculations at each and every center point relating to this point and that point. And we put that in a newer data structure. The new value, which took the information from this three by three, plus this three by three and replacing this value. So let's see what we can come up with as a general approach to looking at that. Just to help us with terminology, and I used that term again in a previous slides too, is imagine this to be a function h ij, just a small matrix. And we'll refer to this as a kernel image. This kernel image in this instance is a three by three. And the area around each original pixel is with the one that we actually, the neighborhood is around one pixel. In this one, my neighborhood is of the size one. So my k is equal to 1. Again, k equal to 1 allows me to create a three by three neighborhood. I can imagine if the k was equal to 2, we would be able to generate a five by five neighborhood, and so on. again, the size of the neighborhood is important because, again, you want to use that to generalize our observation and how we go about creating these types of filters. And now looking at that equation what you can say is well, the window size will therefor be 2k plus 1. k was 1, 2k plus 1 equals 3. If k is equal to 2, 2k plus 1, is equal to 5. And so on. So this is an important, parameter that we need to remember is, we need to always remember the neighborhood size, and this will allow us to start figuring out the size of a kernel. In this case, our kernel is 3 by 3. Again, we will see that many of times these kernels will be rather big. Or again, depending on how big a neighborhood we want to smooth over we will employ different types of sizes of these images. Important to note that in doing this kind stuff we are again taking a three by three kernel, applying it to a three by three window here, and then move one after the other. And that's what I mean by rubbing an image over, rubbing a kernel over a bigger image.

08 - Observations Continued.srt

To help us generalize this now, let's refer to an input image as F index over i and j, output as G and h[i,j] as the kernel. And these are the terms you'll be using again and again throughout some of the stuff we deal with image processings. Again let's take our nine by time, nine by nine sample image. This is our three by three kernel with K is equal to 1. Just to help us do some simple math, I'm now going to take an element here with a neighborhood, with you know i and j at 3. So now looking for value of g at the output at 3. We want to figure out what was the math for coming up with this. And I've given it some variables, A, B, C, D, E, F, G, H, I. Just to kind of mirror the lowercase ones here. For generality, let's just now play around with just these two for now, this is the one that is moving around inside this. So G 3,3. The value here and the output would therefore be the summation of A. Lower case a with capital A, lower case b, capital B, and doing that for each and every element, the nine elements here. So this would be my equation. Now, since there are nine of these, I do want to always normalize it. So, actually, I would come up with a scaling factor, 1 over 9, to help me kind of normalize the values from all of the, them. For the case of just doing a filter that's an average. So right now, to do an average, my kernel would just be 1 over 9, 1 over 9, 1 over 9. Same values in all nine elements. So now if I do 1 over 9, and this is my image here. the averaging would be best defined by this equation. So, G 3,3 for a kernel which is an average kernel. 1 over 9, all, all the nine elements would be 1 over 9 and the summation of all of them. Very similar to the, what we had looked in the 1D case now applied to a 2D case here.

09 - Mathematical Representation.srt

Let's look at this example of the 1 over 9 summation to be able to see now how we can generalize this mathematical formulation. So the general form of this equation here would be the following. So rather than just for the 3 and 3, for any part of the image, I want to have 1 over something which would be normalizing over this equation. So for example, in the instance where k is equal to 1, this would be 3, 1 over 3. That generalizes to that. And , what we're doing is sum, summing over the whole two dimensional region here. Now, , if you think about it, , this is my index. And if I'm moving in this direction here and also up above. This direction is minus u. This direction is positive u. This direction is positive v. This direction is negative v. So allowing this by just stepping through one by one, we can actually start with this, so the first term would be i plus u would be this element moved to this moved to this, depending on how we loop over a two dimensional array like this, we'd be able to actually now accomplish something very similar to this. So, in essence, this is allowing to us to loop over all pixels in the neighborhood around image pixel F i,j. This is an attribute uniform attribute uniform weight on each pixel. , this allows us to do normalization like the way I did for here for 1 over 9. The same equation, remember from last time, we want actually to have the more general form, where we no longer doing the averaging, but a general form of the filter would be actually this equation, right? Where we have taken the lower case a and the upper case A for the, for the, again, the third element. Generalizing this, we get this formulation, again going from minus k to plus k in both u and v directions. We have actually now done is, we've moved this part, which was the attribute weights inside, so it actually also now depends on how things could change as we move around. And the same equation or the same terminology is right there. So this is now a most general form of what we want to actually do in trying to do simple filtering. So in this instance, this is the attribute, but it's the same as non-uniform weights, because it does depend on where I am even inside this. Remember, in the case where we did this, all of the values were the same. In this case, they may not be. And , the same thing is still true for how we loop over both the matrices. This whole process is referred to as cross correlation, sometimes also referred to or written as X-correlation. And it's something we will actually cover a little bit more in detail in the next lecture.

10 - Box or Average Filter.srt

So now that we have learned how to do this, let's start applying it to doing filtering on averaging for images. I'm going to show this simple example using the classic image of peppers showing a black and white image, 5 by 12, 512 by 512 image resolution. I'm going to apply a average filter. Now this time around I'm not going to apply a 3 by 3 but a 21 by 21 neighborhood kernel, okay? So that's important to note. That we've been playing around with just 3 by 3's but these can be quite big. And that's one of the things I want to show you here. This is a 5 by 12, 512 by 512 image and I'm going to apply a kernel that's 21 by 21. This is the output of that process. , if you notice, the whole image is quite blurry. What has happened really is, at each and every point we've replaced the value by the average of the neighborhood in a 12 by 12, 21 by 21 neighborhood. Now all of a sudden, if you notice, all the sharp edges are gone. Remember when we looked at images in the histograms and stuff like that? There are lots of peaks and valleys in these types of images. Well by blurring it, using a filter, we've gotten rid of a whole lot of really sharp peaks next to each other. Therefore, this image now looks a little blurred. Also notice the edges. This is what I was referring to earlier because what we're doing is we're adding information. Some of the information will be lost, and in this case to do a 21 by 21, but 10 pixels were added on all three sides, and the average of this does degrade information at the edges. Just to showcase this, the box filter if you were to look at it in the same height map configuration we've looked at, would just be a flat box like this and it'd have a shape and again in the image form of the same values across the whole sub-image that we have. Again this is a 21 by 21. The intensity is the same for each and every one of them. In this case I would assume would be, you know, 1 over 1 over 21, and this would be my kind of the kernel, what it would look like.

11 - Median Filter.srt

I want to now actually talk about a special case which is Median Filtering. Now averaging was, great to actually look at the neighborhood, and figure out, a numerical value that's the average of the neighborhood. But we can actually apply other statistical functions in here. we're going to play with Median. Again, let's make, take my simple example. My input image. Look at my neighborhood of three by three. And this time around, let's calculate its median, of this right three by three. what we can do is open up the whole matrix like so, and show, the whole thing. And , the median of this signal or this neighborhood of nine by nine, oh, three by three, nine elements, is 20. Just to, compare, the average of this signal is, would have been equal to 19. So , just by doing median to, averaging, we have changed the intensity value of that output that I want to put in the output image by one. What we want to do again, just like anything else, is we want to run this application and run it each and every , pixel, rubbing it over the whole image as we've done before to generate, , a median image. So notice this is a little different. We're not doing cross correlation as we did before. For, we're not actually putting the elements in, we're actually taking the element, nine by nine, and running a separate process, a separate function on it to come up with a new value. So we are no longer kind of doing the kernel, because in this case, in median filtering, there is no kernel. Kernel is actually a function, rather than an image. Let's look at an example of how media filtering works. So just to make sure we are clear median filtering is a nonlinear operation and it's often used as a very strong tool in image processing. And I'll show you examples of how actually, strong of a process it is. Mostly what it is used, is to reduce noise, but other good things with it as actually also preserves edges and sharp lines, which is a valuable thing. And as I said previously a median of all pixels. Rather than using a kernel function as a stronger tool, than just computing the average mean of that neighborhood. And I showed you an example of just doing it over nine elements or a three by three size window. It could be again for larger things

12 - Median Filtering for Smoothing Images.srt

So now let's look an example. Same peppers image. And look at the output of the median filter. Here I have applied an 11 by 11 median filter. If you notice, yes it is a little bit blurred out, but the edges look quite good. Again, because it didn't just take the average and replace it by a new pixel value. It took the one in the neighborhood that was the best, the median score of that neighborhood and replaced that central value with that one. Just to compare, this is the average, and this is the median. Average, median. Average, blurry edges. Median, sharp edges.

13 - Median Filtering for Noise Removal.srt

One of the classic examples of median filtering is to applying it to remove noise. So here I've shown an example of an image that has what is referred to as salt and pepper noise. what happens in this kind of this image, and it could be because of a bad sensor, is some of the pixels or some bad sensor or some of the processing errors, some of the pixels have either black value or white value. So if you notice, that's why it's called salt and pepper, many blacks and many whites. Again, they're scattered all over the image in a random matter. We don't know where they are. But if you apply a median filter to it, you notice that all of the salt and pepper noise is gone. You might be curious as to what happened if we do this as a simple averaging. This is the result of doing an averaging by 3 by 3 colonel, and actually if you notice, to perhaps only magnified the salt and pepper noise. Now I'm just going to flip them next to each other so you can see and compare. This is the average. That's the median. Average. Median.

14 - Summary.srt

So, to quickly summarize this lecture, I used image smoothing as an example to teach you about how we can actually do neighborhood-based. Ways of actually taking information from the pixel neighborhoods to be able to impart smoothing functions or filtering functions at a new point in the pixel to create new images. Use that as a format to kind of introduce a concept of kernel and smoothing over an image using a kernel, and, , show how we can do averaging and median filtering. The intention, , was to, to show you how we can actually do neighborhood types of processing, beyond simple pixel processing, to be able to achieve simple filtering approaches on images. Now, what we will do is I did introduce the concept of cross-correlation. We will now start looking at cross-correlation and another method called convolution. To help us understand more how we can do simple image processing. Again, the goal has been for us to look at intuitively, how this all works. Because as we go towards and try to start doing higher levels of efforts and computational photography. These will be the foundational blocks we will be picking up on to learn about how we can do image processing and computer vision applications. Just again want to thank people that I've borrowed information from.

# 02-05 Convolution and cross-correlation.txt

01 - Image Processing.srt

So far we have looked at simple mathematical operations of trying to do point processes across images or looking at neighborhoods of images or neighborhoods of pixels to be able to do simple mathematics on it. Now let's try to create a formal mathematical representation of it. For that purpose, I'm going to introduce the concept of cross-correlation leading up to the concept of convolution. These will provide us with simple mathematical formulations and mathematical tools that will allow us to do things like smoothing images, kind of doing blurring on images, and also reducing noise and finding different types of information from images.

02 - Lesson Objectives.srt

In this lesson, the objectives are for us to now get a little bit more deeper into understanding of how we can do simple image processing and filtering. One of the things I'll be introducing you to today would be the concept of cross-correlation that I've actually already introduced in previous lecture. But now I'm going to try to define it much more accurately and mathematically. Using that concept, we will look at the whole concept of convolution and the relationship between convolution, and cross-correlation will also be introduced in this lecture. I will also describe some of the properties of how these types of methods can be used to do filtering types of processes that we won't actually do to be able to do computer vision and image processing on images.

03 - Mathematical Representation for Smoothing.srt

Now in the previous lecture, I talked about how we can actually mathematically represent the whole concept of smoothing applied to images. We looked at a specific instance of being able to smooth an image with a kernel. You may recall that in that lecture, we looked at how we can actually mathematically represent this process of taking a kernel and applying it to a small region, a three by three one, in this sample image. While this was for just doing the averaging, we can also do this in general terms for any kind of an image. And we came up with a term, or a mathematical formulation, which now lets you loop over the whole image, but also provide non-uniform weights to each and every one of them to be able to create a general equation of how we can do any kind of filtering. In the last lecture, I referred to this term, or this process, as cross-correlation, which was an attempt to loop over an image, or as part of an image in this instance, and expand this over the whole image with a kernel, which was not trying to change, and put non-uniform weights to be able to impact the output image. we would loop over the entire image this way. Now we want to actually start looking at this whole concept of cross-correlation in a little bit more in detail

04 - Cross-Correlation Method.srt

So what do I mean by the cross-correlation method? In signal processing, cross-correlation is a measure of similarity of two different waveforms as a function of the time-lag applied to one of them. What that means is that I have two different signals, two different waveforms, and I want to combine them to figure out what are the best ways that I can correlate the two different signals together, and allow me to kind of do things, of measuring the similarity between those signals. We will look at that for a variety of reasons when we get into feature detection and stuff in the later lectures, but this is an important part of what we want to actually look at a little bit more carefully now. And then actually we'll use that to develop other concepts. Another way of looking at the cross-correlation is also considering it as a sliding dot product, or an inner-product of two different signals. And you witness this when we actually look at a smaller kernel and we slid it over a bigger image and actually computed at the center, or a representative point, the output which was the combination, or an inert dot product of those two signals. And that's an important part on how the process unfolded as we looked at how we did the processing or filtering in the last lecture. Mathematical notation of cross-correlation is shown here. Again, we are using the two summations and looping over the whole image here with non-uniform attribute weights. Mathematically, we will denote this by symbol here, where the kernel h is being cross-correlated with the out, input signal, F, to generate an output G. So what do we mean when we say now, we are filtering an image? What we mean by filtering an image here is, what we are doing is replacing each pixel in the output with a linear combination of its neighbors with a kernel matrix. And for each one of them, there is a kernel or a mask signal, h here. And that is a prescription, a function of weights which is applied as a linear combination to generate an output G. And we saw this as I rubbed over, again h over the input image F to generate the output G, earlier.

05 - Example Box Filter.srt

Also what we did was we looked at different types of filtering mechanisms. So here I want to be able to take a box filter. In this one I'm showing a 21 by 21 box filter with uniform values and we can apply that to smoothing of an image. If you look at this part of the image here. And you sense, , you see a flat, gray kernel. And, if you look at it in a height map that we've looked at it it's got a fixed value just like this. And that is now our average or a box filter. Let's look at how we would apply a box filter to an original image like this, the pepper's image. Again, it's the flat kernel. When applied you get a blurry image. This is a 21 by 21 kernel. The linear combination of the weights here with the linear, with the constant values of the box here, result in a rather blurry image. Notice again, this is a flat kernel or a box filter.

06 - Example Gaussian Filter.srt

To help us understand filtering a little bit let's actually not just look at box filters. And here, I'm going to actually use a Gaussian filter. A Gaussian filter, based on the equation of a Gaussian, it can be used to generate a kernel. Now in this case, the kernel in this is, is still 21 by 21. But, the values are normal distribution. So you notice, more bright values or higher values here. And as I go away from the center, they get darker. In the height map we have looking at this, you will see this to be a little different. So now if we're for example notice, that there is a peak. And then there are values, in this one. Most intensity is here. Lower ones almost zero here. And now we want to actually look at this. And apply this, as a kernel to do, smoothing or filtering operations. So let's take our example again and now this time around I'll apply the Gaussian Filter. Which is again, shown by this 21 by 21. Look at the values that we just saw in the previous slide. And applied here, you see a result which is also smooth. But here you might be able to notice that some of the edges have a little bit more detail. Again primarily because it's in the linear combination. It's attribute weight is it's giving more weight to the pixel, at the center and actually is giving more value to that, and less value in the linear combination to the values that are away from the center. So as we rub it, it tries to give more values or more, more detail at the level, where the center is. Just for comparison's sake, I'd like to show you both these results next to each other. Hopefully your video resolution is good enough for you to see that this looks a little bit more blurrier than this one. Just for sake of completeness we're going to zoom into to see some more detail. Here is the zoom region that I've chosen. Where the stem of this pepper is the one we want to focus on. Zooming in, you will see that this one is much more blurred, and actually just flattened out. While in this case, you'll see a little bit more of detail. Now , it's not very like the median filter we had looked at before. But Gaussian filter is just a different kernel. And you can see other details also which are completely lost

07 - Using Gaussian Filters for Smoothing.srt

Here I'm showing a simple 256 by 256 sample image. And what we will do now is look at how we can apply various types of Gaussian, kernels to it. Let's start off with the most simplest one. Now here I'm using sigma of 1 pointed, kernel, round, and as you go away from the center, 0 0, it gets closer to, to being 0 values. The resulting image here is blurred, but not extremely blurred. It's got a little bit of defocusing going on. But as I increase the variance to sigma 3, you can start noticing more blurring. Similarly as I move to sigma 6, even more blurring is visible. And, finally moving all the way up here, we can sigma 9, and it's the most blurred image there is of the sequence here. And that's an important part for us to remember, that now by changing sigma, which is the variance, we've increased the extent of the smoothing from one point across the whole image. And again the kernel here was different sizes. And we can play around with, , the neighborhood size of the kernel too. So this now shows us we can use Gaussian Filters for smoothing. It's something we will actually play around with a lot in the rest of the lecture as we start getting into things like kernel filters and stuff like that

08 - Filtering by a Kernel.srt

So let's expand on this whole concept of filtering using a kernel. And this time around, what I'd like to take the concept of cross-correlation to define the concept of convolution. Let's start off with a, again, a simple image. Here, actually, there's, , an interesting variant on this image. This is sometimes referred to as an impulse image because in essence what I have is zeroes everywhere else and then a peak. A brightest value and then down to zero. there's only one simple impulse in this image right here. So zeroes everywhere else and a bright peak in the middle is what we're looking at with this image. We want to do a cross correlation. In this case, a filter with a kernel like this. And we will take a kernel just to have values. We've been playing around with these parametric values of a, b, c, d, e, f, g, h, i, to represent this kernel. We're now interested in asking the question, what happens if we slide this kernel over this image here, this example image, then what would be the output? So this would be the output that we would actually like to generate. , this is the output G[i,j]. What we would like to do is now take the cross correlation of this with kernel and start filling in the values here. Let's start off with a region here. Again we just do a cross-correlation of this with that. All of the values here are multiplied by 0. So their answer here would be 0. Let's now move this one like that. And the computation here, if you look at this region with this, f multiplied by 1 the answer here would be f. Move this by 1 again. Do the computation e multiplies by 1, the rest of them are 0. The response output here would be e. Next one, d multiplies by 1. Response here would be d. Moving one more, we get to 0. And , now, we should fill in the other two rows also. So this is what my response would look like if I took an impulse function with a cross-correlation kernel. A kernel like this, do a cross-correlation and that's what, our response would be this. To reiterate, taking this kernel on an impulse image like this, doing cross-correlation. This is the response we get. Let's look at this three by three region. If you notice, an interesting thing happened, and we constructed this and you noticed what was happening. D, e, and f got switched, and similarly b, e, h, h, e, b have been switched. So in essence what has happened is this there's been a flip in this axis. Right, and there has been a flip in this axis, both axes there's been a flip. So this is what my output region this three by three looks like, and again as I noted they've been rotated. One axis, another axis, and I can flip it. And this is my kernel. In essence, what happened by doing this process was the internal parts all got flipped around. So, the bottom line of this exercise we went through was to showcase that if you have an impulse function like this and you have a kernel that has different properties like this, these values here. We do cross correlation, the result is a reverse response. Let's look at that with a real example. Here what I've done is created an original impulse function. Black values here, white peak in the middle. Let's actually create a simpler version of this too. Showcased here, black value, white here, and ramps going up this way and also coming up this way. So different gray values here. Zero, one and different values here. And let's see what happens when we run this process. So the actual output is, again, if you noticed just like this, but flipped in both axes, both in x and y or horizontal and vertical. So now let's look at exactly what happened. This is referred to as a convolution method, and a convolution is a mathematical operation where we take two different function, F was our input, h was our kernel. And it actually produced a third function that actually modifies the answer. And it actually gives the area of overlap between the two first functions. And it actually does that by showing an amount that one of the original functions is translated by. Lots of words, showing what we actually looked at in mathematical version, or at least a practical version, let's look at it with math.

09 - Convolution Method.srt

So let's look at the convolution method. This is the mathematical formulation of the convolution method. Some of you may remember the cost correlation method, you might actually find some similarities. And we'll compare them side-by-side in a bit for giving you a sense of how what the differences this would be between the two. One thing to note in here, is we would denote this by a symbol here. So now, rather than using the symbol which was this for cross correlation, we use a star for doing convolutions. Again, the h kernel and F, and it allows you to generate an input, or output G. One thing we notice is that in how we use cross-correlation, the filter is flipped in both dimensions. So first it's flipped bottom to top. That is, the top row is now at the bottom here. And the other flip is when it's right to left. And then we can apply cross correlation. So in essence we would have been able to do cross correlation, we're using this formulation if you've taken the kernel, and process it to has two flips. One a flip in the horizontal axis, so the top row would be switched with the bottom. And then the second flip where we would actually do a flip from the right to left, where the right row and the left rows would be swapped. So that would be a complete flip to generate a new kernel, and if you applied this kernel in this formulation, this would become a colon to a cross correlation method.

10 - Convolution vs Cross-Correlation.srt

Let's compare the two methods side by side. Recall, this was my cross-correlation formulation. This is my convolution formulation. You will notice the big difference here. This one, the index i minus u, j minus v. While here we're adding it. Plus u and j plus v. Again, the same looping criteria that we looked at prior, which has been being able to loop over the whole image in both the two dimensions. So notice between the two formulations of cross-correlation and convolution, there is one difference here. There is a plus index here and a minus index here. What does that mean? This is, was our kernel. We've been playing around with this kernel all the time. Let's actually play around with it a little bit more to understand how and what are the differences between cross-correlation and convolution. To help me let's actually I'm going to just give you an example of just a small three by three. Again notice if it's a three by three, the neighborhood is k is equal to 1. So now let's look at this formulation here. k is equal to 1. Remember, this one is an element that has indices 0 and 0 in two dimensions. k is equal to 1. The first element we will be doing is i minus, 1, so this is the index minus 1 by minus 1. And similarly as I'm looping through it, you would notice that this would be 0 minus 1, then ,. This would be 1 and minus 1. Traversing through this way, this you pretty much know the answer for this one is 1 and 1, and this would be minus 1 and 1. So notice this is how we loop around, so in essence when I did cross-correlation, I started gotten a, first b, first c, first d, first e, first f, first and then g, h, and i. And that's how I actually did the looping and how I got the result. And now let's look at this example, now my k is minus 1, but minus, minus becomes 1, so the first element I pick up in convolution would be this 1, okay. And the second one I would pick up would be this one, 3, 4, 5, 6, 7, 8, 9. Here in this case for this instance, I started with first. Then traversing it this way, and traversing it this way. In case of convolution, traversing it. In essence, by just changing the sign here, I've actually now given myself the ability to the flip, before I do anything. And that's important part on what we want to actually do. And that was a difference by just changing this no, sign here allows us to do. So in essence that's how we're able to get this completely flipped kernel.

11 - Predict Box Filter Output.srt

Just to help us practice with this, I'm going to give you a couple of interesting quiz questions. Here is my input image. I have a box kernel like this. And the goal is for you to figure out what would be the output of cross-correlation or convolution. Check the boxes of the right answers, which could be either this or this for cross-correlation, or convolution would be one this one.

12 - Predict Box Filter Output.srt

While I'm sure you by now have figured out that in this case, our kernel, is , a symmetric kernel. So the answer would be this for cross-correlation. And even if I flipped it and I read a convolution through it, will still be the same answer. There is not going to be any kind of impact on this thing, because this kernel would apply both of the cross-correlation and convolution the same way.

13 - Predict Gaussian Filter Output.srt

Same question, for now using a Gaussian kernel. So this time around I'm actually using a Gaussian, and the question for you to now, try to answer is, what would be the output, if I applied cross-correlation with a Gaussian kernel, and convolution. Please choose the right box.

14 - Predict the output 2.srt

Now in this instance also if you notice this kernel. If I did any kind of horizontal or vertical flips to it, it would still appear the same. So , because of that, both cross-correlation and convolution would have the same output. And these would be the wrong ones.

15 - Convolution vs X-Correlation 1.srt

Just to reiterate the last quiz, and now actually looking at more general terms, let's look at these two questions. In this question, I want you to think about any image which is convolved with a box filter will have an averaged output, or averaged output with flipped images, both vertically and horizontally. The same case of a Gaussian filter. What would be the output? Would it be flipped or will it be just blurred?

16 - Convolution vs X-Correlation 1.srt

Again as we witnessed in the last set of things, the answer for this would be we just get the average output, and case of the Gaussian, we get a blurred output. Neither of them be flipped.

17 - Convolution vs X-Correlation 2.srt

Another question. And this one, I want you to think about what an impulse image is. Remember, again, we looked at an impulse image. And it was zero values everywhere and a peak value in the middle. So one here, zeros everywhere else. So in this instance the question is, an impulse image, convolved with a box filter will have. Output just with averages or will it be flipped? So, another question, an impulse image cross correlated with a Gaussian Filter will have what kind of output?

18 - Convolution vs X-Correlation 2.srt

Now I know by now you most quite of thinking this is all obvious and simple. And that's exactly the point. That convolution on an averaged output with a box filter because it's a flat average filter, would still result in just an average output. And, again, for the case of the impulse with cross-correlated with the Gaussian filter will also just result in a blurred output.

19 - Convolution vs X-Correlation 3.srt

The result of convolution and correlation is the same when these choose the right answer by checking one of the boxes here. When the kernel uses symmetric in both X and Y or U and V. Or the kernel uses not a box response. Kernel used has negative values. Image being filtered is gray scale. Or image has symmetric histogram.

20 - Convolution vs X-Correlation 3.srt

The answer to this one, I've gone, we're just going through the last few quizzes, should be obvious. That whenever the kernel is symmetric, which is the case with the box filter or Gaussian filter and all those other types of filters we've looked at so far. The as, the result of convolution and correlation should be the same.

21 - Properties of Convolution.srt

Let's look at a few properties of the convolution method. One property is that a convolution output process of convoluting an image means that it's linear and shift invariant. What that primarily means is that it's the same everywhere. The value of the output depends on the pattern of the image neighborhood, not the position of the neighborhood. So it doesn't matter how we apply, it'll always will come up with the same answer. And that's an important part of it. Doesn't matter how much you shift the image. Another property that's extremely valuable is the commutative nature of convolution. That means that I can apply F and G in this order, or I can flip it and apply G and F in this order. It's also associative, which means that first, I can do a convolution of F and G and then convolve with H. Or, I can change the order and do a convolution of G and H before. And, and afterwards I can do a convolution with an F. Another property, and I think we've experimented with this already, if we take an identity, a unit response in this case shown by the simple thing. And if I actually do a convolution of a image with, or a function with an identity, we get the original one back. Question for you folks to think about on the forums is, is that true for cross-correlation? I look forward to seeing you discuss that there. Just to kind of prove this point here, I take this kernel, or an image here, sorry. And this is my impulse. And now notice it's not symmetric. And if I just do a, a convolution with this, we will get the original one back. One more important feature to remember is that convolution is a separable process. That means is that we can actually do the convolution just by actually having kernel that's captures the rows, and we can actually do it separately for the columns. We can actually now allow you to separate out even kernels to have two different kernels. That is one just has the rows and the columns themselves, and we can actually, you know, run the process one by one. Best part of this is actually man, many computational advantages including, , keeping a lot of less things in memory.

22 - Linear Filters.srt

Let me show you a few examples and how we can apply these to create linear filters. This is my original image, I'm just showing a small 64 by 64 and I hope you can see this in detail. Let's take this original image and we're going to apply a bunch of filters to it. Let's apply a simple kernel here, just a simple, again, an identity. What do you expect the output to be? Next one, we can apply where there is not just the identity, but now we have shifted this by one. What should the output be? Here , we've seen this kind of stuff before, just the average. We can not guesstimate as to what the output would be, what the output should be for that one. For this one, as you've looked at before, it should be the same image again. We haven't actually done anything, right? But just doing a convolution. We're doing this, , we'll see that the image has shifted by one. what we've done is kind of given a shift to the image. This one we've seen many a times, and again applying a convolution to this one will give us a box blur, or an average blur. Let's look at another example. And this will give us example of how we can combine filters. Again, my original image, and this time around I have two different kernels. I want to take this kernel multiplied by 2, which is 1 now. , it's saying is that I'm going to now try to give it a little bit more strength at the value itself. But then after that, I want to subtract it by an average. So, I can do this process. Remember, the properties of convolution allows me to do something before I apply it to the image. I can actually do this process individually. And this would be my new kernel, right? I mean, twice. 1 subtracted by 1 over 9, 1.9, and this becomes, and this is what my height field of this kernel looks like. Applying this, any guesses what would the output be? In essence, by doing this, we have created a simple sharpening filter. What we've done is allowed it to actually create much more resolution information at the peak. And you move it further as you go down. So in essence this raise the information then we subtract it by the average, and the output is a simple sharpening filter. So, by doing these kinds of combinations, we can actually generate a lot of different types of things. And we will, trust me, get into a whole lot of these types of filters. And for those of you who are now started playing around with your different types of development platforms, please remember you can actually do this in Mac Lab, OpenCV Python, or filtering, or any times of tools that you've been playing around with this. And we will be doing assignments on this too

23 - Summary.srt

So in summary, for this lesson, I've covered basic concepts of cross-correlation, convolution. I've discussed the differences between cross-correlation and convolution, as applied to image filtering, and , I've also discussed the properties of convolution, and how we actually now can generalize the whole process of how doing, how we can do filtering with images. This has just been a foundational types of things. And now we're actually with next lecture, we're going to start finding more information from images, specifically things like edges and images, which then will actually create the foundations for doing something we want to do. That is feature matching. All right? So we will be actually, taking what we've learned about filtering, and now start getting content information out of images, rather than just doing blurring and filtering that we've been doing. And I noted I'd been using MatLab for some, some of this stuff, but you should feel free to use whatever you want. And I just always like to thank other people that I've borrowed some concepts from.

# 02-06 Gradients.txt

01 - Intro.srt

So, in the whole computational photography pipeline, what we're interested in is finding interesting features within an image. So far, we've looked at point processes and convolutions. Now, what we want to actually start doing is building on these concepts, find interesting features in an image that would be match able across different images. For that purpose, building on concepts of point processes and neighborhood computation on an image, I'm going to introduce the whole concept of image gradients, which says that at any point in an image, where is the likelier direction of change? And using that, we will be extracting things like edges and other features that we'll talk about next. But, in this one, I'm going to actually use the whole concepts of convolution and cross-correlation to introduce you the concept of how we can do processing on an image, to compute image gradients.

02 - Lesson Objectives.srt

The specific objectives of this lesson are for you to learn about how we can detect features in an image. We will specifically be looking at edges in a image and why are edges important and how they represent the information that's available in an image. We would look at those within the context of how we can compute image gradients and how we can actually use the computation of image gradients to support computation of edges. I will also give you a little bit of an, a background on how we can actually compute those image gradients, in a mathematical form for both doing continuous functions and also discreet forms of images

03 - Recall Convolution and Cross-Correlation.srt

So, I hope you remember from the last lecture the concepts of convolution and cross-correlation. We'd looked at in detail, this was how we denote the symbol for doing convolution between a kernel and an input image to get an output. Just related also the convolution method. The big difference between them is how we loop around the matrix or the input image to be able to then assign attributes from the filter. And , using this method, we can apply various types of filters to process images. And we actually looked at how we can do this for doing noise removal and blurring and smoothing of images. , you should by now remember that depending on h, if it's symmetric or not, you have to choose which one of these to apply.

04 - Using Filters to Find Features.srt

So now let's ask the question, how can we use the filtering mechanisms that we have looked at so far to find interesting features in an image? So here, for an example, are the two images. You may recall them from the instance of the panorama building experiment or exercise we did earlier. And we want to be able to take these two images. You know, one this image here and that image. And actually for this case, for just sake of, let's say building a panorama, we want to be able to find features that are similar in between those things. Then we can do things like alignment. So in this case what I'm interested in is finding information that's similar in both of them. Here I've created about one, two, three, four, five circles that kind of point to specific features or information in this image, and again the same five features are available in this image. Again, if you remember, this was a pan camera motion from this to that. So there were two different images . And you can notice that there are different images, except that they have a little bit of overlap. And what we want to do is find the same feature in both of them. Here I've just drawn the lines between them pointing out the correspondence that this same point here is also visible here, that one is visible here, and so on. So to achieve this process of being able to then align these two images, or find different things that are similar in them so then we can actually register them together, we need to extract some sort of higher level features. So one of the biggest advantages of mapping or extracting these features from an image is it allows us to map these raw pixels into an intermediate representation. Which means, is that now we can take an image, extract those features, and not actually be looking at the image any more, but just those features. Such an intermediate representation allows us to have, to have a lot of information but also kind of reduce the amount of data. Because what I would just do is, after a while, process this image, find all those 5 or maybe sometimes 20 different features. And 20 different features in here, and just attach that information and not anymore require me to carry around the image or the information in there. So this is a kind of a data preservation in of information, which we can use for a variety of things.

05 - Good Features to Match Between Images.srt

So now let's start thinking about what are Good Features that we can find between two images of the similar scene or a same object, that we can then use to be able to register that same feature for across two different images. I'm going to use this simple object of a water bottle to help us illustrate this. So as I've said before, we are interested in extracting features. , we want to find parts of an image that lets us encode that image in a compact form, so then I can do comparisons from one image to the other. I'm going to propose the concept of Edges here. Edges on an image, in a variety of ways, could be one way of encoding such an information. So what kinds of information do we want to look at? what we want to look for is various forms of discontinuities in an image or a scene, and that is one of the ways to think about what an edge is. So, an edge in an image is trying to capture the discontinuity in a scene, and use that to be able to then find information that would be again, somewhat repeatable across images. We'll look at these more in detail. So let's look at, for example, this water bottle. And see what kind of changes in the scene we could actually look for. And actually look for certain discontinuities in this. So, for example, at the water bottle at the top of it you if I was to start growing surface normals, you would see the surface normals going this way. And, after a while, there are no more surface normals because they are on the other side. So, this form of discontinuity, if there was a sudden change in it, would allow us to start thinking about how we could actually look for discontinuities of this form and the surface normals. So those would be surface normal discontinuities. Let's look at depth discontinuities. In this case, , we know there's an object , and it might be in front of something, so by just being able to look at the, you know, the sides here, you know that anything behind that would be at a different depth. This would be in front of everything behind that. So that would be a depth discontinuity. How can we actually look at the term water here, and be able to separate it out? Well, in this case, if you can notice is, water is written with a different color than the background. So surface color is also a cue, a discontinuity that we can look at, that points out how we can actually separate out terms like water, or even this band here. Or, in fact, even looking at how we can take difference between the bottle and the cap here. Another form of discontinuity is illumination. This object, , is being lit in different ways. And, it, creates a shadow. Or it could actually have other types of reflections and specularities and stuff like that, that are also coming in because of the lighting conditions on it. That is referring to discontinuities because of lighting changes and such on the object. So hopefully you see now, that by just looking at various types of discontinuities, we can start extracting some information about an image. So , what we now want to do is go back to the image itself and look for certain sets of discontinuities that would best capture the changes because of the normals the depth, surface color, and illumination. , that is a rather big problem and we should be looking at it, and this is one of the bigger things we look at in computer vision when we do analysis of scenes. Now one thing I wanted to emphasize here, this is an extremely important concept. The Edges are encoding change in a spatial way of looking at an image. Anywhere afterwards I look for a change, and that chain encodes a lot of information. And in fact some may argue that this is a very important set of information about an image. So in a information theoretic manner, what we an actually say that edges and are encoding change, and therefore are an efficient way to encode an image. So any time I can think about an image is, if I can start looking at where the changes are, that starts giving me a lot of information about what the image would be, because now I've actually taken a a differential code to represent an image. Wherever there are changes, I'm going to highlight those. We'll look at that a little bit more carefully in a bit.

06 - Edges in Real Images.srt

Just to help us understand this, I want to actually also emphasize that let's find these type of changes in an image, a real image like this one. , photo taken by my colleague Henry Christensen. And let's look at various types of things within this image. First, let's look at illumination changes. Now if you look at this scene, there is, , because lighting changes, there is shadows. Those are illumination changes. You can, , see a lot of different types of things. There's not a lot of specular reflection here. But that kind of tells us that there could be some changes happening in the scene because if the, if the lights were on, we would actually see a lot more illumination information. We can separate out things like the word tech because the color is different in front of that. Also it has a specific shape that actually helps us also distinguish and separate that out. Depth discontinuity again comes in, in this tower here is in front of something else. We can look at that depth and see that this has to be in front so there should be some sort of a discontinuity here at the tower. Surface normal, again if you look at the steps here you can see a lot of discontinuities because sometimes a surface normals of these steps are changing around. You can see this kinds of things across in the shape here. Each one of these shapes has a different surface normal, and it helps us kind of distinguish that. objects have well-defined boundaries. This you know fire flume kind of things here or the roof decks here at different objects, the windows and such, each one of them have different levels of detail. And again that could be look for separately. And also additionally reflectance change. Now this is most visible here because and I go down this roof top you can see the changes in colors. A much more subtle change, but there is definitely a reflectance change here. That starts telling me that this is a different shape and there is much changes in this one too. Hopefully this helps you try to start thinking about how we can actually look for these types of changes in real images.

07 - Recall Images as Functions.srt

Recalling in something we have talked about before. We've always thought about images as being functions. Now the one of the best ways to visualize that this kind of thing is looking into this height map that we've looked at before. Here you can see in the 3D height map edges appear as ridges. So if you look at it when it comes around, you can see the information that lets you differentiate between this part of the image and that. Something we will come back, hopefully connect up to what we have looked at in the past to what we are talking about now.

08 - Edge Detection.srt

So let's look at edge detection with a simple example. I'm going to give you my usual, simple image. Let's use this as an example, and see what we can do with it. So the basic idea of edge detection is, we want to look for a neighborhood with strong signs of change. So let's, for example, look at this pixel, or this neighborhood of one, and look at it with respect to the other pixels around it. How much is the change from here to there? How much is the change from here to here and also just in that neighborhood of these pixels here, figure out, oh is there a big discontinuity here? And , there is a significant discontinuity 12 intensity to 90. We can start looking at that one by one throughout the whole image, so the same when I look at this. And I compare it with this. There's a discontinuity, a significant one between these two. However, not much so between 90 and 89. Looking at this pixel here, I would also, also have the same type of comparison. Much more discontinuity here. , I didn't go down all the way here, but if you notice, 89, 86, 87, 82 were somewhat similar. Not exactly the same. I mean there is discontinuity, just not a significant one. But between 12 and 88 much more. Again we can keep coming down this way. And 9 and 10 have discontinuity, 15 and 12 have discontinuity, but small. So when I hit 12, you notice a significant discontinuity here. And also between 12 and 84. So this process as a scan through an image. I will be looking for each and every pixel and look in the neighborhood, the four connected next to it itself and say, okay, do I need to put something between those pixels to create an edge? And I can just go through the whole process and see how this looks for this instance, and you can see that, , just by looking at discontinuities between different types of things, we can create this simple, line, which separates out this part of the image with that one. Or points out that there is a discontinuity between it. Here is another way of looking at it. , now I've just given those gray values and now I can see the edge between the two. Becomes much more clearer this way also. So as we do this, we have to think about various things. What is the size of the neighborhood? I started off with the assumption that the neighborhood size. Which I've always referred to was one because I look at it with respect to that. Then also what metric represents a change? And in this case we looked at saying is, well let's not think they are different if the pixel intensity's different by three. But if it's several orders of in this case, , almost 78, we would put that as an edge. Or, in this case, you know, again, about 76, so all of those radiations are much larger. So we came up with some sort of a threshold, and defined that as where we want to actually look for those changes. If there was with certain amounts, if it was above it or not, we would actually look for that discontinuity. there is discontinuity, much smaller one between each and every pixel. But we will just be looking for larger ones. So that's one more thing we have to look at.

09 - Edge Detection.srt

So, using this idea, let's actually have your practice on the same concept briefly. So, what I want you is to go through this example image. And, , at each and every point, again our neighborhood is size 1, look at where the edge would be. And, I've given you the threshold. So, for example, in this case by looking at 12, you would say that this should be a edge here. But, no edge between 12 and 10. And, use that to be able to then construct a edge that goes through this whole example image.

10 - Edge Detection.srt

The answer for this , is looking at these edges, this way. This is the answer to this problem here. Just a simple exercise to demonstrate how you could actually look at the issues of both neighborhoods and thresholds.

11 - Finding Edges using Image Derivatives.srt

Let's dive in a little deeper into this whole concept of edges. But let's now look at images and derivatives of images. So I'd like to define an edge as a point in an image, pixel or group of pixel but there is a rapid change in the im, image intensity function. So again we want to look at, the image intensities, at a three pixel value, in a discrete form or in a function across the function itself. And look for where there is any kind of rapid change. Significant change. Again the rapid change will somewhere where the threshold value will come in. Small gradual change will be a few pixels, but also not just looking at how one pixel to the other but a significant size of the neighborhood. Just to keep things simple we're going to look at just the one slice through this image. This is a sample image that we will play with here. And if I was to just construct , the slice of this intensity, I would get this. So in essence, as I traverse through an image, the intensity is high. Where this grey values are. Goes down to a little whole number but it's darker and then comes back up. So this is , on this slice is the intensity map. Here's what we want to do to this now data here, this the intensity map. We want to actually just do a simple difference operation on it. That is I want to take, for example, this value and difference it from the next one coming after it. And then similarly I want to each and every value, I want to difference it from the previous one. This is a simple first derivative by just doing simple differences. And if you look at this, and actually if you look at this kind of stuff here too, there is no changes here, and all of a sudden, many comes here. There is a significant change. This would when you look at it in difference form, would give you this change . When it gets here, it goes back to zero again, no change. Another way to look at it, this would be for example the slope of this line, the slope of this line is zero. All of a sudden the slope goes from zero to a larger number and then goes back to zero again. So when you notice here, goes to larger number, goes back to zero. Similarly when we come back to here, it's this way and again the slope changes. End of this line again, comes back to zero. So if you notice this is an interesting phenomenon, that we will actually now capture. This point here is an output of information that came from this region here. And this region here was captured by this point. So if you notice now, is, these extrema points refer to this edge, and this extrema point refers to this edge. We'll only look for derivatives in x, and these are the two edges. But there's a rapid change. As I traverse this way on the image. , using this now edges, we would be able to find something that would always be similar. Even if I actually took the image of this one slightly moved around to a next part, or if this was translated in some way.

12 - Compute derivative to find extrema.srt

Just to help you practice for this, I'd like for you to try out a simple quiz in the same, same kind of a setup as the previous one. Remember what we were doing is, we were going down, from one column to the other and finding another value of indifference it, be able to then find what would be the first derivative by just simple differences. In this case, we are going to just look at absolute values, so we are going to drop the negatives, so for example here, between those two, the value should be 0. Similarly, between 0 and 9, the value should be 9. Just fill the rest of it out and then use this to kind of extract from it, what you think are the most appropriate extrema points, I'm just also drawn here the curve for the same thing here, right, 0, 0, goes to 9, comes to 9, and then again 8, and this is approximate, comes down to here. And this is just my 1-D profile, of what the signal looks like, and we're just looking at this in one 1-D, also, check marks here as to where the extremas would be, so that will the edges be for this image example.

13 - Compute derivative to find extrema.srt

Here is the answer to this question. By looking at the differences, you notice that, , there's a peak here, comes down and then goes back up a little bit. So the two extrema points here and if you just look at the numbers, by just doing differences it's one intensity, but when I'm coming here there is a bigger difference this way between eight one. So this allows us to kind of start capturing this kind of information.

14 - Differential Operators for Images.srt

So now, let's look at differential operators for images. Recall, that we'd looked at how to do various types of efficient methods to do processing on images, by taking a kernel and applying it to an image. Well, let's start to build on that concept, but now, actually, let's use it to be able to do things like differential calculations on images. So what we need, , is an operation that, when applied to an image, returns its derivatives. It would be a lot of fun if you can actually create these operators that are like masks or kernels because we've already learned how to do things like convolution or cross correlation. And that would allows to now compute image gradients by running operations like this. And , then we need to do is, threshold this gradient information to select images. We will remember this term again, how to thresholding to get to edges. And we'll cover more of this in the next lecture, when we talk specifically about edges

15 - Image Gradient.srt

So now let's dive into understanding what an image gradient is. To help us define an image gradient, let's take the example of this sample image again, where we're interested in is at each and every point we want to compute the gradient. And with respect to the neighborhood that we're in traversing and again the both dimensions of the image. A Gradient is defined as a measure of change in an Image function. So that at this point, what are the changes of the image function with respect to how things change in the x direction or in this case, the columns or the y direction, the rows of an image? We'd like to be able to compute this both in discontinuous and the continuous form. So how do we define it mathematically? So , the differential of the function F, which represents an image, with respect to x. And then differential of the function image F, with respect to y. And , both dimensions of this is represented by the del F, is the gradient of the image F. In essence, this is the way to measure the change in image function F, in both x and y. Let's look at that with a couple of simple examples. Let's take a simple ramp image here, starts with a dark value, let's say zero, to 255 where the pure white is. At this point, I want to be able to measure the change of this image function, as respect to how it is in this neighborhood here. Very obvious if you take the same function here, you notice that all of the change in this direction is for the x direction, none for the y. So that , this is a very simple case where you can just represent the del of F is nothing else at this point to be this function here. The counter, , is when there is a pixel and actually it's now, or this image here as a ramp here. And , going from two fifth, 0 to 255, from this point onwards, and, and in this case, we can simply look at that there is not going to be any change in the x direction. That seems to be fixed, and just in the y direction. Combining these two, we can look at if the changes are in both directions, then they will be, the gradient direction would be at an angle, which is there. In essence, what that means now is that the gradient point in the direction of the most rapid increase in intensity. >From here, the intensity is increasing rapidly in this direction, less so in this or this direction, so this is my most dominant angle. Let's look at this a little bit more carefully. How would we compute the gradient direction? Just take the inverse of the tangent, I mean, this is my vector, del F with x and y. If I know the y and the x changes, I can now do the inverse tangent to get data. The magnitude of this vector, which would be the modulus of this value here, which would be the square plus square, square root. So I have this thing, it'll give me the magnitude of the vector. So the angle of the vector and the magnitude of the vector are the ones we're looking for, and that would define the gradient at any point in an image. So while we're at it one thing I want everybody to think about is how does this, how does this relate to edge direction. When I have a point here and I have the gradient direction here the theta and the magnitude. Where is the edge? Hopefully you will have the answer in a bit.

16 - Discrete Image Gradient.srt

So I want to use this opportunity to, completely, define, an image gradient in a discrete form. For a two dimensional function, F of x and y, the partial derivative would be del of F x, y with respect to x, and this is in just x direction. And what we would be looking for is a small epsilon x over x, between the two different values with epsilon gong to infinity. For the discrete data, we can approximate this using finite differences. Which says for x, the del F x, y could be approximated with just moving 1 in, index point, in x, so , this would be as we have always looked at images to be discrete samples of moving from one row to the others. In this case, , one column to the other keeping y the same, and approximating with the difference like this. This is in essence what we just did in the previous cases of 1-D example, when we just took one row and subtracted from the other one, we would do the same, for this one, for rows, for y, and , columns for x. And this is an approximation that allows us to do simple image differences and use derivatives for images.

17 - Differentiating an Image in X and Y.srt

So let's see how we can do this for a simple image like this one. What I'm going to do is take this image and go column by column and create a difference new image, which is going to encode nothing else but the difference in the intensity values as we go down from one column to the other. This generates, , this new differential image. Let's look at it again. Notice the ridge showing up, a white value here, because we've gone from black to white here. And , a much darker black value here because we've gone from white to black. Nothing is visible here even though you can just perceptually see it. There is nothing there in the in this window here. , let's look for the same thing in the y direction. This is my equation for doing differentials in y. Same process now applied to getting the partials in y. Again we can see the y changes just like we did here for this image. Let's look at a much more natural image next. Here's a zebra image. Let's do the same exact procedure. Notice how the stripes have been separated out. Doing the same in y. Now, unlike in this case where we just have predominant emotions or actually edges or gradient changes just in x and y here, , here, you see a little bit of mixture. None of the lines are perfectly horizontal or vertical, but you can actually extract all of the differential information. This is quite valuable, and we'll be using this next.

18 - Gradient Magnitude.srt

So, as we're learning about gradients now, let's spend a minute to kind of review what we know about gradients. I've shown you four sample images here and the question is, which of these gradients, for this whole range from zero to 255, for four of these images, is the largest in terms of magnitude? >From this point to that. So, this is the region. And we're interested, , in which one has the lost, largest change as I move from this location to that. For examples, all of them are exactly, as I said, starting from zero to 255, but if you notice, each one of them in the direction that I'm pointing out, does have different changes. Answer by choosing which of the boxes applies to the correct answer. Thank you.

19 - Gradient Magnitude.srt

, this one is the most largest change goes from zero value pretty fast to 255. And in fact, it's 255 already here, while the other ones are not hitting 255 until much later.

20 - Gradient Direction.srt

Now, let's talk a little bit about grade interactions. Simple quiz again. Choose which one is the most dominant and the right, correct direction for the gradient for the four example images.

21 - Gradient Direction.srt

A quick visualization of this should show you that, , I'm going from gray values to, pretty fast to white values here. , that would be a positive gradient here. These two are actually in, in the wrong direction. This one is the name, same direction except that it's going from. It does have a gradient change in this direction, but it's a negative change. And if you were to look at this in absolute manner, this would have been also a fine direction to choose. But, in the real relative direction, this would be the one where we're going from dark to bright. So, the right answer, , is this one.

22 - Gradient Images Cont.srt

I want to actually get you to understand a little bit more about gradients, and I'm going to use this example. Remember, again, gradient magnitude is this and the angle comes from this equation here. This was simply the derivative, in x. This is the image for this one. This is the derivative in y. This is the magnitude, the whole value of this thing. , the brightness and stuff like that is kind of compromised because we scaling it up from between 0 and 255, but this now starts showing you exactly where the gradient magnitude is for this image. If you notice the zebra all of the changes anywhere are now visible here. This image is the gradient angle theta. So at any point it says, where is the direction of the gradient. Now this is extremely hard to visualize and understand very carefully, so what we'll actually do is come up with another way of looking at it. One way of looking at this is being able to now look at all the gradients, and drawing a arrow in the direction of where the gradient increases, and also giving that vector. Because if you remember, in essence, this is the magnitude of the vector and this is the direction where the great intensity is, and the direction of the gradient. I've shown this with the zebra image here. it's hard to see, so let's zoom in. Zooming in now starts showing you the gradient vectors and developmental detail. Even zooming in more you can now see what I was referring to. Now note again, this is something that was important, that the magnitude and the direction of the gradient is this way. However, the edge that we want is this way. Direction this way, edge this way. Similarly, there is another edge here. And the direction for this one is also this way. So that's an important thing for us to look at. And use that to be able to kind of model these things. , here we're looking for just the absolute directions and stuff like that. Even more close up you can see the edge and the red directions and the magnitudes of the gradient. Let me show you this thing in a different way.

23 - Visualizing Gradients.srt

Remember, when I initially started talking about images and continuous functions, one of the things I had talked about was how we can represent these as height maps. Turning this, this way, you notice what I was referring to. See the white stripes are all above, and the black spots are all below. In essence, you see nice ridges. And in essence, you can actually tell where the edges would be on that image. Just showing them as static images here. One, two, three. These are a little hard to see but you can see the ridges I was talking about. Right? Much more visible here. There is a kind of a black point here. So in essence, the rapid most change is right here. Right? The rapid most change is right at these points here, and this one here. The gradient directions are this way, the magnitude is this way, but the edge is going to be, here. Hopefully like this, we've simply visualizing this. this is something we've been looking at before. I'm just connecting the dots. Just to look at it again, you could look at the edges and the magnitude vectors here, . This is what we're really computing.

24 - Summary.srt

So, to summarize, I introduced you the concepts of how we can detect features for doing things like matching in images. We came with the simplest form of them by using gradients to compute edges. And how those edges could be used for doing other types of things will become apparent as we actually get in depths of how we can compute edges. So, in next class, what I'd like to do, or as next session, I'm going to get into depths of how we can compute edges from gradients. This was more about learning how we can compute gradients themselves.

# 02-07 Edges.txt

01 - Intro.srt

All right, now we know what image gradients are. In this lecture, what we're going to do is, we're going to build on the concept and start computing edges. Now, edges are one information theoretic way of looking at very minimal information within an image. What we will actually propose is, using this will help us get towards feature detection, which we will actually study later. But, in this lecture, what we're going to do is now learn more about how we can, in reliable ways, extract information from images, like edges, based on image-creating computations that we learned how to do in the previous lecture.

02 - Lesson Objectives.srt

So the specific objectives of this lesson are for you to learn how to compute edges from images. We will look at how we can actually do derivatives using kernels and neighborhood pixels, the concepts that we have learned again. And with that, I'm going to introduce to you three different methods of computing edges using kernels. I will also showcase that as soon as image noise shows up into these images, how it actually complicates the whole process of computing edges or compromises the whole process of computing gradients, which impacts how edges are computed. And then I'll actually introduce a special type of an edge detector.

03 - Recall Differentiating an Image.srt

Now recall from a previous lesson where we actually looked at the whole concept of how we can differentiate an image in x and y, and use that to be able to compute, for example, a derivative of an image in x, which is looking at how the columns, when I parse through this, can give me highlights, or where the maximal change is, and in this it detects, for example, the edges here in this rectangle, and , I could do the same, going row-wise in this instance, and we're going columns in this one, and we can use that now look for maximum change, this one then from black to white, and in this one, , white to black. This shows you very nicely where the edges, or at least with of gradient change is that we can actually use to compute things like edges. This was also showcasing a much nicer example, of looking at how we can do with the zebra, and here you can notice that you see, , information on how the gradients that is how rapid the change is as I traverse in this direction, and, also, if I traverse in this direction, so, in the columns and in the rows, x and y. Basic concept that we now want to build on to be able to do much more with.

04 - Derivative as a local product.srt

Just to help us understand how we can do this kind of stuff, I'm going to simplify this concept and now showcase a simple way of how we can do a derivative of an image. Again, you've seen this already just a of the image itself. Now we're going to look at it with x, which means, in a I'm going to look for a value, x plus 1, which means in this one. And x plus the next column, keep the y the same, and , I look for the original column, x itself. And I just subtract that and that'll give me the difference. We have done this in the previous style sub lecture also. Let's just simplify this and look what we can actually do to extract this concept of derivative as a local product. Those of you remember from previous lessons, local product, dot product, inner product was something which was relevant in the concept of cross correlation. Here, I've just simplified this by multiplying the first term with 1 and the second term with minus 1 to get rid of the plus sign here. Let's rearrange this whole term. So when rearranged, now we have moved the negative term in front. And the positive term later. we can rewrite this form as a dot product. Of minus and the 1 as a simple vector. And then, , the f x and the f x plus 1,y as the other one. So this dot product is now, in essence, how we could compute something like this. Now, , this is interesting because, in simple terms, sometimes I could replace this dot product in case of images with a cross correlation because in essence that's what this process would be. So this should start telling us something. This, for example, could be my kernel and this is my input. Right? So looking at that kind of wave we can now start thinking about how we can actually start doing similar types of processing. Now this is still a dot product. I'm simplifying this to kind of say this is cross correlation. We'll look at that in a bit.

05 - Derivative using cross correlation.srt

So let's look at the whole concept of using, again, computing derivatives using cross-correlation. Let's take a simple example here of an image, of a monarch butterfly. Here I'm interested, now looking for this, how can I actually extract from this, the two derivatives, both in the x, which is in the columns, and in y. What I can do is now I can come up with a kernel and cross-correlate with that to be able to extract my derivative, I could look for derivatives in columns that is x or, I need to look for derivatives in y. For doing this computation for the original, for del F x, y in x, how about I use the simple kernel like this, where I take the negative value and the positive value, this is what we did in the pa, last slide, and see if we do an inner dot product, with each and every aspect of the image, this is what we get as an output. This is exactly what we wanted, right, because we wanted to be able to kind of look at where there was traverse this way, maximal change, in an image, which was referred to, , as the derivative in x. Similarly for del y, the chain in y, we need to create a falter, that has a looks for the derivative change, in, this direction, which is minus 1 and 1 here, again, we showed example in the last slide of this, but in this case we would do this for y, this was for del x. Looking at this process, we will actually now notice the same monarch butterfly starts giving me information as I traverse down this way. So what we've done is taken these two kernels and applied them separately to this original image using the cross-correlation method, something you looked at before. Just in the last image you may not have been able to see all the details, clearly I'm just zooming up the images so you can see the monarch butterfly and the dx and del dy of this. So in this image, as I traverse, in the column, you can see again, gradient changes, happened here, and , kind of a bright line, dark line, which again shows that there was a beacon, troth, or changes in the image gradient itself, and x and y, we know there's the same phenomenon coming down this way, so here for example, you can see more details of this part, here, but, we don't see it here, but , coming down in this axis you see more details here, and you don't see that here. , as we've looked at with gradients we can, want to compute more information out of it. This output was generated using a kernel process where we came up with the kernel that I showed you and used that, to generate these images.

06 - Computing discrete gradients.srt

So how do we compute discrete gradients? Well, what we want is, we want to come up with a, an operator, a mask or a kernel. Remember this, what we looked at before. That effectively computes the discrete derivatives using cross-correlation. I've always been talking about this whole concept to find a difference, which is nothing else but a numerical solution where we'd solve a differential equation using an approximation of derivatives. And in this case we come up with differences from one column to the other, or one row to the other. So what we are interested in, and we did this previously, is come up with a simple kernel, that when applied to an original image, in a cross-correlation framework, could be used to, generate a derivative. How do we do this for discrete, images? Well we want to be able to compute discrete gradients. We want to be looking for gradients in x and y direction. Again, remember, this was my equation looking for a change in x and y. In this case y is going to remain the same because I'm going column by column. In this case, I will be going, row by row and keep the same column. So, kinds of kernels we can generate to help assist with this are, for example this one. Where now I have zeros here, minus one, one, zero and zero. I use this in a cross correlation framework. I'll be able to now generate a del x, our gradient change in x image. So those of you who are thinking how would I get the Hy, well I would actually just transpose this. Now again, so far we're talking about computing gradients using cross correlation. We're using Hx and Hy as the two kernels. Those of you who remember our conversation about cross correlation and convolution should be, be able to easily kind of predict how would we use the method of convolution here. And what we would have to do, the Hx and Hy, to be able to actually use the convolution process rather than the cross correlation process. because remember, this is not a, symmetric in both an x and y kernel. This symmetric in one axis but not both. So, ideally, a kernel should have some symmetry about an image point. And, overall, by combining these two, you might actually notice there is overall symmetry. So one question remains. Is where is the middle point of these kernels? Remember the processing we have actually done before where we have to look at and create a nine by nine kernel and then use that to kind of place the middle value. In this one, , there is no middle term here. So depending on how the operations are done, sometimes you would actually always have, have an image middle point here, or also kind always keep an offset. And I'll put an image point here. This does have an im, small impact. And we looked at it when we did simple differences from moving around the row. You will always be losing one column if I'm going around columns and if I'm coming down on rows you would lose one, row. So this is kind of way we will be looking at for information. But , you will also notice that sometimes we want to find kernels which allow you to have both symmetry and also, have a much well defined midpoint. For example, this is one kernel that is widely used. This should remind you of all of the other kernels we have looked, looked at for doing averaging. Here, what it's doing is taking the average of the information one column to the right and one column to the left. And , zeroing out everything else. So it's , an average of the left and the right derivatives. So is this a better kernel? Well, it does have a some features because it does have well defined midpoint. And also can actually, be used for doing various types of symmetry calculations. Transposing this, we can get the same kind of kernel in the y direction

07 - Various Kernels for Computing Gradients.srt

Let me show you three different types of kernels that are used in doing various types of H detection. The Prewitt, the Sobel, and the Roberts. The Prewitt kernel uses negative values in this and positive values, and a transpose. The Sobel gives it a little bit more heavier thing to the middle kernel. Our middle values. Remember we looked at trying to look at points where sometimes just giving it a ramp in the middle gives a little bit more information locally. Transpose of this is, , here. The Roberts one is even simpler, except it actually puts in values at the two different end diagonals. And uses that to compute radiance. Let's try this on our input image. I'm just showing you the response for the derivative in x. You can extrapolate and see what it would look like in y. You notice that, , it does actually a pretty good job in trying to compute the gradients. Or the gradient in x, in this instance. I'm aware that the, perhaps at resolution, you can't see the details. But hopefully, you'll trust me. And we'll try to make some of these available to you. , we're interested in computing edges. So from this set of input images in Hx, let's see what we can what edges would look like. I have yet to explain how we're going to go to edge from here. But I just wanted to show you examples right now. This is doing edge detection from the original image using the Prewitt, the Sabel operator, and the Roberts one.

08 - Impact of Noise on Gradients.srt

Before we go on, I actually do want to, throw a little bit of a, a wrench into this, whole analysis method. We're going to use a 1-D example to help us understand this. Imagine a 1-D signal, has a lot of value down here, minus 1, then a ramp up and positive 1, you can imagine this to be a simple signal, that could be an image, which has black, here, and white there, and , we is counting it this way. And , there's some sort of blurry, boundary between them, therefore the ramp. , now the question is, what happens when I actually do a column by column, that is just in this direction a differential of this. There's a lot of noise here, and you can see some of the noise in this, there. The derivative of this signal, just the 1-D signal, in its x will only be kind of capturing the details of the local differences, and in fact, when it gets to this point here, you can see a little bit of change, but not much, and in fact, by looking at this, you can imagine that, it would be hard to actually look for a sudden change in gradient, just because it's mostly capturing the noise. So the essence, in essence, it's, really gets harder to detect an edge, and there is significant noise in the signal, something we will have to start thinking about how we can avoid.

09 - Match Gradient Plots.srt

I just wanted to have you look at this concept a little bit more in detail, I've given you, , three different signals, A and B and C. And this is the original signal. And in essence, what this is a 600 by 600 image which has a white bar in the middle and black. So this is black, white, black, and I have only showed the signal as I this way. And , there is filtering going on, so there is a ramp from black to white, so you've noticed 0, goes up to 1, and then comes down to 0. The same image, I've then given 0.1 of the magnitude noise in there, you can see a lot of you know, noisy pixels coming in, and I've increased the noise. I just want you to look at these three and figure out, which is the right derivative for each one of them? Just put the no, the character of the choices A, B and C in the appropriate boxes.

10 - Match Gradient Plots.srt

The, the answer for this one is simple. If you look at it here, there is a gradient change. So, if I'm going up there, all of a sudden there was a big change and then back to zero. This one, again, shows a negative. So, this one is A. B has a lot of noise. But, it's not enough, and you can actually kind of see a replica of this thing, except with a lot of noise. There is signal, there's a little bit of hump, end coming up, and there's a little bit of hump coming down. So, while there is significant noise here, it's not enough, and you can most probably detect it, but it's going to be hard. But if I really increase the 0.25, all of a sudden you notice, there is just no information here that allowed me to kind of look for a sudden gradient change, which is where, perhaps, an edge would be.

11 - Increasing Noise.srt

it would be unfair of me to kind of just show-and-go, leave this topic of noise, without looking at images with noise. Take my example of the zebra image here. If I was to compute the gradient of the zebra image, I would get something like this. Really well-defined ridges as we have looked at before. Something that'll let me kind of figure out exactly the details I need. However, let's add some noise. Here you notice the graininess has shown up. And while there is still well defined edges and you can see them, let's see what happens when we do that variant gradient computation on it. And here, I'm showing you the gradient magnitudes. More noise has started showing up here and the lines here are not as well defined. Even more noise has been added. It's really getting hard to connect these lines up anymore, and in fact more of the image is starting to have the kind of flavor that we saw on the 1D example previously. Even more noise has been added. Human perception is very good. You might still be able to see the ridges and stuff like that, but purely looking at the signal. You see absolutely no information here anymore. So as we go from no noise to a lot of noise, we , the gradient magnitude here now seems to not be able to kind of showcase how we would actually be able to locate the gradient changes. The rapid or not rapid changes that, where perhaps an edge might be. So any guesses what we would have to do to help us still compute gradients that would be used for doing things like edge calculations? . Well, simple answer is we should be doing smoothing to an image before applying any kind of gradient operations. So take this original image, and if you run it through some sort of a filter, remember all the blur convolutions and Gaussian convolutions and stuff like that we've looked at. If we'd apply it before, we should be able to have, remove some of the noise, and then use that to compute our edges or gradients that could be used for other things, including computing edges.

12 - Convolution and Gradients.srt

Now let's remember, the whole concept of convolution and gradients, again, recall that convolution is taking our, kernel, involving with the input to get a new image. So this is of convolution, we'll now need to ask the question, what happens if I take this convolution formulation and just do a derivative of this in x, , we would do it separately for y, if I was to do a, derivative of this, means that I would have derivative of the convolution. To help us, let say if D is the kernel to compute the derivatives, and H is the kernel for smoothing. So , what we're kind of saying is, dell of the x could actually become now a operation because we have actually represented that as a kernel to compute derivatives, and H is a kernel, for smoothing, we can rewrite this formulation as, so again, if D and H are known, we could now define the derivative and smoothing operation as one formulation, D, convolved with, H convolved with F. , remembering our understanding of the basic properties of convolution, we could actually do a derivative of the kernel, and then just apply the convolution of the kernel itself directly to the input image. So we could actually now not just have to do, this process and then come up with the derivative, I could take any kernel and keep it in my storage as a new kernel that has a derivative itself, and use that and apply directly to the image. So this actually becomes an interesting way of looking at things, it's widely used in how we do types of image processing, types of techniques to extract gradients.

13 - Gradient to Edges.srt

So now let's look at the whole pipeline of how we would go for an image, to extract an edge image out of it. Again an edge image would be showcasing where most of the contrast changes are in an image and best represents the changes in the spatial x and y of an image like this. As we noticed before first thing we want to do is we want to suppress noise. You want to run some sort of a smoothing operation, a blur kernel using a Gaussian kernel to be able to get rid of some of the information because as we noticed, the gradient image calculation is really sensitive to noise. , next step is we compute the gradient image after the image has been smoothed out a little bit. And you would get something like this where all of the intensity information, where the gradient is the largest, is shown by the white lines here. And this is a gradient magnitude image. Third step is we apply some sort of edge enhancement, which means is that we will filter this image for contrast. Bring out the most value of where the gradients are much, much more higher than anything else. Otherwise, if you notice, this image is just black and white, with lot less white. We want to bring out the white information here to actually have more local information around the whites. because we don't want to lose any information because we've just linearly looked at the space between zero to 255. So we want to kind of enhance this data structure, which is an intermediate one from here to there so we can actually start looking for edges more carefully. Next thing we want to do is we want to start localizing edges. Which means is we need to determine which local maximum from any filter output is actually referring to edges as opposed to noise. So in essence we want to kind of look for filter responses, and start attracting it various types of ways. A differential that lets us get rid of noise and pay attention to edges. Many different methods for this exist. I recommend you to look at the text for this. I'll just showcase one of them in a bit. Finally, the last step is, we want to threshold information after we've done all of this, and also sometimes do something which is referred to as thinning, which is a morphological operator on images that lets you combine pixels and close bytes into a much more thinner representation. because in essence we're looking for a fine resolution image where the edges are likely to be in this, image. The output is a image like this, which looks similar but if you notice now it's mostly just one pixels of lines everywhere. It's very hard to see because again the way it's represented here, in black and white. But you'll see many examples of this as you do this work yourself. But, this is one of the best ways to look for an edge image.

14 - Canny Edge Detector.srt

Now I'll actually talk about one specific form of an edge detector. Perhaps the most widely used edge detector out there, referred to as the Canny edge detector. You start off by filtering the image with a derivative of a Gaussian. Remember, the whole concept of, actually, you can actually just apply the Gaussian to the derivative operator and generate a new kernel. And that's what this process is about. , we would actually filter the image with a derivative of a Gaussian. And that will give us a magnitude and orientation of the gradient. So this is my original image. These are the two derivatives in x and y. Again, just for showcasing purposes, I'm showing you some results. They are not the actual outputs all of this using this process, but I want to make sure that you guys understand what's going on here more rather than actually giving you the actual outputs. This is, , the magnitude of the gradient. And, , the, the angle, the orientation of the gradient everywhere. We've looked at how to compute all of this so far. So these are the first two steps. The third step of computing edges is, again, taking this magnitude and doing some local processing to enhance the edges. So in Canny edge detector, what is done is non-maximum suppression. Which is all about thinning the multiple pixels. So there are lots of pixels, for example, here you see right next to each other. And taking these into a single pixel width line. So anywhere where I see a lot of these types of regions that seem to have more than one pixel, we want to kind of start combining them into single pixels and lines of single pixels. These are, again, approximations to kind of look at what's going on. But , in essence, it comes down to, just take the wide ridges, remember those ridges from the way we've looked at a height map of these types of things, and reduce them down to something that's one pixel width. The fourth step in Canny edge detector is taking the gradient image again and coming with a method of both linking and thresholding pixel groupings from one level to the other. So in essence, what we do is, anywhere out there we would define two thresholds. These two thresholds would be the low value and the high value. And then use the high threshold to start an edge. So wherever there's a high value information. again, this is just for demonstration purposes. I would, let's say find the high value here. I would start the curve at this thing. And , anywhere in the thing wherever I find low values, I'll continue finding the curve. So using this kind of a technique, we can actually start kind of building more local edges out of it. Doing this process and kind of doing various types of filtering mechanisms to keep on enhancing them within the thresholding that's going on, we should be able to generate a edge map. So here you see now a complete edge map. All of the edges are one pixel. Sometimes they form complete lines out of it. , this is still an image, and these are not line segments per se, but then there are other methods that can be used to actually combine these into line segments, you know, Hough transform methods and stuff that you can read about on your own. And we will not cover them. We will actually come back at looking these types these types of edges for some other work in a later lecture. This, , is the actual output of edges of the original tiger image that we were looking at. Just to reiterate, these were the steps we went through. We filtered the image using a derivative for Gaussian, found the gradient information, did non-maximum suppression to find the ridges, down to single pixels, then once we took these single pixels, we started linking and thresholding the images, or, the edges. We're looking at the low and high and use the high threshold to start edge curves and the low ones to continue it to get an output like this.

15 - Summary.srt

So, to summarize this, we brought together concepts of convolution and correlation, specifically cross-correlation, to compute image gradients, use that to compute image edges, we talked about how smoothing, is important in computing gradients, and I showed four different ways of actually using, a specific type of kernel and a pipeline to compute edges from images. Next, we're going to take a little bit of a different turn in our work, so far, we've been talking a lot about images, we're going to go back and now talk about cameras, optics, and lenses, this will allow us to start thinking about the computation photography pipeline, and a little bit more carefully, and then we'll actually return back to edges and its detection, as we get into whole concept of feature matching. We'll also be looking at very ideal ways we can actually look at an image not just at its own scale of resolution but also multiple resolutions of the image. More exciting stuff is on the way.

# 03-01 Cameras.txt

01 - Intro.srt

As we talked about computational photography, let's not forget the most basic element of computational photography is still a camera, camera like this one. This was actually built in the, about the mid-1950s, and was given to my father by his father. And I was lucky enough to get it from my father recently. This was the first camera I ever used to take my first ever picture. The bottom line with all cameras is they're still the same in principle. What they do is they capture rays of light and get it to a sensor, which can be a digital sensor or film. And that pipeline is then used to create a photograph. In this lecture we will talk about that pipeline. How rays of light are brought onto a sensor to generate a picture. The most basic example of that camera is a pinhole camera. And the concept of pinhole photography was known even in the early 300 BC. But, in about the early 1800s, that was converted into creating cameras. And from there on, a lot more efforts went into building optics and lenses that were then used to create better cameras. In this lecture we'll talk about the pinhole camera. And we'll talk about the use of lenses to actually get the right kinds of rays of light to the right location, so we can actually store images.

02 - Lesson Objectives.srt

The specific lesson objectives for this lesson are, we will look at the basics of what we had started off in our introductory lecture, which talks about how we're going to go from rays of light in a scene to pixels. To help us understand this, what I'd like to do is first I'm going to introduce a concept of a camera which has no optics. And, then next thing we will look at is add lens and our optics into the camera, so we can actually start seeing what other foundational elements within a camera. And, just to kind of provide us with the mathematical foundations, we're also going to look at the lens equation on how light goes through a lens and where in an image plane is it formed to create a 3-D scene on a 2-D plane of an image.

03 - Recall Context of Computational Photography.srt

Now recall from our previous lectures, where we had talked about how computational photography is a process that takes information from a three-dimensional scene, and goes through a series of different steps to be able to generate an image that a user can interact with. So in the whole traditional aspect of computational photography, we start 3D scene, that is illuminated, optics is used to be able to get information from the scene onto a sensor, which converts it into pixels. And we've looked at in detail how image processing techniques can take the pixels for enhancement or even for analysis of images, like finding features that would be then used for various computational photography processes. The thing I want everybody to remember is, what we're interested in is going from rays of light in a 3D scene, and going to be able to generate pixels that can be used for various types of computational processes. , we want the computational processes to also impact illumination, optics, sensor, not just image processing. So, one thing I want to emphasize also is that, in the study of computational photography, we do rely on various types of disciplines, ranging from optics, sensors, computer vision, computer graphics and image processing, and they impact each and every aspect of this pipeline, of we're trying to get information all the way from 3D scenes, to actually generate an image that could be processed and displayed. And these are the disciplines that we have, kind of will be studying in different details to help us build the foundations of computational photography. We already looked at image processing to kind of help us extract information from images, but we will be looking at more scene analysis and stuff like that from optics and also from computer vision. And we'll study things like sensors, especially camera sensors and how light is actually captured.

04 - Pixels vs Rays.srt

So one thing I want to emphasize in this part of the lecture is that the basic element we want to be able to do analysis on most of the time in a 3D scene is rays of light. And you want to be able to use a rays of light to be able to extract information. So here's a beautiful image of a sun that actually if you see this image, all of the rays of light are coming in. And we want to be able to kind of model a lot more information about how rays of light from this light source are hitting different parts of a scene. And, , it's because of the phenomenon of this being hit these tree branches being hit, and us being able to see it, that kind of gives us the more visual feel of the scene. So we've looked at previously, and we've talked about the whole concept that an image is a 2D area of pixels and that's how we actually capture a 3D scene into a 2D image. But we want to remember that the fundamental primitives at least in the whole analysis of a picture of computational photography process is rays, you want to be able to be able to look at information that's coming and hitting the scene. And then, getting reflected and we want to be able to capture as much information from the environment based on that illumination and the rays that are hitting the scene. One of the things that's important when you start thinking about rays of light is that they follow a path from the source to the scene. So therefore, we can start using things like geometry of the rays of light to be able to extract information about what the scene is like. And more importantly, what we are interested in is how we can actually use computation, that can control all parameters of how the optics and the sensor, that is, the way the process of illumination goes, the rays of light, to go through optics, to the sensor that actually allows us to generate those pixels, that we can do image processing on. So, one of the important things for us to remember now is that we want to be able to use computation, to control how the illumination, which would then create the rays of light, that would be then controlled again with the optics to create information on a sensor that would generate the pixels that we actually will be doing processing on to be able to generate an image.

05 - Evolution of the Camera.srt

The earliest versions of the camera have been known to have existed from 300 BC where even philosophers like Aristotle were actually looking at how they can capture light onto a background plate and be able to use that for looking at scenes. The earliest forms of the camera started appearing in mid 1800s, and this is one of those traditional classic examples of a camera where a box with a lens or an optical capturing device, which could be a pinhole, was used to generate images. Then came various types of processes where you could now be able to save those images onto print forms, and this is various types of printing mechanisms that started in the early 1900s. 1948 is where we started getting more of these compact representations. 1986 perhaps is when we start seeing examples of these disposable cameras. This is an instance of a first generation digital single-lens reflex, or SLR, camera, a professional camera that was a digital camera came about in early 1990s. And this was about to 1 to 2 megapixel resolution image. And more interesting thing was when I started my career on research and computation photography, this camera was available for about $25,000. And I actually got access to going to the lab to be able to use this to take a few pictures so I can actually start get playing around with the images that came from these specialized cameras. , 2000 is where the first generation of mobile cellphone cameras started coming about. So if you really think about it, we've gone a long way, where first instance of a real camera that actually was into a form that we think of a camera to be in right now, was in mid to early 1800s, to where we are now. And , cameras have taken very interesting shapes and forms since then. Interesting things to note now is cameras can be really small. Just few examples. As small as a pencil or even as, you know, small enough to be on your, just your fingertip. Most popular cameras that you see these days are, , the cellphone cameras. They provide a good resolution. We played around with them a lot, I'm sure. And , these days, people wearing them in different forms, including, , much more variable cameras like this one. One thing I'd like to note is one of the other things that has changed is how we interact with these cameras. In the old days, when we had the cameras in much more of a bigger form factor, we had much more of a formal relationship, where we use this tripods and all that kind of stuff. Then we went to much more of a standup, but if you notice, most of our camera interaction is much more now stand-away, and, you know, we use viewfinders to be able to see it, and it's become more casual. We've gone from more formal types of photography to more casual photography more and more now

06 - Single Lens Reflex Camera.srt

So first let's look at a little bit more in detail a, a very well-known form of a camera, a single-lens reflex camera. This is an example of one. It has a lens and the whole body. I'm going to take the lens off. And it now can see more details of this camera. We'll look at it much more in detail in a few seconds. But you see that there are certain aspects of this camera. There's a viewfinder. In there is a mirror which moves away. We'll see examples of that a lot. And , this is where the shutter release is. So this is a standard film camera here. I wanted to also show you this. Most of your concern that I've exposed film, this was an exposed film already. We will be looking at more detail what a film like this looks like. But this is again, you know, one of the more widely used form factors for professional photography. So here I'm showing you now a cut out of the same camera, or similar camera. Here you can see all of the details that we looked at. This where the view finder was. The focus, or zoom ring and the lens. And I took this out before I showed you any of this stuff, were right here. The shutter release button is right there. This is the one that when you press opens the mechanism, so let the light in to where the film would be. The lenses are all in the mechanisms right here. There is a diaphragm. And we'll talk about the diaphragm and all of that kinds of stuff. Because that's where all of the openings are controlled between these lenses to allow the right kinds of light into the camera. And this was a film camera, so there is where the film is. In the new modern digital cameras , the film has been replaced by a CCD sensor or CMOS sensor. Again, something we will be looking at in much more detail later. And all of the focal plane shutters are all mechanisms that are built in here. That actually let again, different things happen in terms of how the focal planes and everything as I suggested, to get the right kinds of light into, to where the film is. Now, , you noticed, and I showed you the thing, there was a glass here. single-lens reflex, this thing opens to allow you to get the light in correctly. And this, the prismatic sensor, that lets the light into the viewfinder so the viewfinder can see exactly what the lens is seeing. And as soon as the shutter is released, the mirror moves up and then the light goes and hits the film. So you actually get to see exactly what you're seeing will the one, the image that will be registered. And actually then imaged on the sensor or the film

07 - When you take a picture.srt

So while we now know how a camera, or what's inside a camera, let's go back to basics and begin understanding what is happening or what we are trying to do when we take a picture. What we're trying to do is capture a 3D scene and create an image of something that can be displayed to a user and, , what we're going to capture is the light in the scene. In addition to that, what we also want to capture is the 3Dness of the scene. For example here, there is a lot of geometry in the scene, right? Which is shown in the perspective. For example, these two train tracks are kind of converging at a point. I mean this is actually something in the imaging field known as the vanishing point. All things like this, if I was to draw a line here on this train track and another one on this train track, you kind of see that they're converging far away at some point. This is showing perspective, that is, things that are farther away appear smaller. Things that are closer appear you know, larger. So this distance is large here but, , train tracks kind of get smaller. So what we're trying to capture here is the geometry of a scene. In addition to that, we're also trying to capture from a scene varies types of light changes. So here, , a scene just showing sand with water. You see specularity, you see waves, you see reflections, you see diffusity of an image and all that kind of stuff. So, , in this image what we're trying to do is capture all of the light variations. Besides just the color, the specularities and stuff like that. So in essence, what we are trying to capture is the light scattering. So the two essential things we are capturing in a scene like this, shown by these two images, is the geometry, that is the perspective changes, the size of the space, the 3Dness of the space and how the light is kind of reflecting off the environment across the board. To achieve this, , that's what the camera is there, it's trying to capture and depict the scene from 3D into a 2D representation. , available to us is a camera when we looked at the insides of a camera, we saw different things it has. First thing is, , the optics and the lens of the camera. Here I'm showing you that by just opening the aperture of the camera, you can actually get more light into the camera. More light into the camera is an important thing and also, how much of it will actually get into the sensor is another part of the whole process we want to try to understand because, in essence, what we are trying to do is capture light. So the other thing available to us, , is the sensor itself and the sensor itself could have a variety of different characteristics on how it captures the color and the light variations that are captured from the scenes. So, in essence, the light goes through this, when it hits the sensor, and that's what we saw in one of the slides earlier, it also kind of adds a variability on what we would capture. So, in essence, going from 3D scene to a display, what we are going through is a pipeline of wanting to capture perspective and light scattering from a scene using the light opening and what the kind of sensor we used to be able to create an image, a depiction of the 3D scene that we're trying to capture.

08 - Cameras Without Optics.srt

So to help us understand how imaging can be done with cameras, let's try to build one from the basic first principles. Help us do this, I'm starting off with a simple tree, shown here. , it's kind of attempting to show a 3-D scene. And what we're interested in is capturing this 3-D scene in a way that it actually captures the geometry and all of the light variations of this. , nothing of the scene would be visible without a light source, right? If you walk into a dark room, you can't see anything. There has to be a light source, there could be an external one with sun or simple light bulbs and stuff like that in internal situations. Light source from this hits the tree, and then it reflects the light back and it's the reflections from the scene that hit our eye, that's why we see the scene, and again we want that reflections to hit the camera, and store that image, and that's what we're interested in. So in essence, any light source illuminates the scene like this, and it's these reflections from this, kind of what we see, and again that's one of the reasons we see color, we see specularities, and stuff like that, depending on the type of surface it might be. For example, if it's kind of reflecting black, well it means that in essence it's absorbed everything. If it's reflecting white, it means it's reflecting everything. And you know, color come in, because again, the color properties of the surface that actually has those color materials reflect differently, and that's why we see those colors. So this in essence, suggests that scenes have to be illuminated. Now let's try to capture this information. So to do this I'm actually going to pop up a, a screen of some sort, and I'm now interested in saying this is my screen. I want to capture this 3D structure onto the screen. We can actually refer to this screen as a sensor. Now we will talk about different types of sensors but let's assume this is a sensor that actually captures this light and saves it for us. But, before we get to the saving part, let's see how light is actually created or gathered on this surface. , as I said in the last slide, if the sun was illuminating this tree, each and every point of this tree would be reflecting back the light. And that light would be hitting our eyes, and that's how we see it and all that kind of stuff. Well, so in essence, each and every point is reflecting light and it's hitting the sensor. And , no image is forming on this one because it's coming across in all possible ways. And , at present we're not thinking of this as a sensor. Let's just claim it to be a screen and all of the light is there. And it's for this reason, you walk around in an environment, you actually don't see reflections of everything exactly on this environment. What we want to now figure out is how to get these beams of those rays of light focused. So, to achieve this, let's put an obstruction in front of our screen. And here I show an obstruction to be , a big, you know, let's say a object of some sort, except that I do have an opening. So, because of this obstruction, what will happen to all these sources of light? Only the ones that actually go through this obstruction, this hole, would make it to my screen. The rest of them would just reflect off and diffuse themselves in this part of the environment, and nothing will get through, only the ones that get through would be through this opening. So let's look at exactly what that means. If a ray of light goes from the bottom of the tree, it goes through here, it goes through the hole and hits the screen. Let's see what happens on the other end. Another ray of light goes from the top, goes through here and hits the screen. Simple geometric kind of a way of looking at rays of light. Remember, rays of light are the ones we are trying to capture to create an image. So one thing interesting is bottom is there, top is there. So just by looking at this triangles, we can say that this tree should be upside down when it forms on the sensor. And again it makes perfect sense, this point is actually coming in and registering here. This point is coming and registering here and all of the points on this tree would be going through this small hole and creating an image there. So that in essence is basis of a camera per se.

09 - Camera Obscura.srt

So this is the basic definition of a pinhole camera or a camera obscura. The basic idea is that if you have a 3D scene and if you can create a small device like this one, which has a small hole, a pinhole, then all the light from this 3D scene would go through this pinhole. And if you put some sort of a screen or a sensor inside this box, it'll appear upside down inside this box. The simplest definition of a camera, therefore, would be as how light goes through this hole, is generated upside down in the back of this box. And if you notice, all the light is going in from one hole and appearing upside down here. And now, , the simplest thing we can do to make this into a camera that would preserve this image, would be to be able to store this image in some form or the other. I'm just showing you a few more examples. These are the classic examples of pinhole cameras. And I did mention earlier that this definition of a camera has existed from about three to 400 BC. Where you would put a light source here, go through a pinhole, and an image would be formed upside down inside the box. And this was a classic method used in the old days also, where 3D scenes like this would actually be captured through a pinhole upside down. And people would actually use various types of sketching mechanisms to preserve this. And they would actually do sketching or painting on a wall to preserve this. , we have many different methods of preserving this, and there's a whole lot of stuff about both films, and , digital sensors that can do this.

10 - Pinhole Photograph.srt

Now I want to show all of you a classic example of a pinhole photograph. Here is one and you notice that very interesting things are happening in this one. All lines are straight. There is no form of distortion. Also if you look at it, the whole image is, has infinite depth. That has everything is in focus. This is a pinhole where there is no lens whatsoever. somebody built a box, put a small hole in it, and put a film or a censor on the other end and generated this image. Actually it looks pretty nice. And what's important is there is no form of blurring, no de-focus artifacts, and each and every line if you were to draw it, would appear completely straight. when we start going over from pinhole cameras, we'll notice that we will be compromising on these types of perfect capabilities that a pinhole camera might have.

11 - Pinhole Size and Image Quality.srt

So now, I’m going to give you a very concrete example to suggest the importance of pinhole size and its relationship to the image quality. Let’s start off with our simple example, our tree again. This time around again, we have an obstruction and a screen, a sensor. What we will do, , is we will create this quote, unquote, the eye to the world, a pinhole in this obstruction. So then, we can actually capture what's been actually visible at this point here, to the screen there. , as we expect, all rays of light going from this point will go and form on the other side here. And all rays from here, and all rays in between will actually use that to create a inward a tree on the other side. Just for sake of explanation I'm showcasing a example image here, this is not exactly the same tree here just showing what's going on. That this would be a nice crisp image if this was a nice crisp hole because everything would be going through perfectly. And it would get a nice crisp image on the other side. Now let's look at another example of the same situation, again a screen a tree and obstruction. Here I'm actually, going to create a little bit of a bigger one hole than this one. And we going to just do same kinds of tracing of rays. So for example, I take tree of light. And now, the hole is bigger there is one going form the top of the hole and another one from the bottom of the hole, when it gets here. And in this case also, one ray is going to the top and to the bottom. Because again, the hole is bigger here. I showed you just one that in assume, it assumed that only, you know, the hole was small enough that not many rays would go through it. You know, just by looking at the triangles here. If this size is bigger, you can expect this to be much, you know, getting bigger as you get farther away from this point. And, , what would happen in this instance is, because of these two rays. The trees would kind of start cut into, you know, inter, interfering with each other as they form on the surface. And as I put more of them there, you would see that, , on the screen we have a blurry image. putting in more and more these and the tree, , is looking blurry. think of it as nothing else as the, because of the size of this the triangles are getting bigger and this is the blur point here. , the resulting example could be something like this where now, , it's much more blurry. So this starts implying is, that this size of the pinhole is an important parameter in how well we can capture images. Now, one thing we haven't start calling is, we want to start actually referring to this opening as of the pinhole, as the aperture. Again, remember what we looked at when we looked at the insides of a camera. The size of the aperture was the opening. because that is the amount of light that was allowed into the camera before it reached the sensor. Well in essence, this is what it is. Right? It's the aperture. And the size of this aperture is extremely important parameter for us to know as we move towards capturing light into a camera. Just as a note, these images are simulated just to kind of showcase this point.

12 - Light Diffracts.srt

Now let's look at a couple of other important things about light itself. One thing about light is it diffracts. because it's the, you know, it's light is best represented as waveforms, and the wave nature of light suggests that when it hits, and it's coming up this way, and it hits. Hits a surface like this, and if there's a point, once it hits this point it is going to start creating diffraction patterns. So smaller the aperture, would actually result in more diffraction from this point on. So in essence what this shows is that once the light hits this point it actually starts creating these diffraction patterns rather than going through straight. If this was no surface here, it would continue going straight. But once you put an obstruction and a small hole, depending on the size of hole, you'd actually get diffraction patterns. This is best also shown here, through the light frames of This is also best shown here. Light is coming in, once it hits this. Because of the opening, a diffraction pattern is created and you can see these diffraction patterns as rotations. And, again, there's a little bit more intensity here that dissipates as you go there.

13 - Effect of Pinhole Size.srt

Another way of looking at that whole concept is this schematic here. Let's assume I can put a light source that actually has a shape of a plus sign here. I have an opening with an aperture which is much bigger here. When light goes through here, what's going to happen as we've discussed in the case of the tree, it's going to get a little blurry. Similarly, if the hole is small, it'll be less blurry. Another hole instance of this would be is when we actually have a point light source. This is not a point light source. These were, you know, if have different shapes in it. A point light source is, has no dimension. It's just a very high intensity beam. When it hits this point here, because if it's a very small one, it's going to go through correctly, and there will be no kind of diffraction patterns here. But if I made it even so much smaller, related to the wave length of this light source, it's going to start creating diffraction patterns. So we'll have another whole artifact on this image. In essence, what it comes down to is, we need to figure out how to control the various types of stains in this image, which suggests that a larger pinhole, which is right here, creates geometric blur. A smaller pinhole would create diffraction blur. So this is geometric blur, a diffraction blur. And a best pinhole, , would be an ideal in creating other images that we want, but will let in a lot less light. We haven't actually discussed the whole concept of amount of light, but it should be obvious. The smaller the hole, the less amount of light will pass through. So again, let's look at the whole concept of pinhole size. And let's try to study it with a little bit of math. Let me get a few terms across. First term is d, which is the diameter of the pinhole. Again, we also refer to that as the aperture, the size of the opening. because the amount of light goes through this one. Another term we'll use is f, which is the focal length, which is the distance from the obstruction where the pinhole is, and to the screen of the sensor. How far away should that be? because that's where everything is going to be focused and best produced. Let's call that f, and it's also referred to as the focal length of a camera. Another parameter we'll use is pi, which is the wavelength of light. So in essence this captures, is the you know, the physical characteristic of the light itself. Using this, we can come up with an equation which suggests that the size of the hole depends on the focal lamp. How far I want the subject to be or the, the image to be captured. They'll be of length in this form of a equation here.

14 - Replacing the Pinhole with a Lens.srt

Now remember the image that we had before. A tree, light going through a small pin hole, generating an image here. Now as I said, smaller the pin hole, the less amount of light will actually pass through. So this would be a much darker image. What we want to do is open this up a little bit more, so more light will go through. Again, the size of this thing is going to be important as to the amount of light that goes through from the obstruction to the other side to form an image. So one of the best ways, , to do this is to replace this hole with a lens. So here we still have our tree, our sensor or a screen, and in place of this we want to replace it with a lens. But before we do that, let's look at the characteristics of a lens. Here I'm just showing an example and now I can add a lens in between the 3D scene, in this case a tree, and the sensor. One of the interesting things about a lens is that when a ray of light, a wave front hits it, it can be made to converge to a point. And that's what we actually want to be able to do is we want to get all the rays of light going in and hitting this lens and then converging, to a point. And then that would allow us to get a light pass from a much bigger opening, as big as this, as opposed to just a point. But also we want to actually simulate exactly something like a point, where all the light would be. So this ray of light goes through, hits the lens, and, , now is converging to the screen at this point. A ray of light from the same point here will go through the lens and converge to this point here. And that will allow us to create the top of the tree. And another one would go through the lens at this point and converge. And so all three lines are converging here, and this is how would I be able to construct just the top point of the tree here. And , rays of light are coming in from the entire tree and go through the whole lens to generate an image on that side. And this is how we get an inverted tree, with the resulting image simulated here.

15 - Geometrical Optics.srt

I'm going to introduce some simple concept of geometrical optics to help us understand how light traverses through the lens and what happens on the other side of the lens after light has passed through it. Let's look at a simple example schematic of a lens. This is referred to as a principle axis and we'll look at as light hits this lens and what happens on this side. So this is a set, is the principal axis is a point here which is also the focal length. If you remember the whole concept of the pinhole, that's where we want the image to be formed in the camera obscura pinhole camera. In this case now when light hits through the lens let's see what happens. So any light that's parallel to the principle axis, when it hits the lens will converge to the point where the focal length of that lens is. So parallel rays converge at the focal length point of a lens. Any ray of light that goes through the point of the principal axis will go through straight unchanged. It will not have any deviations. So any rays of light that pass through the center principal point of a lens will traverse through without any divergence. So, in essence, this is the point that behaves like a pinhole.

16 - Ray Tracing with Lenses.srt

Let's look at the same setup again, lens is a symmetric so the focal lens will be on both sides of the lens, lets imagine that I have a 3D object on this part of the scene here, and, the image plane is being formed on this side. So I put an object here, remembering what we, looked at in the last side, any parallel light goes through, goes through the focal point here, continues on, anything that goes through, the optic will point here, goes to unadulterated and then the third line, , the opposite of the converse of this one, anything that goes with the focal point will become parallel on the other side. So in essence, these three lines, will converge at this point and this is where the image will be formed, so this is where the image point is for this object. So, rays from points on a plane parallel, to the lens, focus on, a point that is far away here, and we will actually now come up with an equation that connects the object distance from the lens, to the focal length to where the image is formed. So here, would be my tree again, as we've noticed, on the other side, it'll be formed upside down, just calling this distance of the object from the lens itself to be o and the distance to the image as i. So this is what is referred to as a lens equation, and mathematically, , it's 1 over the distance of the object from the lens plus 1 over distance of the image from the lens is equal to 1 over the focal length.

17 - Summary.srt

So, in summary, what we discussed in this lecture was some of the foundations of how a camera works, presented the concept of a pinhole camera, and then introduced the concept of optics, and how lenses can be used, and what role they play in ma, making a camera. And how they used to be able to generate images that could be stored within a camera. What we will do now is building on these concepts, we'll look at focal length, aperture, if you recall aperture was nothing else but the pinhole size, and also we'll add things like what a shutter, that is how long the opening is, the light opening is used to generate an image. And then we'll actually talk about how sensors are used to be able to store those images. For additional references, I've listed two books here and also additional material is available on some of the sites listed here, and I'll be providing more resources on the website. And thanks to Mark Levoy, whose slides some of cost slide concepts were used in making these slides.

# 03-02 Lenses.txt

01 - Intro.srt

In the last lecture I showed you this traditional old camera from 1950s. An important part of this camera is still that it has these lenses that are used to focus the rays of light into the sensor chamber. This is a more recent digital SLR, this is the camera used these days. And , it also has a lens. And one of the things that we will talk about in this lesson is what kinds of things that a lens like this does. And , it has things like focal length, variety of things of this lens actually then control the view that we actually get from the image or from the scene that actually is represented on the image. And we also talk about things like what sensor sizes these types of cameras have. Varieties from digital SLRs to full frame ones to simple cellphone cameras that we use. We'll also talk about how images are formed and also things like perspective projections. The simple mathematical representations will be introduced to show you how we can actually go from information in a real 3D world to what's actually seen inside a camera.

02 - Lesson Objectives.srt

The specific lesson objectives for this lesson are, I'm going to talk a lot more about the concept of focal length, we will and then discuss the whole issues of how the field of view of a camera, or a imaging sensor are related to focal length. We will also discuss the importance of, the sensor size for each camera, and we will look in detail at the whole image formation pipeline and how images are captured and discuss the basics of perspective projection, that is important in trying to capturing the 3D aspect of a scene.

03 - Recall Ray Tracing With Lenses.srt

You may remember this from the last lecture where we talked using a lens and tracing rays of light through a lens and see how an object is then created on the image or a sensor, and that's how we go from a 3D scene to create images. Here I'm starting with a simple object and let's see by just doing simple ray tracing what happens to this, a ray of light goes through the lens, and , goes through the focal length here and it's now on this side, another ray of light that goes through the optical center of the lens, which actually acts, if you recall, like a, pinhole, goes straight through, and similarly, another light comes on, and now , we're seeing that an image should be formed here, the image is upside down. I'd also introduced in the last lecture the whole concept of a thin lens equation which is shown here, which relates the focal length of an image with respect to what the distance is of the image, on this side of the lens, where the object is on the other side. Couple of things to clarify also and point out here, this lens is a thin concave lens, which is the shape like this, it's also sometimes referred to as a positive lens, because all rays of light converge, inside, to allow it to create an image on this side. , convex lenses that have shapes like this are also used, sometimes they're referred to as negative lenses, and most of the time, when you buy an expensive lens for a camera, it's a combination of a series of these types of concave, convex lenses, that move together to create an effective lens that's used to be able to allow you, to take the, you know, rays of light from a scene onto your sensor. And we will be seeing some examples of this, but there's a lot of material that you can look up on your own to kind of study the importance of these types of lenses.

04 - Image Formation.srt

Let's study some of the basic premises of how an image is formed from a 3D scene. We start off with just a simple representation of a 3D scene, we have a lens, and , because of this lens, now we know that we can actually create an image of this 3D scene on the other side of the lens. Here, I'm just now showing you a simple paradigm where, again, a lens, all the light goes through, the lens here generates a 3D scene. And this is all the light from this thing is now going through this lens and captured here. Most of the time we will actually be putting a sensor, where we want to capture the image, at the focal plane of a lens. So at f, distance away from the lens, we want to be able to capture the image. And this is where actually we want to be able to put our sensor. And that, you will see various examples why that's important. Let's look at a few examples of what happens if certain things change in this image formation pipeline. Again we have a 3D object, lens, and at the image being formed at the focal point of this lens. So as I said, now I'm going to push the sensor or film or any form of a screen here that's going to be where we want to capture images. You may remember this was the concept that was there in the pinhole camera in the previous lecture, too. We always wanted to create an image at the focal plane away from the lens. So what happens if I was to now move this tree further away from the lens? So everything else is the same, I've just moved the tree here. By doing simple geometry on this one you will notice that when you take and trace the rays of light through the lens, a tree is formed on the image sensor. Except that now, , it's much smaller. This is the image size here. it's much smaller on this side here. And that's valuable because in essence as I move the tree farther away, it should look more distant. So this is exactly doing what we expect it to do. And if I had let's say two trees, one in front and one in back, the further one should be looking smaller. So this is kind of giving me a sense of depth of perspective, which says things farther away would be smaller. Now let's look at what happens if I move the tree inwards. Same lens, same situation, same location of the sensor and, , if you just draw the rays of light and do the projection again, you will notice that this time, the tree is bigger. So this tree is smaller and this tree is much bigger. same example applies, it's closer to a sensor, therefore it appears to have more, size and it appears to be larger. So this just demonstrates that by changing distances of the object to the lensing system, the camera here, we can now actually see closer or farther off objects.

05 - Changes in Focal Length.srt

Let's also look at examples of what happens when we change the lens that has a different focal length. Let's look at the example we have looked at before, a tree, a lens, an images of a lens here. , let's assume that this lens has a focal length of three inches. So this time around, I'm only going to change the lens and give it another lens with a different focal length. So now, we have another lens, this lens has a focal length of say six inches. Remember, we want the image to be formed, at the focal plane of this lens, so now, the best image that will be formed at this focal length, because of the fact that it's farther away would be larger than this one. So just, by changing the lens focal length, we notice that the image formed is much bigger. So this allows us to now be able to control how we form images on our sensor, by changing the focal length, we can actually play around and generate different types of images, and most times, the focal length f could be longer, or shorter. And we will see examples of the impact of this on what kinds of images are formed in a bit. Now remember, focal lengths are specific to lenses themselves when they're manufactured, most of times, , when you see complex lenses, you'll also get a variety of lenses composited together and the dynamics of these lenses, are the ones that give you the effective focal length of that lens. In this case, we're just simplifying all of our lenses to be just one simple lens here.

06 - Focussing.srt

Continuing the same talk, let's not talk about the whole concept of how images are focused. Here as I have said, image is being formed at the focal plane of this lens. All light is hitting there. It's converging to a point. And I should get a nice crisp image here. What happens if I was to now try to move the image further backwards? If I move it backwards, what should happen is multiple kind of copies should be available because none of the rays of light that are coming through the lens would converge at that point. They mostly converge, in fact they always converge at the focal point, so you should see a little bit of a blurry image, , I'm just simulating that here. Similarly, if I move the image plane in front of the focal plane, I should also get a blurry image. So focusing most of the time in a camera is done by moving the sensor, or sometimes the lens, forward and backwards to make, make sure that the image is formed at the focal point. And the distance between the image and the sensor plane where the image is formed is set to the focal length of the lens that's been given. So the focusing concept is where we move the lens in relationship with the imaging plane, which should be on the focus plane. By just moving those two things together, you can now make sure that the image is formed exactly at the image plane, which is now at the focal plane. So image plane has to be at the focal plane to be able to generate a smooth crisp image. If it's after or before the focal plane you will see blurriness. So focusing is done by moving the sensor or the lens in relationship to each other to be able to create a relationship where the distance between the lens and the image plane is exactly aligned to where the focus plane for that lens would be. In doing so, we guarantee that the image is going to be crisp and focused on the image plane. If it's in front, that is the sensor or the image plane is in front of the focal plane, or after it, it's behind it, you'll get a blurry image. Let's look at another example of this. Here I'm just showing you three images. Again, for this image here, if all the light is going through my lens, and it's forming on the image plane or the sensor here. , by changing this relationship, if I get this closer, if the object is farther away from it, you will get to see the smaller object. But in essence, what I was talking about with focusing is here, this is the best focus point here and how the lens moves with respect to this, is essential to be able to get a crisp image or a blurry images as are noticed here. We looked at this whole concept a little bit for pinhole cameras. Again, in that size , the size of the pinhole was the reason for creating blurry images or not.

07 - Field of View.srt

Let me introduce another whole concept, and that is the field of view of the camera itself. So in this case, we're trying to, , capture this tree with this lens, focal length 3, and this is what the image has generated. So, in this case, the field of view is this angle. Let me introduce a new term to help us understand how we can actually compute the field of view and refer to this as, that, and I will refer to that as h which in essence is the sensor size. So in this case, the sensor size is the entire region here, and therefore I can reconstruct the entire tree. But now based on this, we can start coming up with a formula that will let us figure out exactly what this angle would be. So given this H, we are interested in computing this angle, which we refer to as the field of view, and for simplicity's sake, let's say, call this angle theta for now. So we are interested in computing theta. Okay? To help us do this, first we can actually look at that there is a triangle here, and this angle theta is exactly the same angle theta here. Let's just take this half triangle. h goes over there, so this would be h over 2. This would be f. This is theta over 2. So theta over 2 is inverse tangent of h 2f. Correct? So in essence, theta would be 2 tangent inverse h over 2f, which is primarily what this equation is. To help us look at this now, let's take the same equation, but now, actually, we can create a focal length which is twice that one. , now I have this much larger tree. Except, let's say, our sensor is exactly the same size h. So if I was to use this sensor size, everything of that image would be cut off here, there, and therefore also means my field of view should also be smaller. This is my field of view. Which is the same, I guess, for this one. And , I can't see the entire tree. Let's look at this a little bit more carefully again. , the sensor size can be small, then the field of view is also small. And smaller sensors, , can capture fewer number of pixels and may also have other issues, like noisier pixels and stuff. So let's take the same example, except now I have h here. This was the equation we looked at, and this was my field of view. What happens if I make h smaller? Well, my field of view would get smaller and smaller. So in this case, let's say I go for a smaller h, which is this, my field of view is only now just this much. So that is because field view is directly related to both focal length and the size of the sensor. I will actually showcase this using real pictures next.

08 - Focal Length.srt

To showcase the importance of both the field of view and its relationship to the focal length lens I'm going to show you a series of pictures a friend of mine, a colleague of mine at Georgia Tech, Henry Christenson. Professor Christensen took from his apartment and this would be actually interesting for all of you to see interesting perspective of the Georgia Tech campus from the city of Atlanta. Here is the focal of the camera we're using or the lens we're using is 12 millimeter. This is the view, 24 millimeter, 50 millimeter, 85 millimeter lens, moving to a 116. 220 millimeter, all the way to 300 millimeters. So we've gone from 12 millimeter to 300 millimeter, and now, , you're seeing the iconic Georgia Tech tower in front of our football stadium. Right behind was the Coca-Cola campus. So this is an interesting way of kind of now showing you the values, to field of views really kind of getting focused in. Let's look at it a little bit in backward direction. And forward again. Let's look at the same images in a different form next.

09 - Focal Length and Field of View.srt

Here I'm going to actually show you the, focal length and the field of views using this chart, as we go in forward. So, the field of view was 120 degrees, in the horizontal, when we were looking at 12 millimeter lens. When we move to 24, 74 degrees, 15 millimeter, 40 degrees, 85 millimeter, 24 degrees, 116 millimeter, 17 degrees, 9 degrees field of view for 220 millimeter, and 300 millimeter shows 7 degrees, so much closer viewpoint. Interesting way of looking at how we can go from a focal length of 12 millimeters to 300 millimeters and how the field of view changes.

10 - Sensor Sizes.srt

Sensor sizes play a role in all of this too. Here I'm showing you the aspect ratio and sensor size of a 35 millimeter standard, full frame camera. So this is a 1 to 1 full frame, 35 millimeter, 24 millimeter. Sensor that impacts the field of view of a camera that you use to capture images. Slightly smaller. This is another sensor size that's widely used 28.7 to 19.1. And the aspect ratio is no longer full frame, but 1 to 1.26. This is yet another well known and widely used. The aspect ratio in this one is 1 to 1.5. These days micro four-third cameras are getting more common and this is the sensor size for that. a micro two-third camera sensor size is this. Other sensor sizes exist, 7.25 millimeter to 5.33. now you notice that we're moving from a much more professional digital SLR all the way to handheld cameras or pocket cameras or more instant you know, cell phone cameras. This, for example, is a sensor for iPhone 5, 4.54 times 3.52 millimeters. And we can keep on getting other type of sensor sizes. But you noticed this is a sensor for a professional camera all the way to kind of what's in mobile phones like this.

11 - Focal Length vs Viewpoint.srt

Let's also look at examples of what focal length does in terms of what viewpoint do you want to capture. Here I'm showing two examples. Again, our two friends that have come visited us before, and here I've shown two images. This image was captured with a focal length of 18 millimeter on a 35 millimeter sensor. That was a full frame camera. Distance to the first subject is half a meter, and distance to the second subject was two meters. In this case, I used a focal length ten times as larger 180mm. Again, the same camera. And, the distance to the first subject is no longer 0.5, but I moved 2.5 meters away. And, , that meant the distance to the second subject is also now 2.5 meters more, 4.5. Exactly the same scene, shot by two different focal lengths with the same camera. And you can note this that each one of them has a different impact. What this shows is that changing the focal length allows us to move back and still capture the entire scene. But, , changing this form of a viewpoint also does change a perspective change. For example, these two scientists are appearing to be actually much closer than they are. They are actually almost 1.5 meter apart. But by looking at this image, you can't tell. They almost look to be sitting next to each other. One of the best ways to kind of visualize this is to see the vertigo effect. I recommend you look it up on YouTube. But this kind of scene perspective tricks are widely used in movies like The Lord of the Rings trilogy, where you could see Gandalf and Frodo sitting next to each other, but looking much different sizes. , there's still approximately same human size, but in case Frodo looks much shorter. And this is exactly the kinds of tricks that were done in movies like this, by moving farther away and using a high-focal length lens to kind of show them to be different sizes. So this is the kind of tricks again that are widely used in movie industry. But just by changing the focal length we can actually change the perspective and the viewpoint.

12 - Camera as a Window.srt

So as we have discussed before, camera is , is trying to capture a 2D view of a 3D world, to an essence, it's a window into a 3D world. But , in the case of the, camera, the world is 3D, so here for example, the train tracks kind of give you a sense that as you go further away things are getting smaller and smaller, gives you concept of perspective. In this case, by just not looking at that side but just looking at this side, I get less sense of perspective, same image but just moved on to the side and showing the train tracks. This is an important part again, how viewpoints and field of view play into photography. So imagine this is my subject, and my camera is here, I'm focused on the subject except now this is my field of view from an 85 millimeter lens. I can move closer, use a 50 millimeter lens and still capture approximately the same details of the same subject. I can move even closer, and use now a 20 millimeter lens and actually have approximately the same field of view, but I really kind of moved up a little bit. We will see how this impacts our scenes, that we want to capture also.

13 - A Camera Model.srt

Let's now try to bring back some of the details on how images are captured from these cameras, and try to get into some representations of how mathematically we can look at all of these. Here we start off with an image that we want to capture, and this is my sensor. As we looked on the image processing and image representations, most of the time we represent an image in this form with x and y, or columns and rows. the 3D world has this coordinate axis, Y up X, and Z which could be sometimes, in this case, be respective as we go down into the scene. So imagine if this is my object in the scene, what would it actually look like in a 3D camera? So to help us kind of get into understanding this kind of concept of a camera model, let's imagine I have an object in this 3D scene. To help us, we will actually unwrap this. And now I'm showing the Z-axis, and this way X and Y, so I've kind of turned it around and looking at this scene from this direction, and that, that is the way I'm looking at it here. So this is my object, and again shown here. , I'm just going to give it some value, X, Y, and Z, which is this point here of this object. We can look at all the values, so X is this, Z is this, sorry Z is, Z would be this value. X is this value and Y would be up how high this object is. the tracing the ray through and imagine this to be a pinhole, so this is my pinhole of the camera itself. Ray goes through here. And on the other side, I should be able to now think about where the image will be formed. We'll get an inverted image on that side there, and it's on that side of the X axis. And this would be the value in the image plane on the sensor. And it's my sensor, this is where the image forms on that side of the camera. So the camera pinhole is here, the sensor is here, 3D scene is here. We know that, , the sensor should be at the focal plain. Now using similar triangles we can actually start finding out more information about where the values of this would be. The similar triangle says that the ratio of xi over f, which is xi over f, should be the same as X-non which is this value here. So xi is this and focal length is this. Similarly, X0 is this and in place of f we have the Z value here. So xi over f is equal to X-non over Z-non. And similarly for y, we can do the same thing and come up with this equation. We can simplify this to now be able to get the values of xi and yi with this relationship. So as long as I know, , the, how far the object is from the scene and where it is, I should be able to now figure out where the pixel value would be on that image itself. , we not, not all the time do we know these values from the real word, especially just from an image.

14 - Perspective.srt

Let's look at how the camera model and the perspective of the camera model is extracted from this. We can actually simplify this a little bit. What I can do is now take the focal length which was on this side and actually just start saying is, okay, let's mirror it on the front side of this too. Which means is now I can move the sensor here, and if I move the sensor here I can put the x i and y i values on this side. And one interesting aspect of this simplification is now I can see the image to be also upright as the object is. you notice that if this object was farther away, the image would be smaller. If the object was closer, the image would be larger. And that's exactly what we notice when we were looking at, , lenses and how the impact of lenses with moving objects was. Again, this is a idealized version. Pinhole is this point here. We would be replacing this with a lens. We looked at these equations before this is how we would compute the values of both x and y put in each and every object the location of the pixel that would be related to where this point is on the scene.

15 - Focal Length for Portraits.srt

Just as an aside, I'd like to actually show you an interesting example of what would be an ideal focal length for taking images or portraits. Again, I have my scene. I have my camera. I can use an 85 millimeter lens and get the whole field of view that I want. In this case, I'm using a simple example, a bust of Abraham Lincoln. I'm using this 85 millimeter lens and this is what the shot I get. I move closer. This is a 50 millimeter lens now looking at the same subject. Here's the output. It does get darker a little bit for a variety of reasons. We'll be looking at that in more detail when we talk about the exposure triangle. But you notice that there is a little bit of a difference between these two images. Right? If I was to just say, these lips here seem to be now much more smaller than this one. But actually let's go in further and see if we can move further and see what happens. Moving the camera further in. And since I want the same field of view, I would , and I have the same sensors, the same camera, I would have to change the focal length. This is what telephoto lenses are usually very good at. And , now you see a 20 millimeter lens, and this is the image we get. If you notice this face now has a little bit more distortion. It feels more curved, and much more kind of all things, the dimensions seem to be, and the dimensions have shrunk a little bit. And the nose seems a little bit more kind of projected outwards as opposed to in this case the face looks much more facelike. So here there's a lot more distortion. Now, , most photography is, you know, again, what you want to do with it, but just by changing these types of things or parameters on your camera, you can change the look and feel of an image. So traditionally, most portrait lenses that people use are 75 millimeters to 135. I actually prefer to use an 85 millimeter. This is one of my favorite lenses. because it actually gives your portrait, and there are other things that are going on that we'll also look at, as, you know, issues around,. Little bit of more blurring in the background, which is less kind of visible in these ones, but are much nicer in this kind of an image. So in the next lecture, when we look at the exposure triangle, we'll be looking at the relationship, of again details that allow us to kind of also get more information the back and the front of an image.

16 - Summary.srt

So in this lesson, I brought together concepts of focal length, field of view, and sensor size. I discussed the impact focal length has on how an image is formed. Both on the creative sides a little bit, but also on the technical side. As to what you can get in terms of field of view and focal length. Also discussed perspective projection, which allows you to kind of start looking at objects that are further away or closer. And how that is impacted with focal lengths. Lot of basic material, but we will continue to advance further and see more details in the next lesson. For example, we look at exposure triangle. We, we'll actually look at aperture, shutter speed and also more information about the sensor size or sensor values. That has, perhaps gives us more control on what the photography process is all about. , we're looking at this because we want to try to understand how we're going to bring them towards computation. Again, I've listed a couple of books for you to look at if you're interested. I do use a lot of resources on the web, please look at, look them up. More will be available on the website. thanks to Professor Christensen who provided us with some amazing images of the Tech Tower and Georgia Tech campus.

# 03-03 Exposure.txt

01 - Intro.srt

Now we want to actually start looking at the details of how a camera operates. And we'll be specifically be talking about the details of what happens when a camera takes a picture. These are more photographic principles, and all I'm going to be talking about is what's referred to in the photographic circles as the exposure triangle. In an exposure triangle, a photographer controls variety of parameters that lets him or her capture the best image. The three things that are controlled are, the aperture, which is the size of the opening that actually lets the light into the sensor. The shutter speed, which is , the, how fast the shutter opens and closes. And the ISO, which is the sensitivity of the sensor. Most of the time, a photographer attempts to optimize all of these three parameters, that's why this is referred to as an exposure triangle, to get the best picture. What we're interested in is learning these concepts, because we want to be able to figure out how we're going to control these in a computational manner.

02 - Lesson Objectives.srt

So the specific objectives of this lesson are for us to learn, in detail, how does the aperture, the shutter speed, and ISO which is the sensitivity of the film or the sensor, together form a triangle, lets you then control the exposure of the image that you're trying to capture, based on the light that's coming into the camera. Exposure triangle is a well known concept in photography. And the reason we want to cover it here is, we want to be able to learn about the different aspects of a picture or an image that's actually captured on a camera by controlling things like the aperture size, the shutter speed, and the sensitivity. And these are, , the three aspects of the exposure triangle that camera operators use to be able to take the photograph that they're interested in. We will also show you various types of examples today. As to what variations of each one of them. And how it impacts what kind of an image you get.

03 - Recall Focal Length vs Viewpoint.srt

We looked at these images in the last lecture. We started off with one image and then I also showed you another one. This one. In both of these images, the same two subjects just taken from a different viewpoint and, , also different focal length. In this image, I'd taken the picture with a smaller focal length and here, , with a larger focal length. , the two subjects were at different distances. The reason for us to look at these two images was because I'm interested in showing you that by changing the focal length, allows us to move back and still capture the same scene and by changing the viewpoint, we can change the perspective of the scene. For example, here, these two characters look pretty much sitting next to each other even though they are almost a meter and a half away.

04 - Exposure.srt

So let me introduce some terms here. One, first term I want introduce is Exposure. Now if you notice in lenses, we can open the lens to be really small, or much wider. But just having this kind of a change on how big I want the size of the aperture, which is the opening here. I can now vary a lot of different types of things. So now, let's see what impact does aperture have on exposure. If we define aperture by capital H, what we're interested in is that exposure is equal to irradiance times the time that the exposure was open. We can simply write this equation down as H, where the h is the exposure, irradiance, we define by term E, and time by T. , an exposure is irradiance on the sensor, for the amount of time that it was hitting the sensor. That is , the amount of light that’s coming in and hitting the sensor for, and how long defines exposure. So what is irradiance? So again, irradiance, in essence, is the amount of light, the measure of amount of light that falls on the unit area of a sensor per second. And it's denoted by capital E here. So the amount of light that will hit a sensor depends on the opening of the aperture. So in essence it's controlled by the aperture size. Here a small opening. A larger opening, , larger opening will let a lot more amount of light per unit area of the sensor. And this , will have a lot less light going through. And , exposure time is the timing that we allow the light to go through this opening or that opening. That is best designated by the, how long the shutter is kept open. So now we've looked at aperture, which is the opening. And shutter is how long we keep this open. That lets me get to the exposure that I'm am interested in for an image.

05 - Inside a Camera SLR.srt

This is a reminder, again. Let's look inside a camera. We've looked at this image before. This is a film camera with film here, the optics and the lens here, the mirror that moves away when the shutter is released, and that hits the film, and the, the prism that lets you see exactly what are you going to capture. So light goes in this way, and the mirror, depending on when I press the shutter, moves up and then it hits the sensor. And , what we now need to figure out is how big the opening would be here, and also how long do we keep this shutter open to get the light in at the appropriate level to get the picture we want. Here I'm showing you another schematic of a camera. in this instance, it's again an SLR camera. This is the lens assembly. And here you notice, the lens is a bunch of different types of lenses. It could be a mixture of concave and convex lenses with the purpose of kind of focusing the light towards where we want the image sensor to get the best information. So this is the size of the aperture that would let the right kind of light in here. This is the shutter. Opens and closes when we press the shutter release on a camera. Again, it'll be open for a specific time. The aperture, , would be open all the time at different size to let the kind of light in. This is the only device that will control amount of light that hits the sensor. So the shutter will only open and close to get the right kind of light on to the sensor. Here you see an example of the mirror going up. When the mirror moves up, , all the light goes and hits the sensor. And above is where the whole eyepiece pentaprism is, to make sure that we can see exactly what's actually going to hit the sensor after the mirror moves up.

06 - Shutter Speed.srt

Now I'd like to show you a series of examples that best showcase what the shutter speed does to a image that's captured on the sensor. Remember again, the amount of time the sensor is exposed to the light is the way, the best way to kind of see what's going to be registered on the sensor. Here we see an example of an image captured with a shutter speed of one second. , what comes out of it is a whole lot of blurry streaks of the water. Here I'm showing you an image of a waterfall and it's being exposed at different shutter speeds shown here. So it starts of with one second all the way to different shutter speeds of one-third, one-thirtieth, one-two-hundredth. So this is one second, one-third, one-thirtieth, one-two-hundredth, and one-eight-hundredth speed. So again, you can see the different details when this happens. We see, , water, and the water fountain becoming streaking. Or becoming much more crisper. So you can streaks of water, blurry water, and really crisp images there. So the amount of time the sensor is exposed to light is what allows you to have these different effects. Usually these amounts are denoted by fractions of a second. We saw various examples. It could be 1/2000 would be a very fast shutter, or 1/1000. All the way to even how long you want to keep it open. You can actually keep a shutter open for 30 seconds. And sometimes most cameras, in fact, have a version called bulb where if the shutter's open for the exact time that you've actually pressed the shutter and released it. And now you kind of see how the effects of motion blur to streaks are captured by a photographer who wants to control these types of parameters. Another way of looking at the same thing is here I'm showing an example of a waterfall again at a faster shutter speed of 1/125th of a second. when I go to a slower shutter speed, one-tenth of a second you can start seeing a lot more of the streaks. And even more blurriness and streak kind of shows here with one-half of a second. Another example, the same category. Still sharp of this thing, shows each and every petal. Very crisp, increase it and now I start seeing a little bit more. And I really have a much wider or much more open shutter then you'll see everything blurring.

07 - Aperture.srt

So while we looked at shutter speeds, now let's talk about the concept of aperture. Which , again, is nothing else, but the opening that lets in the light into the sensor. So as I said before, aperture is dependent on irradiance. Irradiance on a sensor is the amount of light captured and is proportional to the area of the aperture opening. So , the amount of light that's going to go through the aperture is depending on how big the area is small area, larger area. Traditionally when you just do simple math you can see that the area really depends on pi, f over 2N. Where f is the focal length, and N is the number that we'll talk about in a second. Here, f is the focal length of the lens being used. And , you can start now looking at it as how does one compute the area of something here. And based on this, can we actually now figure out what is the diameter of the aperture. So , an area of a circle is pi r squared, where r would be the radius. This is replacing the radius, by this term here. Usually, in most cameras you'll actually see the aperture number. Or even if you look at any of the details of the metadata of your image, the aperture number would be actually written as a capital N. And usually written as f over N. Let's look at this in a little bit more detail. So the focal length f and aperture number are known. Now one of the interesting things to note is aperture number is given irrespective. It gives the irradiance of what's hitting the sensor irrespective of the lens being used. Because it's the f that's dependent on the lens. And is size of the aperture. So for example, when somebody gives you a lens f 2.0 on a 50 millimeters lens. So f over 2 Aperture for a 50 millimeter lens. Is going to say that the aperture size is 25 millimeters. So for a f over 2 N number, for a 200 millimeter lens, the aperture would be 100 millimeters. So this should start telling us that if the f number is low for a telephoto lens. That means it should be a much bigger, wider lens. See here, I'm just showing you the lens that I had earlier shown you folks. And this is, , a standard telephoto lens. Has a lot of range and goes from 700, sorry 70, to 300 millimeter focal length. It also has the f numbers written here from f 1 4, to 5.6. And, , if I wanted the f number to be low on this lens, I would have had to have, , a much bigger lens. This one is limited by the size of this, and , the size depends exactly on my aperture has to be smaller than this. So if I really want a low aperture number. I would need a thicker lens. And you may have seen sports photographers carry those huge lenses out on the sidelines to be able to capture images.

08 - Aperture f-number.srt

So let's continue to look at aperture. Here I've just schematically shown you f/2.8 to f/16, larger aperture opening, smaller number, smaller aperture opening, larger number. So the f number here is 16, the f number here is 2.8. more, the larger the opening, more the light will enter the sensor, smaller the opening, less the light will enter the sensor, assuming we keep the fixed same amount of shutter time, of the shutter speed. So, just by looking at it, we should be able to discern that, if I double the N, it would reduce the area by 2 times, and therefore, it would reduce the light by 4 times. So let's look at a specific example. We will look at f/2.8 and f/5.6, where the end number has doubled. So going from f/2.8 to f/5.6 will cut the light by four times. And you can actually look at the math and figure this one out on your own as to why that's happening. Partly it has something to do with the fact that you have a square of this value that's going to impact the size of the area. If you want to cut the light by two times, what you need to do is increase the N by the square root of 2. So that starts kind of giving us a sense as to why these numbers are there. you can do the same math between f/4 and f/8, or f/8 and f/16, to be able to get a four times reduction of light.

09 - Aperture Examples.srt

Now what I'd like to do is show you a bunch of examples on how the change of just the aperture itself can change the look and feel of an image. We saw that for changes of shutter speed, where we show motion blur and streaks of light. Let's start off with, completely , a closed aperture. , I'm showing a white. Doesn't mean it should be completely black. There's no light entered. But let's look at, as soon I open the aperture, what kind of images I get. So this is an opening of f over 3.5, so the end number is 3.5. These are my three friends again, and actually hopefully you can see this, and we'll be putting these images up for you to look at more carefully, also online. Much more crisper. You can actually see the beard and stuff. But here things start getting blurry. In fact, my dear friend Einstein is almost out of focus here. This effect is also sometimes known as shallow depth of field. And sometimes it's intentionally done by photographers because they want to focus more on the front person, and not actually pay attention to the people behind. So look at this, was here, , you can't tell much detail of this one. Let's now start changing the aperture values. I've closed the aperture a little bit. Now we are at a 4.0. Might be hard to tell but things have gotten a little bit more in focus at the back. Now you should be able to tell a little bit more difference here. again, now we are 5.6, smaller aperture. Here actually you should be able to see a little bit more detail. We are at f/8. Much smaller aperture. You can actually start seeing his hair. And Mr. Einstein has gotten a little bit more in focus. And you can actually start seeing the vase and all the intricacies of this vase right here. The trees are also getting a little bit more in focus back there. Moving to f/11, increased focus again, smaller aperture. F/14, now here, you should be able to see a lot more detail. Getting much and much crisper, and in fact, this is and this is pretty much the same kinds of crispness and sharpness. There's a little bit more blur, but let's see what we can do with that next. But f/18, very small aperture, much more crisper, here you actually can see the vase in more detail. The tree, the leaves are much clearer. And the whole thing seems to be a little bit more full and focused. And there seems to be no, you know, shallow depth of field kind of stuff back there. Gone, taking it to f/22, much more crisper. Just to help to visualize this, I'm going to just show you three different ones here. 3.5. Very blurry. F/11, less blurry. Tree is in focus. Just switching back for you to see what it was before. Forward, let's go one more. F/22. Twice. And , less light is coming in, for various reasons. Yes, we're keeping the shutter open longer, but that's not the point here. The point is, I'm showing the effects just of this kind of stuff. You see that this is much more in focus. Just looking at it closely again, next to each other, these are the two differences, wider, smaller.

10 - Aperture, Shutter and Focal Length.srt

So , traditionally, photographers used the aperture, shutter, and focal length values of the camera to extensively be able to define how they want to be able to take, and what kind of pictures they want to take. I'm just going to show you a few examples that I found just to kind of show the value of this kind of stuff. Here, shallow depth of field is shown by how the pictures are taken, things are right there are perfectly in focus. These are not that far behind, but they seem to be a little bit out of focus. So this is the concept of shallow depth of field. Motion blur, here you can see, for example, the birds are showing motion because of shutter speed changes. You can see that the wings are blurred. Here you can see the hand is blurred but the text is not. And traditional effects of light streaking at nights and stuff like are again because shutter is kept open a little longer. And when the cars move have this impact. And, , there's a whole tradition of people doing light art. I found this beautiful site that I point here that actually if you want you can look at on your own. It shows a lot of fun things that you can do with just how you can control and how you can play around with light.

11 - ISO Sensitivity.srt

So the third part that we now want to look at is ISO, which is sensitivity. So we've looked at, for example, how we can control the aperture and the shutter speed. But now we also need to also count for the fact that we have another part of the triangle which is, how sensitive is my film? Now, this is dependent on a concept that has, has been done with films a lot before. And when you had actually picked up a film yourself, you may have seen it has numbers like ISO 100. Here I'm just showing you two examples. ISO 100, ISO 1600. So this is an actually an important parameter on how we get the right exposure of an image that we're looking for. Film, a concept that started while film was how sensitive is the film? Now, it really depended. The sensitivity of the film really depended on the chemicals that were put on it. That is, how much light sensitive is the film, and , to achieve, that you would have to put more chemicals. Just by the limitations of how much chemicals you could put on a film grain would result in how much granularity you would have. because you really can't put in a lot of light sensitive chemicals on a film with restrictions on, you know, the density of the film and all that kind of stuff. When we look at film, we'll talk a little bit about that in the next sub-lecture. In case of digital, it comes out to is, what is the sensitivity of the sensor itself. Again, when we look at sensors, we'll discuss this a little bit more carefully. But here, , the bottom line is, the sensor sensitivity is can be now replicated by a number equivalent to the ISO. Just as a important point to note, the relationship between different ISO values is linear. A 200 ISO needs half the light of a 100 ISO. So this one actually would take lots of nature lights, but 1600 would be something I can use indoor without a lot of light. Just to show you the details of this kind of stuff. Here you can see the images of the ISO zoomed in, ISO 100 zoomed in. And this is ISO 1,600 zoomed in. And here you might be able to see a lot of noise in the two different colors of yellow and green. And the boundaries should also be a little bit blurred, because again, more chemicals have been put into this film, which means that there's mixing going on a little bit here. These find is crisp. , ISO 1600 would be used something where there's low light conditions. ISO 100 would be used when it's nice and sunny. But, again, photographers play around with this depending on what their needs are.

12 - Exposure Triangle Examples.srt

Now what I'd like to do is just, to kind of help us think through this. I'm going to show you a whole lot of examples to cover the space of what it means to go from different values of the exposure triangle. Let's take f number 5.6. And I will start off with a shutter speed of one over ten. This is the same scene. And , because of the fact that the apertures open for a while. And I'm also keeping the shutter speed. There is a lot of light entering the sensor. This image might be referred to as over-exposed. Same aperture value. I've now gotten the shutter to work at one one-twentieth, twice the speed of this. Twice as fast. here you can start seeing that image is less over-exposed then this one. I'm going down again, doubling the shutter speed. Doubling it more. And now you can start seeing that this image actually seems to have less overexposure than any one of them. But I keep on going there to one one-sixtieth. Same aperture value. Now it might start getting darker. More darker. Even more darker and now we are getting into the range of underexposed. For those of you interested, we may actually now start thinking about what would histograms of these images would look like. Based on the lecture we looked at before. Darker image, nothing like this, this would be. Overexposed, underexposed, extremely underexposed because now we really can't see any details. Let's now actually move up and down these axes. So now, I'm actually going to take this speed, one over one-hundredth of a second. And then I'm going to go in and actually give it an f number of 4.0. F 5.0. This was 5.6. 7.1. 9.0. 13.0. And 20. Again, you can see what the impact was. It was, these the images are much smaller. So you can't details of the depth of field kinds of stuff here. But this seems to be a nicely exposed image. A little over-exposed, but fine. But as you go on increasing the aperture size, oh sorry, decreasing the aperture size, we notice that now we're getting a lot less light in there. So here, we went this direction and we make the shutter speed faster. And here we went down this way and made the aperture smaller. These two are underexposed. These two are overexposed. We can imagine actually filling this complete matrix out, but that would have been a too crowded of a slide. I just wanted to show you the impact of all of this.

13 - Aperture and Shutter.srt

Let's look at the same example, starting off with this one. But this one I'll actually be doing comparisons on ISO versus aperture and shutters. Filled out the same thing as before. We're now going from increasing the speed from one-tenth to 2500. So we increase the shutter speed. Now, let's actually look at different ISO values. One hundred, 200 is the ISO. This was 400, 800, 1600, 3200, 6400. Now again you notice that as I got to 6400, it got much more brighter. And this one is underexposed at 100. So here somewhere in this range would have been a good choice of a ISO value, with respect to what we want. , we could fill this out also. So this is, ISO is increasing and this is shutter speed. So you see the effect, photographers do the best they can to figure out what first scene that they want a different effect. That's why they also carry a light meter sometimes, to kind of get a sense what would be the best combination of the shutter speed, the aperture value and ISO they could use for the for the purpose of the pictures they are taking.

14 - Exposure Triangle.srt

So this, is my exposure triangle, shutter speed, aperture, ISO. By changing the aperture, we can actually can impact the depth of field, increasing ISO, we get more grain, and decreasing the shutter speed, we can get motion blur. So, it's a combination of these three things, that allows you to explore how we can actually, best take the pictures we want. So, aperture opening, is one of the parameters we want to control, shutter speed, that is how long do I want to keep the shutter open to get the amount of light in, to my sensor is something I want to do, and once it hits the sensor, I want to actually know more about the sensitivity of the film, or, the digital sensor that we're using. photographers optimize these based on their experience to get a desired exposure. On the website for this class, I'm going to put a small applet that actually people at Stanford built, that will let you explore, the variability of the these three to be able to get the best exposure in a synthetic kind of a play around mode.

15 - Recap Exposure.srt

So just to keep recap, we came up with the whole definition of how the exposure triangle works, reminding that we are interested in this simple math, where we want to be able to get the irradiance times the time, and use that to generate the exposure. Irradiance is controlled by the aperture opening, and by lowering the f stop doubles the exposure as the aperture is open more. And by lowering two f stop doubles depth of field. So this is kind of the impact we're looking at throughout. That by playing around with how the f stops change we can actually impact the size of the aperture opening that impacts what the image would look like. And sometimes by playing around with this, we can actually see impacts on things like depth of field and similarly for shutters and stuff like that. I've showed you many examples in this lecture for you to kind of now take it in and play around with your own cameras. , not all cameras allow you to have this kind of control. These days even mobile cell phone cameras do have some code available for you to play around and change the aperture and shutter speeds and stuff like that. And we'll try to make them available, but feel free to discuss them amongst yourselves and find other examples of this kind of stuff. , most of the examples I showed you were generated using high-end SLR cameras. In the previous slide I talked about the change of aperture and now exposure time controlled by shutter speed. When you double this the time, the shutters open, well your exposure is doubled also. When you double T well it might actually also double motion blur. And , when you're take ISO, you're doubling ISO needs half the light to be able to generate a similar type of an image.

16 - Summary.srt

So in summary, I presented how a camera operates. Brought together the concepts of aperture opening, shutter speed, and film sensitivity, for optimized photographic exposure. And we discussed the impact of exposure triangle and it combines various aspects of photography. Now, this part of the lecture was much more driven by the basic concept of photography. Do remember, we are interested in being able to kind of generate images and also understand the computational aspect of how we can control various sets of parameters which would then allow us to do computational photography.

# 03-04 Sensor.txt

01 - Intro.srt

So far we've been looking at cameras but we've been actually looking at the details of the optics, that is how light enters through the optics. And at the back, at the focal lens of the camera where the image is formed. Now let's talk about how to save the image. In this kind of camera, a film camera, a film is placed here and it's the chemical process of the film that takes the light and preserves it. By changing the chemistry on the film and that is stored to create prints. And this lesson will talk about the composition of the film itself. But also, since we are actually moving to the world of digital photography we'll talk about the composition. And how something like a CCD, a charge couple device, is used to save information in an electronic form or electrical energy form, from light. We'll also discuss concepts of bare filtering, that allow us to talk about different colors. And also a CMOS sensor, which is the one that's widely in use for all the cameras that we actually currently use these days.

02 - Lesson Objectives.srt

The specific lesson objectives for this lesson are, we will look at the photographic process for both digital and film cameras. I will describe to you, the eight different layers that constitute a color film. I will also describe the five layers of a generic CCD that's found in most common cameras. We will discuss the differences between a charged couple device, a CCD and a CMOS sensor. And I will discuss some of the benefits of using a camera raw format. This will connect back to the lecture we had on different types of image formats. Remember, it's the image format that captures the pixels. And it's the pixel values, the intensities, which we want to be doing most of our processing on. We will just take that up a notch and talk about a camera raw format a bit today.

03 - Recall Inside a Camera.srt

Now recall from the last lecture, we had looked at the insides of a camera. The lens assembly. Light goes in here. The shutter opens. Light hits the sensor, which could be the film or a digital sensor. Only when the shutter is pressed, that's when the mirror pops up. Otherwise the eye can see through this pentaprism, where what is the scene here. So, in essence, what happens, is, when somebody presses the shutter, this op this mirror moves up, the shutter opens, and the light hits the sensor. So, specifically now, we are going to start looking at is, what is this sensor? And what constitutes a sensor like this?

04 - Film vs Digital.srt

So as noted previously, there are two types of primary sensors we would be interested in looking at, film versus digital. There's a long-term debate that has been going on between film versus digital, use of either film or digital in cameras. I'm not going to get into that religious debate, to me that debate has, similar feeling, as a debate between digital music or through vinyl recordings. We will look at both of them. Film usually comes in canisters like this. You've seen different formats of this. For digital, what we need is some sort of a storage media. Here I'm just showing you some generic forms of, flash disk that are not commonly used in, cameras. I'm going to make a claim that film and digital cameras are practically the same. For example, if I showed you this camera, this large camera here, again, a digit, this is an SLR camera. Or, I showed you this camera. Both of them look exactly the same. Both of them have you know, the shutter. They have lenses. And this is a single lens reflex camera, which means if I was to open this up, and put this right here, you can see it has a mirror, and if the mirror pops up, there should be a sensor behind it. And, , the shutter controls how the shutter would open. And the mirror is moved up and, , the optics of the lens here are also the same. So in essence, hopefully you'll agree with me, that both cameras have pretty much the same form factor. , you may be arguing with me that, oh, things are really different now. For example let's say, you know, I bring in a cell phone which has a camera here. This is a complete change in thinking about how photography's done, or let's look at, for example, an instamatic kind of camera here. This, , is a digital one. But if you look at all instamatic cameras, they pretty much have the same form factor lens sensor, which could be film, and in this case, , if you look at it, the film is not there. And, , we have is a small disc. So this whole thing suggests that cameras are still alike, in their, you know, kind of the whole form factor that we've seen them for a long time. They've just gotten smaller and also have a lot more different types of functionalities. I will admit that with the innovations there have been significant improvements in actuators and lenses. Lens quality, , in the old days, was very good and is still very good. But the actuators, the motors that are come in, are actually much better than they ever used to be. Furthermore, the sensors and stuff that we use in cameras now may not have the same kinds of fidelity as film. But , instant gratification. We get pictures the moment we take them, and we see them right away. The bottom line with all cameras, in general, is there is no difference in how light is transported through the lens, but it's how it's trapped and preserved. So the whole process of how it's sensed, trapped, and preserved, either in film or in digital form, is what's changed. For film, , there's a chemical process. For digital, , it's the electronic process of how light energy is converted to electronic signals, which are then saved as digital signals and then are converted into pixels. In case of film, it's a chemical process that converts light energy and preserves it in a format that can be saved on film. I, I had earlier already shown you this camera that I have popped open. And it had a film in it, like right here. I navel, I know it's an exposed film. But , that's what we have in old traditional cameras, a thin frame here. I'm not going to take it out right now. So here you just see the film. I've, , exposed it. But that's in a canister and, , it goes through as soon as I take a shot. I forward the film, and I expose the next one. This is, , the film taken out of the camera. This is exposed but not developed film. After a film is exposed in the camera, you take it to a darkroom, and then other chemical processes are added to develop to be able to get the images out in the negative format that we have seen before. I'm sure all of you have played around with negatives and have seen them in person. So the bottom line is, we have a canister, we have digital format in form of a disk, right now.

05 - How Film Converts Light to Image.srt

Now I do want us to look at the film a little bit more carefully. a film is the whole process of how a chemical process is used to convert light. And the, it is the energy of the light itself that changes the chemicals. And then we have to frill, figure out how to preserve that changed in chemical process to then be able to store light into film. Here I have just laid out a, a simple 35 millimeter film, pretty much like the same one that I'm showing here and I kind of now unwrapped it. In the schematic to show different parts of a film. So in this instance, light is coming in. And if you notice any one in this one, the light was coming in from that side, and that was the one that was pointing to the optics, the lens of the camera. The light comes in and, , there are many different layers. Let's just go quickly over them and see what each one of them does. First layer is the protective layer. We need to be able to protect all of these light sensitive layers that are actually going to exist. , in most types of cameras and films like this we also put in a UV filter. The UV filter is there to just make sure that the right kinds of light, and not damaging lights, are hitting the different types of sensors. Ignore the colors of these types of things. Each one of them has a different feature. So the first one that the light hits is a blue light layer, then a yellow filter is added to kind of get the yellow light out, green light layer, red light layer. So, again, we have green, red, and blue lights, then there is an adhesive which is put in there, so these are all very thin layers. And, , thickest one is the film base, which gives it the structure and keeps it stable. So in essence, all of this is tied up in something this thin. So it's important to note that not only in the, in the present days, people can pack things up in small things, a lot of this packed up into a very small thing there. So in essence, a film is a sheet of plastic. And that's what I've shown you so far. It's coated with an emulsion that is full of light sensitive, silver halite salts, which have been bonded by a gelatin. So in essence there are silver halite salts inside the layers. These are the light sensitive ones that actually store the light changes. And then there are variable crystals all over the film also, that determine the sensitivity, the contrast, and the resolution of the film. This is where some issues like ISO come in, because this is where all of that practicability of how sensitive that film is comes in.

06 - Reaction Between Light and Chemicals.srt

Just another way of looking at the same concept. Here this is the film thing. Light comes in from down here. There is protective coating. The emulsion is where most of the details of the light silver halides and everything else is. The adhesive, and there's the base, and that's covered up to prevent it from getting damaged across the board. Sounds simple, and , it's a very interesting process. You know, full credit to the people who innovated this whole idea ways back when. And some of the earliest cameras, as we talked about before, were developed in the mid 1800's. Purely a chemical process, and , it took them a while to even figure out how to preserve and save and then print out, develop and print out, from these chemical processes.

07 - Digital Sensors Convert Light to Data.srt

Now let's look at the process of how light is converted to data. And look at the digital process in parallel to what we just looked at which was the process for doing it for films. The foundational device, foundational technology if you will, for converting light to data is a CCD, right, Charge-Coupled Device. It's an electronic device that converts electrical charges into a digital value. So in essence, a pixel on a output that comes out from the charge coupled device, or a CCD, is , are different capacitors. Which can work and store incoming photons as electron charges. Photons are the ones that have photoelectric value. And once they have these capacitors these are converted into electron charges. And it's this electron charge which is stored and given a digital value. It's important to note here that Dr. Boyle and Dr. Smith in 1969. While they were working at Bell Labs invented the concept of a CCD, a Charge-Coupled Device. And , they were awarded with the Nobel prize in 2009. And actually you know, being person who works in the photography field this was an amazing piece of technology that they have developed. >From 1969 to 2009 and it's one of the most common pieces of technology in our homes, our businesses. And everybody has one nowadays is a camera that is somehow the other related to this concept of charge coupled device.

08 - Charge-Coupled Device.srt

So now let's look at the whole schematic of a simple charge couple device. Here I'm showing an animation of each and every layer of a charge couple device. We are going to now peel this back out as an onion, and look at each one of them separately. But , point is, light is coming down this way and hitting a variety of layers. And then, , it hits the bottom layer, where actually the current is stored. Let's again, look at this by peeling the onion layer by layer. So the first layer has a little bit of an optical flavor. It's what is referred to as a micro lens. What in essence it does, it captures the light and directs it towards the light sensitive areas beneath it. So when light hits these types of things, it focuses it down to the next layer so then nothing is lost. The next layer is referred to as the hot mirror. This is where some of the simple filtering is done. So in the case of the film, this saw the example of a UV filter. Well this is somewhat similar. it lets visible light pass, but reflects light in the invisible spectrum. And doesn't pre, it prevents it from hitting the sensors below. So this simple sort of filtering and prevents the unnecessary light to hit. , different types of cameras would do different things. So for example, if you wanted to build a UV camera, this would not be filtering out the UV light. In fact it would be amplifying it. This is what's referred to as a color filter. Remember light has lots of constituent colors. Most of the time we try to represent them or at least capture them in RGB format. And that's what we're going to try to understand how to do here. So in essence, the photo diodes that are below it are actually color blind. They want to capture the intensity. So this is where we kind of start now looking at and separating out the colors. This is usually referred to as a Bayer Array. And we'll talk about Bayer Arrays in a bit. what it does, it takes a color and separates the light into red, green, and blue. And, once it's been ordered into red, green, and blue it hits the photo diodes that are below which are then measured the intensity of it. , this pattern is unique and knowing this pattern is essential because it'll tell us how to be able to convert the value from, in this case, two greens, red and a blue to figure out what would the value at this point of the image. So as noted, these are the photo diodes. This is exactly where the light energy that now actually is hitting each one of these, is converted to electrons and carries a strong negative charge. The final layer is the Well, or sometimes referred to as the depletion layer. This is where electrons are collected and usually there is a processor that charges the photo diodes with a positive charge, and then all the negative charges are stored and collected on this layer. So in essence, when you're pressing a shutter, what you're doing is opening the shutter. And also giving a charge to this device so that can it capture all of these stuff. In case of a chemical film, all you need to do is open the shutter and that's when exactly the storage happens. And , when the storage happens, you move the film across. In this one, all of the stored, captured off, and we'll talk about that in a bit, too. So in essence, we peeled this whole onion and we saw all the details of how light goes all the way and then ends up as a stored electrical set of values which are then converted to pixels.

09 - Bayer Filter.srt

Let's talk a little bit about the Bayer filter on the sensor. So in essence, a Bayer filter is nothing else but different types of color filters that are above the photo diodes. And most of the time you'll actually see two greens and a blue and a red, so this would be the four region here, that represents one pixel. Four another here, that represents another pixel, four here and so on. , depending on the convention of the Bayer filter used, sometimes you might have two reds and a green and a, a blue, and also two blues and a green and a red. Depends again on it, but knowing exactly what is the Bayer filter is essential because we need to know this to have to compute and then, come up with a final image. what happens with the Bayer filter is each one of them lets only a specific light through. So when the light is hitting the red, , the blue and green are prevented from going through. Red goes through. Same thing with green. Just the green goes through. Same thing with blue, only the bo, blue goes through. So, in essence, we create three different patterns. the photo diodes then store the intensity for each and every one of them. But, this coding scheme, pretty much lets us know which one of the bins is actually blue, green, or red. Then it's the combination of these three channels that gives us the final image, and this is the one that we need to then decode and save for our final things. One of the questions that's important is, how is the raw input in a Bayer mosaic format converted to an image? What's the process to be able to then get the RGB color values, that is, the RGB as we have discussed when we talked about the image processing pipeline, the three channels separated out?

10 - Convert Bayer to RGB.srt

Let’s try to understand that through a simple quiz. I have a green channel 180, green channel 170, and then red 200, and blue 153. , what I want to do is, I want to generate a simple pixel that represents all of this, at the final. What would be the values of R, G, and B? Please enter the answers in these three boxes.

11 - Convert Bayer to RGB.srt

, this was a very simple answer. The red would all, just be the red itself, and the blue would be the blue itself. But the green should just be the average of the two greens here. So , that would be 175. So, as I said before, these could actually also be a different bayer pattern where there could be two blues, and this would be a green and a red. depending on the knowledge of this kind of a lattice structure, we would determine what would be the values of R, G and B, at this point.

12 - Bayer to RGB Demosaicing.srt

This is referred to as the Bayer to RGB demosaicing process. In this case we just take a four by four subset and use that simple math and average to start to create an RGB triple. And that, , is the one that's to create those layers. There's a lot of literature on this kind of demosaicing. I encourage all of you to look it up, and kind of try to understand the whole process as needed.

13 - Reconstructing Image from Bayer Data.srt

Now let's look at the next step in converting light to data. The actual sensor information, with Bayer data, will have something like this. We saw some of that in the previous slide. The red is chunked up into small reds here, because that's the red rows. The green, , also have this Bayer pattern. This is, again, output of a Bayer filter. We want to demosaic this to get our real image. Going through the process that I talked about, with some interpolation, we can construct an image like this. Here you see a little bit of an aliasing, because again, we have intentionally showcased the kind of artifacts of doing this. But simple cleaning and interpolation that's done, this image would be made to look perfect.

14 - CCD vs CMOS Sensors.srt

Now let's look at another whole concept. And that is the use of a CMOS sensor. CMOS stands for Complimentary Metal Oxide Semiconductor. again, has a lot of things that it has learned from CCD. But a CMOS, , is a little cheaper. And it's something which actually you will be, you're using all the time because most of the cameras, the small cameras that you see and in fact, in the high end cameras, now all move to using CMOS sensors as opposed to CCDs. Here are a few interesting differences between CCDs verses CMOS sensors. One thing to note is, , as we talked about before, there are these Bayer patterns. But ben, beneath the Bayer patterns are these photo diodes. We can refer to all of them as photosites. This is where, , information is captured and converted into light intensities and colors. So these regions are the photosites for either one of them. , any one of these regions is a photosite. And we want to be able to capture information from these photosites. So the big difference is, the photosites in a CCD are passive and actually do no work whatsoever. As soon as something is captured, information is then moved over, and there is an amplifier that's used to kind of take the exact values and amplify it to a scale that can be measured by the storage device that comes in. So I'm just showing an example of how things are copied over, one row at a time, and then saved. Photosites in CMOS actually have an amplifier right there. And actually they can do local processing. So the readout at each and every one of them is local as opposed to in this case, not really global, but actually it's done after all the things have been stored. Here, every readout is done at the local sensor itself. So in essence, there is a small local amplifier at each and every one of the photosites. So each and every point here, every photosite has its own amplifier. So, , that allows them to do local processing. One thing to note here is this is one of the traditional problems. If you play around with the video on your hand held camera or a cell phone camera, you see something called a rolling shutter artifact. We'll actually, in one of the applications, talk about how we can get a rolling shutter. It's something which we've done a lot of research on. But one of the reasons for rolling shutter is because of the readout that's happening at each and every photosite, if the camera is moving faster than the readout is, you will see some sort of rolling artifacts, some non-rigidity in a scene, or some bending of the scene. That is just because when this thing is read, by the time this is read, the scene may have changed, and that is an artifact that's common in, especially lower end CMOS sensors. Higher end ones have a much faster readout rate and can do much better.

15 - Camera RAW File Format.srt

Now, in the past lectures, we have talked about file formats. We talked about file formats being nothing else but a data structure that would actually store the intensities of each and every pixel in an image. But in some instances, someone might say, that's a lot of missing information, why don't we get a lot of information that's raw to the sensor and actually just save that. And that's behind the whole concept of a Camera RAW file format. , what you do in a camera RAW file format, is , you store the minimally processed data from each and every part of the sensor. Each and every point that's measured light, you save the original or as close to the original raw data from there. The images in a camera RAW format are encoded in a device-dependent color space. Sometimes, , it's a propriety color space. But each camera has a specific form of a color space. And, what it does is, image is stored in that and after you save it the camera manufacturers do provide information on how to take that color space and make it much more readable on a computer. More importantly, it also captures the radiometric characteristics of the scene. Radiometric meaning that the light and the intensities of the colors from the scene are captured in a raw format rather than manipulated in any way whatsoever. Many of you who do advanced photography know that in some cameras, there is a lot of calibration tools that allow you to control the light and also the colors very specific to the scene, sometimes by doing white balancing or different types of color contrast and stuff like that. One of the beautiful things about camera RAW is, since it's capturing the raw radiometric characteristics of the scene, you can do all of that after the fact. In essence what it does is lets you get the image from the camera's sensor data. So the whole concept of a Camera RAW file format is, it's to kind of replicate the whole photographic negative. One of the things that went away when we started going from film to digital images is, while we had the instant gratification of seeing the image as soon as it was captured, we got it processed, it was not something we could manipulate anymore. One of the beautiful things about doing dark room photography was always that once the film was exposed, when you were sitting in a dark room, you could do a variety of different ways of exposing that film to give you different contrast levels. Well, that now is possible through camera RAW format, because once you have the camera format with things like the radiometric characteristics and stuff like that, you could actually do a lot more. So in essence, one of the advantages of a camera RAW file format is that they're minimally processed, device dependent, and therefore actually, you know, you know more about how the perfect response for that specific camera's devices. Oh. This, so it's device-dependent, so , the colorspace is really known by the manufacturers, and therefore they allow you to read out the best values from it. Radiometrics, all the color values are well defined and captured raw. And , this represents an equivalence class to the form, photographic negative, lets you control things after the fact. And it allows you to do things like changes in dynamic range, color, and most of the information of the captured image is available to you.

16 - Summary.srt

So, in this final sub-mar, sub-lecture about cameras, I just wanted to showcase to you the importance of different types of sensors. We discussed the photographic process that was applied to both digital and film capture. Also discussed how sensors work in a camera. We looked at different types of color filters. Talked about their filtering. Discuss in brief CCD and CMOS sensors. And also discussed a little bit about the camera raw format. The goal, , of all of this was for us to now learn as much as we could, about what a camera is, and how rays of light are converted into pixels. We have looked at how to do pixel-based processing before. We know how light is captured, in this instance. So now we will take a little bit more deeper look at doing computational photography. So in the next class we will actually get into doing some computational photography. We'll start with learning a lot about things like blending and fading. Eventually we'll come to look at things like oh, how can we actually build panoramas and high-dynamic range imaging, some of the well known examples of computational photography. , I'll be exploring and actually also showcasing to you a wide variety of other sets of examples in the same way. Again, just wanted to list a few examples of books that you could look at that give you more details about both cameras, the processing pipeline, and also, you know, how photography really works. While this class is not really about photography, these are foundational things that we will actually be playing around with a lot. Again, look for additional details on the website for the class. And remember, always have fun computing with photographs.

# 04-01 Fourier Transform.txt

01 - Intro.srt

So, we continue to get deeper into understanding what images are. One of the things we also have to start thinking of, the images are nothing else but samples of different intensities. These samples can, , be represented in various forms of frequency spectra. So in this lecture, I'm going to talk about how we're going to take an image and start looking at it, and of various forms of frequency information that could be extracted out of it. And those frequency information can then be used to analyze images differently.

02 - Lesson Objectives.srt

The specific lesson objectives for this lesson are, we'll pick up on the foundations of sines and cosines, and start thinking about how we can use that to construct a signal. Using that concept, I will introduce the concept of a Fourier Transform, and that will allow us to start looking at the frequency domain of a signal. We will start off with 1-D signals, but then will actually move towards playing around with 2-D images. To help us do the calculations, we'll then , spend time looking at the specific properties of convolution as they relate to Fourier Transform.

03 - Recall Images and Camera.srt

Before we go on, let's do a simple and quick review of what we have looked at so far. We have looked at the concept of images and cameras as they form the foundations of computational photography. We have discussed at length how rays of light go through a camera, and then register on the sensor to be able to give us an image. We've looked at how aperture and all of the different parameters or different controllable quantities within a camera are used to generate an optimal image. Then we have studied how a sensor or a film, converts the light information into an image. Which then we have discussed how we can actually convert to digital information in case of a sensor. And then , we have learned about how we can process an image to be able to take something like this and find interesting, and information theoretic features out of it that we can use for a variety of things.

04 - Reconstructing a Signal.srt

Now let's look at a one-d signal and start thinking about how would we actually reconstruct a signal based on the information that's available to us. Here is a signal that's in one dimension and it's a repeating impulse function, which means that this function is zero and all of a sudden there's a peak. It comes down. Peak. Comes down. Peak. Comes down. So, in the four times it repeats. So this is, you know, four different periods. Whenever it hits point one, two, three, and four, it repeats and hits a magnitude. A one. So that's why it's referred to as a repeating impulse function. What we're interested in is, if this is my target signal, how would I go ahead and reconstruct it if I knew anything about the signal? To help us do this, let's create a building block. The simple building block we want to actually use here is this function. Which is a cosine where A is the amplitude and omega here is a frequency. So, just by varying the omega and the times of omega, for a specific A, I want to be able to generate a function, f, and I want to be able to see if I can actually use this to start creating a recreating a signal like this. So we're interested in forces using a function like this, where we can vary the n, omega, the frequency, time is the variable that goes on this axis. And , you want to be able to go, in this case, an amplitude of A which is 1 and generate a signal like this. Any thoughts on how we would do this? Well let's get into it. So if you just remember some of your earlier math concept that you may have studied in past what we want to do is now create a repeating signal that goes and has four periods. And it goes from, in this case, 1 to minus 1, back to 1, minus 1, back to 1 and , it's repeating. And , this is an output of this cosine function, where t varies from 0 to 4 here, and we're changing omega. Now , if you know any things about sines and cosines, you know that if this term here is going to be a specific value, the value of this one will be 1. So that means in a repeating circle like this, we're changing the angles. And sines and cosines of these angles have been computed and plotted on this thing. So let's look at a variety of these things. F1 was when we just replaced it and we have this value. Now in this case I'd claim n was 1, what happens if I make the n 2. So replacing the f n value to be 2. With the same omega and t. We now are getting twice as many repetitions for the same period. So now, , this is higher frequency information. We're getting more signals, and peaks and signals coming in than we did before. Here is now I've gone for the n value of 3, and repeated the signal again, so the more frequencies are coming in. Here f4. Again, higher frequency. So in essence what we've done is we've gone from low frequency information to high frequency information here, now let's look at all of them together. This is the f1, f2, f3, f4. before we go on further, let's actually introduce a way for you to play around with the similar kind of concept on your own, on the browser, or on MatLab, or in OpenCV.

05 - Generating Cosines in Octave.srt

So on your browser you should have a window which allows you to do simple calculations. Here we are going to play around with Octave. Which is, again, very similar to what you will find and be able to interact with on MATLAB also. Let's start off giving it a few sets of things. And then, we'll actually see what we can actually do to run different types of things here. Here the goal for us is to play around with different cosines. We're going to look for four periods, n is equal to 4. And I'm going to create a variable vector here, t, which is going to vary from 0 to 4. But also, what we're going to do is, since we're going to play around with sines and cosines, we're going to look for degrees. And here I'm just going to use 90. And here, we're just giving it an amplitude of two. All right, just to get us started. This was f1, where we have amplitude times cosine. Goes from our n in this case is or at least the num, the a, n value that we had was one. , times 2 pi. And , multiplied by t. That was the equation we looked at. Let's just plot this and see what happens. , here you get a simple response to the cosine function. Remember our amplitude this time around as opposed to in the previous case that I've showed you on the slide as two. It goes from minus two to two. Starts with a value of two. That's the max amplitude. At half comes to negative two and it just varies from two to minus two and we have four different cycles. looking at this plot you can see that we can actually now generate other plots like this. So for example, this would be an f2. Again, the only thing I've changed is the multiplier here. Is now replaced by 2. So this is a simple way of now looking at it. We can plot this. And actually, let's just go ahead and actually generating more of these f values from f1 to f4. So, here, , I've just added f3. And this here is f4. And , we can plot all of them. So let's go ahead and run this and see if we can actually plot all four of these things. So this is f1, amplitude of two, f2, amplitude of two, and if you notice the frequency's increasing if we hit f3 and then f4. So we have all four of them. So now the question for you all is, what happens if we just take all of the f1, f2, f3, f4 and add them together. Here is just me now plotting for the t value, all of the different f1, f2, f3, and f4. This is the amplitude we get. The amplitude has gone up, but if you notice an interesting pattern is showing up. There is an impulse function, repeats again here, repeats again here. There is, , some changes in values between these impulse functions. But an interesting trend is visible. Well, let's take that idea to the next level.

06 - Reconstructing a Signal.srt

So, the exercise we did in on let us take different f values and add them together. These were the four f values that we looked at. So, now lets look at how we can start adding some of these together. Here was my f1, another f2, f3, f4. I've just showed them on top of each other. If we start adding them what happens? Well, then what we have to come up with is an equation like this. Which says that f target here is a summation from 1 to n, where n is how many of them we want to add. At the amplitude and, again, the same function we looked at. So, now, let's actually start adding them one by one. If I just add the first two, f1 and f2, this is the signal profile I get. If I add 3, here I'm keeping the amplitude bounces between 0 and 1, I would start seeing this pattern, f4, this pattern. You saw that in the exercise we just did. What happens if I now start going for a lot more. So if N goes from not just 4. So initially we looked at N was equal to 4, but we really want to go a lot more, right? So what happens if I now go for N is equal to 10. N is equal to 50. Now, you would actually kind of see that, in some instances, this signal and this signal are approximately similar. They have a peak, add 1, 2, 3, 4, 0. But, it's kind of flat between. , it still has a lot of signal here. But overall it's actually trying to mimic the signal. So, in the limit, of N was a much, much larger number. You would actually see this to be true, that this would actually become a signal that repeats and has the same characteristics as that. That's what we want to learn about how to do. In essence, by just taking sines and cosines, and by mixing them up together correctly we can actually recreate a signal that we're interested in. in this case I showed a very simple signal, but we want to be able to do this for more complicated ones too. But more importantly, it gives us another way of looking at a signal.

07 - Fourier Transform.srt

So, this whole concept of how we can actually take a signal, and represent it in a different way, was something proposed by a famous mathematician known as Fourier, and , this whole concept is also referred to as a Fourier transform. So, in the previous slide, when we did the work on trying to reconstruct a signal, have to remember that those impulse functions were repeating themselves after a fixed period, so that's why that was a periodic function. And we looked at them and concluded that a weighted sum of sines and cosines of different frequencies, because we changed the frequencies, allowed us to reconstruct that periodic function. So in essence, what that says is that any periodic function can be rewritten as a weighted sum of sines and cosines of different frequencies. So in essence, what a Fourier transform does, is it transforms a mathematical function of time, into a new function, whose argument is frequencies, not time, and this frequency usually has units of cycles per second repetition, and it's sometimes referred to as hertz, and sometimes also kind of looked at as radians per second. So it's no longer time steps, but is measured in, how the frequency or how many times it's repeating itself. And the new function is a frequency spectrum, of the function f, we'll show what this means in a few seconds too. Now, one thing to note is this operation, that goes from a function to a Fourier transform has to be a reversible operation, and for every frequency omega that goes from 0 to infinity, the Fourier transform holds the amplitude A, and the phase, of the sine function, again, we will look at this carefully, but in essence, we need to be able to both look at the amplitude and the phase variations of the sine function, cause that's something we want to be able to use to represent that signal. So this is the equation that we want to be able to model, frequency, phase, and this is the Fourier transform, A is the amplitude.

08 - Frequency Domain of a Signal.srt

Let's look at this concept of the frequency domain of a signal in a different way. Let's create this three-dimensional axis. A is amplitude, t is time. This is exactly how we've been looking at signals. But, now, actually, let's look at also the third dimension here, which relates these things, but now we're measuring the frequency. So the first one, f1, we looked at, was our first frequency, right? Which had this amplitude and this variations for, again, a period of 4. f2 is, again, A and t are the same, but what we did was we added and multiplied the frequency by twice, to be able to have the same, twice the number of cycles in the same period. So f2 was a step up in the frequency domain, here. And similarly, f3 was another step in the frequency domain. If we were to look at the same plot in a and T, , it went from here to there to this. This is how we actually looked at all of those four different, f1, f2, f3 and f4, in the previous slides. So in essence, what we're trying to do is now look at the signal in these two axes, amplitude versus frequency, not just amplitude and time. In amplitude versus frequency, we have three different plots here, we want to look at it from this direction. For, in the time domain, we have been looking at it in this direction, and we have seen these plots before. But now we want to be able to look at these things from that side in this axis. So one big question is, how many samples do we want in different frequency domains? Here we just showed three. As I showed you before, if you went from N equals to 3 to N equals to 10 to N equals to 50, we were actually much closer to the final signal that we wanted. And , as N grew larger, we were even much closer. So, , one big question remains is, how many samples do we want of different frequencies? Another part of it is, in the frequency domain, as we go down this axis, what each and every one of the signals contributes to the overall signal, and how we can control it. And finally, the most important part, why we want to do this kind of analysis in the frequency domain is, for example here, if I'm looking at f1, I'm only looking at a very coarse signal. When I'm looking at f3, I'm looking at A comparatively from f1 a more finer signal. So if you go down this axis where N is increasing, we , here we're going to be looking at a very coarse signal. And here we're going to be looking at a very fine signal. So if you notice, one important part of it is, now we can do analysis differently for coarse signal and differently for fine signals. And that's an important part of what we want to do,

09 - Time Frequency and Frequency Spectra.srt

Now let's look at a few examples of how time, frequency, and frequency spectra are combined together. Let's imagine this is a signal I want to generate. This signal was generated by simply doing an addition of two different signals. This signal here, again, repeating. And the next one was this one. Let's look at them mathematically. So this output target signal was a sine. Twice of the frequency stuff here. And then one-third of the sine of twice what the frequency is also much higher, three times. So in essence what you look at is this signal times one-third of this signal is giving me this signal. How we plot this in the frequency domain? So again, now we're looking at amplitude and the frequency. The variables here are f1 is f itself, f2 is twice f, f3 is 3 times f. Remember that's, this is what three times f is. So first value, , is 1 amplitude entirely of f itself and then 3 times one-third signal. So this is, that's why this is 0.13. And this was about 1. So this is the frequency spectra of this signal looks like.

10 - Frequency Spectra.srt

Using the same analysis now, we can actually even create signals that have characteristics like a box filter. I will leave this up to you folks to kind of play around with. But in essence, taking infinite or a large number of samples where and could be as large as infinity starting with one, we'd actually now take this sine function and using this we can actually generate an approximation to a box filter. For that specific case, the amplitude and the frequency axis which show this characteristic. Starting at one and as we go down one the whole spectrum will reduce as we go from again, you know, different frequencies. So in essence, in the limit, as I go for large number of samples, we would be able to generate a filter response that would actually have that box filter representation. This is shown by, as I said, you know, in the limit. As we increase N the frequency increases, the amplitude also goes down.

11 - Convolution Theorem and the Fourier Transform.srt

So let's look at the convolution theorem and its relationship to the Fourier transform. So imagine if we were to do a convolution of two filters or two functions here. We'll also want to do or compute a Fourier transform of it. that's a to the product of the individual Fourier transform. So if I want to do a convolution of, of g and h, well what I can always do is convert each one of them to their Fourier transform and just multiply them. The inverse Fourier transform of the product of two Fourier transforms. So here is the two Fourier transforms. And we have inverses of them. , it's a convolution of two inverse Fourier transforms. So to compute this, I would just compute the inverse Fourier transforms with both of them and just do a convolution. And that will result in the answer for this one. The other thing to note is the convolution in the spatial domain, that is the amplitude and time domain. Is equivalent to the multiplication in the frequency domain. Again, these are simple examples. We'll see many practical ways of actually using that

12 - Freuency Spectra for Images.srt

Let’s look at a few specific cases. Now, , now I’m going to also start talking about images. Here, imagine there is an image, and this is a striped image. , I’m giving a very simple synthetic image. And we’re interested now in what would be the frequency spectra for this image. So the frequency spectrum of this would be, again, repeating peaks and pulses at the various frequency, because again, it's a repeating signal. >From here on, you see there is a line this way, and then, , another one and a lighter one. And , there is repeating frequencies this way, which , again, you can see by looking at this signal. So in essence, what it can do is, if we were to represent this image in a frequency or a domain, , now getting more information about the structure, especially in this case, the repetition of the signal. Just to make us understand this, now when I flip this image, so now repetition is in this axis. And as you may have predicted, in essence the Fourier spectrum or the frequency spectrum is also rotated. And now the repetition is in this direction. What would happen if I actually now rotate? So initially, we had this line this way and that way. Now, , we're going to rotate it so it's 45 degrees. The frequency spectrum now, , has a dominant axis going this way. So if you notice, all of a sudden, this is giving us a very interesting way of looking at images and also allows us to look at, for example, the frequency spectrum, which gives us among other things kind of where the orientation and the repetition is happening. In this case, the repetition is happening in this axis. Well, it found it and modeled it very nicely. , these are images if you notice, the gray values and the intensities are different for each and every one of the three examples we looked at. But these are just to kind of showcase the repeating patterns of images. Let's look at the first example we started off with, but this time, let's add some noise. So I've added some noise, again I, hopefully you can see it. What do think is going to happen here? , because of noise, a lot more information, more frequency spectra information is there. But again, the dominant structure is visible again, in terms of the frequency spectra. Let's look at all of them together. Just notice here the variation just because of the signal here is shown this way, in this case, shown this way, and in this case, shown , again, going this way.

13 - Freuency Spectra for Real Images.srt

What happens if we start looking at the Fourier spectra, for real images? So here , bricks. In the previous case I showed you simple examples of just stripe patterns, here has some structure like this, this is a rooftop, also has a nice structure, and , much more natural scene like grass, this , are human made of samples, this is, , purely natural, this spectrum of this one, if you notice, does have two dominant directions, one, , in this direction, the other one also because there are repetitions in this axis too, so you could see a dominant two axis here, shown by , these two variations. What would be your guess for this image here? If you notice, yes, there is a dominant direction because this way, but if you were to look carefully, there is repetition going on in this axis too, and therefore you see, this axis here. See the domain, the axis are visible here and this way, and this one's because primarily there is a, dominant direction repeating this way, in addition to one this way. And you can see that or hopefully clearly in this one. For this one, there's no dominant direction but actually some may claim that frequency repeats on equally in all directions if you were to look at it, and that's showcased very well by, this frequency transform here, it's distributed equally in all directions. , so far, we haven't talked about the magnitude, we'll get to that also.

14 - Fourier Transform Some Observations.srt

So here let me make simple observations about how we can actually do this kind of stuff. Again, notice in this case, looking at it carefully, there is two dominant directions, one in the, one up and down, and the other one this way on this image. We can look at it and actually create a two dimensional axes here. And now we want to actually start looking at what we can actually extract out of. Now recall, when we talked about Fourier transforms we said that when, in a Fourier transform, for every frequency omega that goes from zero to infinity, the Fourier transform holds the amplitude A and the phase phi, of the sine function. That's best represented, and we looked at this before, by this equation. Amplitude A, sine omega t plus phi. So now we're interested in, , representing this. So we can actually extract information in axes like this. The two dimensional axes of an image. what we do is, we can use the real and complex numbers to achieve this. So we'll represent F omega as a real part and an imaginary part here. Imaginary part and the real part. So now if I can start representing my Fourier transform in this way, which means now it's a vector in an imaginary and a real space, we can start doing things like simple vector calculus or vector arithmetic to able to compute things like the magnitude and the angle. So the magnitude of the amplitude, , is nothing else but the square and the real part plus the square of the imaginative part over in a square root and that gives me the magnitude, which would be the A here and v would be the arctangent, or tangent inverse of the imaginary over the real number. So that, in essence, starts giving us information, like we can start computing things, like this axis and the angle, and it can also compute, , the magnitude of this vector. Using these types of things, depending on what kind of signal you're looking at, and what variables you're looking at, we can start computing these kinds of things for an image like this.

15 - Using the Frequency Spectra.srt

Let's take another look at the whole frequency spectrum and see how we can manipulate it and do different types of things with. So one way we could do is now actually can say is, find me all frequencies in this frequency spectrum that's inside this circle, and that would be actually some way of looking at what we want to start doing. Filtering, either, could be low-pass or anything above it. So, , I can now, using this threshold on this spectrum, look at just the frequencies that are above or below this. I can create bounds. So I can only look for high-pass or, in there, just information within it, or just look for information in the spectrum, just in this band here. So that allows me to play around with just using the amplitudes and figuring out which parts of the signal I want to actually be paying more attention to. we can take the same kinds of things for phase radiations, and create bands. And now I in fact just look for information of these kinds of orientations, the image that are just within this or that. And now allows me to kind of customize how I do processing, on which frequency bands or which orientation bands I want to be actually extracting information out of. So that allows us to change the spectrum and also reconstruct this on the kinds of information we're specifically looking at. one of the things we want to look at is, which sometimes in a signal I may want to pay attention only to core signal. Sometimes I may be interested looking at the finer signal. And using that, I can customize my processing to get the right kinds of information out.

16 - Blurring and Frequencies.srt

So let's look at the example of blurring and frequencies a little bit. Remember our peppers image from a while ago? This was my original image. We can blur this by a five by five Gaussian kernel. You can see a little bit of blurriness. It's not very exaggerated because our kernel is very small. Now, I can just take these two images. And if I was to subtract this image from the original, I should figure out where all the high frequency information was, right? And this is shown here, a dark image. I don't know if you can see all the details. But you can see all of the edges. Because, if you remember, when we do Gaussian blurs, that's the one point that we preserve a lot. Because we look at information out there. So while everything is blurred, the sharp edges remain. And now, if I was to just do subtraction of the smooth minus the original, we figure out where all the sharp information is. Now, , this is one way of being able to extract information about where all the high frequency information is. And , this allows us to start paying attention just to high frequency information. And we can run different processes on it. So , the bottom line is, by doing all of this kind of stuff, we can now specifically channel our efforts to improve quality of images, either at the low frequency or the high frequency, and similarly, not just improve processing. We can actually start putting in more attention to the details that are available at high frequencies and low frequencies if you do analysis like this. And that was one of the major parts of us trying to look at the frequency domain.

17 - Summary.srt

So, to conclude this sub-lecture, let's look at what we looked at in the last few minutes. , introduced how sines and cosines can be used to reconstruct a signal. Used that to construct and characterize the whole concept of a Fourier Transform, and talked about terms like frequency, amplitude, and phase. Introduced the whole concept of frequency domain for a signal and an image. And then identified the whole concept of how similar types of things, convolution, and being able to look at different frequencies can be used to actually use a Fourier Transform, doing the kinds of things we want to do. Next, what we'd like to do is start looking at how we can merge and blend different images together. Remember our initial experiment with trying to do this? We want to be able to take different images and merge and blend them together. What we want to also do is start looking for how to do this at different frequencies, so we can merge the frequencies of the image correctly for two images, again, based on similarities of the frequencies for those two images. You'll see examples of this in the next lecture. And just to conclude again, some more information is available on these sites, and it'll be made available also on the class website. Thank you.

# 04-02 Blending.txt

01 - Intro.srt

So now, we are going to start talking about how we blend or merge two different images together. Remember different types of things we have looked up before, where we can do different types of processes of enhancing the quality of images, or being able to do additive information on images. Now, we're going to look at how we can actually take two images, put them on each other and be able to kind of get the pixels registered and actually provide you a much more crisper information after that blending is done. We will look at some concepts of both Fourier analysis we've looked at and see how we can actually apply this to things like panorama building, and use that concept to build further to hold technology of how we can do blending and merging of images.

02 - Lesson Objectives.srt

The objectives for this lesson are for you to learn about how to merge two images. A specific approach that we will also be looking at is, what amount of one image and the signal from one image, do you want to merge with the other image? And we will look at the whole concept of window sizes, that will allow us to quantify how much of one image we want to merge with the other. And then , we'll go back to the whole concept of Fourier analysis to help us look at how we can do this merging and understand more about the frequency domain and the sampling that's available in each image, that could be used to generate a novel image.

03 - Recall Combine, Merge, Blend Images.srt

So, by now you may be a little curious as to why are we trying to do this? So, let's look an example that we have looked at before. You may recall this image that we looked at before, where we generated a panorama. And if you look at this carefully, you'll see, again, that there are different sub images in this image. So for example, you can see some of the boundaries of different images here. And , each and every one of them is a separate image of their own, and what we're interested in now is taking all of these images and merging them together so you would have a full panorama. So this, , is the output. Now you cannot see any lines across the whole spectrum of this panorama. All different images have been merged together. I will flip back to the previous one again. So just looking at these flips, you should be able to see that the basic boundaries between different sub images have vanished to generate a complete smooth image. And that actually is what, the output we were looking at. So in this lecture we will look at is how to go about coming at various techniques we can to be able to blend images so these types of edges that exist when you merge two images together vanish.

04 - Merging Two Images.srt

So to help us go through this whole process of learning how to merge two images, let's look at two specific example images. Here I'm going to use, , a image like this and another image like this. Both of skin of two different well known cats. To help us with our analysis, let's just deal with black and white images for now. So what we're interested in is merging these two images together to form one novel image. What we can do is take one image, take the second one. And, as we learned in one of our earlier lessons, just do some sort of point arithmetic to combine these two images and come up with a weight of taking one percentage of the other, and one percentage of the other. So, for example, I could just take 50% of the pixel values from this image and 50% from the pixel values of this image and just do a simple pixel arithmetic. Just add with the weight 50/50 between those two. That would show us this blended image. So , here you notice now real blending going on. This part of the spot here, and this part of the spot here are kind of now merged. And there's transparency and there's overlap of these types of spots from one image to the other. , we could play around with how we merge this thing, get more information from this, less from that and now actually, we can see that this spot is less visible, that spot is more. So again, variety of ways we can do these blends. Let's look at this again a little bit more carefully and see what we can do with it. My goal in this exercise is for us to now take this image and that image and put them next to each other and blend them just around, in this region. So for example, this is my boundary between those two images and I want to be able to blend aspects of this image, this side and this side, and I want to have a blended image right in the middle where the line is, somewhere in this range, across the whole region. So this box, let's say, you know, just a few pixels from the center here in this here, I want to do all of the blending, so I want to actually have this spot here, and not just have a sharp line, but merge over here, and similarly, these two spots kind of also have a little bit of blending going on. So that's our goal.

05 - Cross Fading Two Images.srt

So one of the ways of doing this is cross fading between the two images. As I go from one side of one image, let's say from here to there, and on this image I go from here to there. So this is left image and the right image. I would cross fade this direction on the left image and this direction for the right image. So, in that, in essence what that means is, we will cross fade this way here, and cross fade this way here, and generate a new image. So to help us do this, let's imagine that now I can create a small ramp. This ramp has a value of 1 here, and 0 there. And then the other one, the value of 1 here, and 0 there. And what I’m interested in really is being able to now go and take information, all of the information, from this part here, from this image, but reduce it as a go and traverse down this way. And similarly, take all of the value from here, and reduce it as I go down there. And if you notice, the mixture of these two would kind of start having blended values in the middle all over here starting from the first. Our first column here and the last column here. one of the ways of doing this, if you remember our approaches of trying to do this with simple point arithmetic, is creating a ramp image like this, all whites, all blacks. And similarly the transpose of the opposite of this where all white here and black here, just doing this multiplication plus some operations will give us an image like this. Again, notice this is cross-fading. All of the information for this image is going down to 0. And this one from 1 to 0 and here we see a little bit of blend here. 50/50 is showing up here. So this is a simple technique of cross-fading between two images along this direction. This direction for this one, this direction for that one. And we get a blended image. On this side, , you see more of this image. And on this side you see more of this image. Now , there is no reason for us to have just ramps that are linear in this manner. We could actually also play around with this, and have them have different types of, of functions perhaps even step functions to give it more emphasis in one region, less emphasis in the other. So what does it mean for these two images? Again the left and right image can generate our target. But now I want to play around with how I do the blending, or the cross-fading between those two. Again, lets look at our ramp functions. And this time around, I have one, but when I get to this point is when I actually start giving it less values from this one, the opposite here, up to here, and then all of sudden I can use a step function or a ramp that starts off a little later to actually do the blending from one to the other. So, in essence, now what we will see is the entire effect of this blending will only be restricted to this region of the image or to this region of this image. So this would be what I would refer to again as the window, the size of the window where I would actually do the blending. I don't want to do any here and I don't want to do any here, but I just want to do it in this small window between these two left and right images to generate a target. , in this case, I could simply create again the simple ramp images to a simple point arithmetic, between those two, this ramp is exactly. On this profile here, this ramp is based on this profile here. The output of this one is this image. Again, if you notice, and look at it carefully here, that these are the lines where most of the blending is. This is originally the same image here. This is originally this image here. Most of the blending is in this region. So in essence, we've created now a ramp function or a cross fade here where the value of blue from here on this side of the left image is exactly the same, crosses over and becomes 0 here. And similarly on the other side the green which is representing the right image, goes up in here, crosses over and becomes 0, so most of the blending is right there. We can play around with these types of how big the window size would be. So the same process, except now my window is much narrower. So this would be the size of my window that I actually have that I want to do the cross-fading in. look at the same examples as previously, we've just done a ramp with a little bit of change in the middle. And this one you'll notice, this, this is exactly what the ground functions look like, black, all zeros here. Quick jump from 0 with a little bit of gradation and colors to white and that's what we see here, and that's what we see here. Doing the same arithmetic, we get this image. And here you notice, again, most of this image is like this one, most of this image is like this one. And here there's a little bit better example of blending. Now, you can still see a little bit of sharp edges here. So the question now comes up as, what's the right size of this window? So one of the things now we need to start looking at is how do we best determine the size of this window to give us the best results of doing these cross faded or blended images.

06 - Cross Fading Window Size.srt

So let's look at three different cross-faded results we looked at, and let's compare them. Here, the whole blend from the entire size of the window, so the window size here was, the entire image. In this case, the window is this region, and in this case, this region. Each one of them has a different effect, here, you can still see a little bit of a sharpness, a drop-off. Here you see, less sharpness much smoothness but you do see a little bit of ghostly artifacts. And here everywhere you see transparency and ghost artifacts. Now you can see all these images clearly, again, more ghosting, transparency artifacts all over the image, just restricted to this region, and this one is much nicer but does seem to have a little bit of a crisp sharp line that does show up in between.

07 - Factors for Optimal Window Size.srt

So now let's ask the question, what are the different factors that impact the size of this window? And we want to be able to find the most optimal window for a series of target images to generate a novel image that gives us the best possible blend. So to avoid seams of any form. The size of the window that we want should be equal to the size of the largest prominent feature. So in this case if you look at it, the results look good here. But we do know that this image does have some issues with this spot kind of feeling a little cut up. Well, if we had actually kept the window size big enough for this one, we would have actually done better. Similarly, I mean this does very well for this because the window size is decent for this one. So in essence what we need is we want to have a window that is equal to the largest prominent feature on an image. Another rule of thumb here is that to avoid any kind of ghosting artifact, where actually you saw in the previous image where some transparent spots were visible across the images. We want to have a window that's less than or equal to 2 times the size of the smallest prominent feature. You take any small feature, we want to make sure our window is twice as big. So again if this was the feature we wanted to make sure our window size is big enough for that to be able to kind of hide away and not have any ghostly artifacts. And again you don't see any here. You do see it a little bit but remember we are only doing kind of blending in this direction here. This is where some of the stuff we discussed earlier with Fourier transforms comes in. Remember this example where we looked at a synthetic image which has these bands are going across the image at 45 degrees. If you were to compute the Fourier transform of this image, we would get this. Which kind of now shows the frequency spectrum and the repetitious nature of this image. Among other things, this repetitious nature also gives us a sense of the scale of different features that exist in this image and perhaps also the repetitions of this feature. So, in essence, this frequency spectrum is showcasing how big the features are in the sampling space, and how we can actually take the sampling size of the signal to account for what we want to look at. Again, something you may recall, we discussed in the last lecture. So using this and by then doing any kind of specific frequencies to look at, either low pass or high pass or band pass. We can now actually look at specific frequency information within an image. So for example, I could just create a region. And I would look at all of the frequency information just in that one and concentrate on just those to be able to extract the window size. And this is just how I'm showing it. I'm ignoring all of the information everywhere else but just here. So, in essence what we are arguing for here is, we want to be able to start using the Fourier domain to help us extract optimal window sizes. Here I share with you some of the well known methods, how we can actually do this. For example, if the largest frequency of the signal was less than or equal to twice the size of the smallest frequency that would help us be able to kind of look for the right window size. And the other thing to remember here is that the image frequency content should occupy one octave. That is just be able to look at within the powers of 2 of the frequency signal itself. And that will allow us to also determine the window size of this kind of stuff. Again, these are theoretical foundations of how we want to do this. As we go on further we'll learn that we actually can learn and model these things quite well to be able to do kind of blending that we saw in the example of the cricket field earlier.

08 - Frequency Spread Needs to be Modeled.srt

So , we are invested in looking at the Fourier transforms of the two images, the left and right image of this parts, so let's look at the Fourier transforms of those images. But this image, this is the Fourier transform. It does have a little bit of a dominant structure, but as you notice, it's got a lot of spectral frequencies distributed around it, because it does have, while it has reputations in x and y. It does have a distribution of this thing because it's not the most structured image. Similarly, for the other one, , here you notice it should have more frequencies. These are larger objects or larger spots. These are smaller spots. When you have smaller spots like this, you expect the frequency diffusion to be much thicker. And, , it does have the directions but not as strong as in this one. So, , this has more frequency. This has less frequency in it. So, continuing to look at the same to examples of the Fourier transform of this on the left and the right spots here. We have these two different frequencies of spectra, we can compute those. So we would compute the fast, the Fourier transform. Here as an FFT, which is fast Fourier transform which is, and this is so far the same thing. And this is the left image which give us Fl, and similarly we do this for the right image which gives us Fr. What we can do now is actually, again if you recall from the last lecture, we can decompose this into different frequency spectra. So we can take the left image and break it into it's F1, F2, F3, all the way. And we can do the same thing for the right image, F1, F2, F3, F4, and that gives us another whole set of you know information that is actually now separated out into its different bands. Now to help us generate these images what we can do is we can feather corresponding octaves. Now I know I haven't covered the concept of feather. I'll talk about that in just a few seconds. What we actually want to be able to do is feather. That in essence means is we want to kind of merge the right kinds of things from two different images. We want to do this now in the frequency domain. So we want to take the frequency, the fast, the Fourier transforms of the left and the right. And we want to do the corresponding octaves, remember the F1, F2, F3, F4. For the left and then the same for the Fr. But we want to do the combination of the F1 for the left and the F1 of the right. And, do a feathering between them, and do it for each and every one of them. And then use that to reconstruct the signal. So in essence, what we want to do is, we want to compute the octaves, feather them, and then compute the inverse fast Fourier transform. And that will allow us to feather everything in the frequency domain and actually the output would be in the spatial domain. So now what we can do is sum the feathered octave images in the frequency domain and we actually have a very nice smooth image. Let's look at that example.

09 - Feathering.srt

So let me, you know, again, just look at the whole concept of feathering. Here are my two images. What I can do is, I can take, and , we would be doing this at the frequency spectra for each and every one of the octaves, F1 and F2. Here I'm showing it in the spatial one. Imagine if I was to take this signal like this and do a, you know, computation as we've done here. Before, again, remember these are just step functions of different types. , you'd see a sharp boundary. So, one way to do this would be, is, I can blur this sharp boundary. So I can run some sort of a blur kernel on these two band images here. So this would actually make this area be not just a sharp step but a ramp between the two. So that would be the output here. And if you actually use this just for feathering between these two images, that's what the output you would get. So now you actually see, there is blending going on, but it's much smoothed out because of the fact that this is no longer a sharp function. , the important part to remember is, we want to do this at each and every different octave. So bottom line is, in doing this process, we can smooth out the images, because in essence doing this blurring and all the entire process of feathering makes the blend smoother. So recall again, what I'm really interested is saying is, we want to do this feathering, but we want to do this for the corresponding octave in the Fourier domain, we want to compute the inverse Fourier transform. And this allows us to do this by summing the different types of octave images in frequency domain, and allow us to generate a smooth image.

10 - Summary.srt

So, to summarize, in this lecture we covered the concepts how we can merge two images. I specifically talked about the cross-fading aspects of it. We talked about different issues related to it. And, in essence, how we can compute the size of the window. And we actually looked at the whole concept of Fourier Domain again, that we had looked at previously to help us coming up with some rules and some processes that would allow us to look at different types of window sizes. In the next lecture, we're going to take this process a little bit more forward, how we can actually do merging and blending. We have looked at merging and blending. We have looked at how we can do this, and understand the issues of window sizes with Fourier domain and all that kind of stuff. But now, we'll put it to much more practical tests, and look at how we can actually do this by use of pyramids, where we can do the blending at the lower frequencies, and higher frequencies separately. As we had discussed, again, we wanted to do feathering and stuff. But we’ll come up with a much more well-defined approach of how to do this, and , you’ll have the opportunity to play around with this also. Again, just more information related to the material we covered right now and additional information will be made available on the website.

# 04-03 Pyramids.txt

01 - Intro.srt

So now, we're going to get a little bit more practical. You will get to see that, if I take an image of Tigger here and Einstein, my two friends that you've seen through some of my lecture now, and if somebody gave me two images of this, how would I blend them together? So in this lecture, I'm going to take what we have learned so far, and introduce the concept of Gaussian pyramids, that will be then used to figure out, how we can merge the images of these two characters, but at the same time, maintain the frequency information of both those images correctly. So we can do the blending, at the right frequency spectra.

02 - Lesson Objectives.srt

So the objectives of this lesson are for you to learn about Gaussian and Laplacian pyramids. I'll introduce what those concepts are. Then I'm going to showcase how we can use these pyramid representations to help us encode the frequency domain. We will learn how to compute the Laplacian pyramid from a Gaussian pyramid. And finally, we will learn how to use the Laplacians and the Gaussians to help us blend two images together. That, in essence, actually encode the frequency information in both the images

03 - Optimal Window Size.srt

To get started, let's recall a few things we've looked at. One, we looked at the whole concept that if you were to do blending of two images, we have to look at and find the best window size that will allow us to do the blending. We studied that , to do this you have to account for the size of the largest prominent feature, because that allows us to kind of blend things together. We also studied that, the window size depends on some information about the size of the prominent features, and to avoid ghosting, you'd have to have a window size that is less than or greater than, the twice the size of the smallest feature. And also, this motivated us to look at us to look at the Fourier domain, where we look at the largest frequency and the smallest frequency find the size of the window, and also looked at the concept of octaves where powers of two allowed us to start looking at different frequency domains and, and let us calculate , the different Fourier transforms that we use. to figure out how to dress do the blends.

04 - Frequency Modeled.srt

Using that, we said, even taking these types of images, like this, we can actually now compute the Fourier transforms that actually encode the frequencies, and we can use that in doing blending. So , what we did was, we computed the Fourier transforms for the left image and the right image. Then we decomposed these Fourier images into octaves, which were, , the different bands here. So for the left one and the right one, we computed all the different octaves. , then to blend them, we have to feather them. And what we did was, we took the octaves, separate ones, for different frequencies, for the left and the right, and feathered them together. And then , after all of that was done, we computed the inverse Fourier transform and also did some feathering in the spatial domain. And then took all of that information together to generate a feathered octave images in the frequency domain. And that was the process we went through. So, , we computed the Fourier transforms of the images, and feathered the corresponding octaves. And used that, and built back up to generate a blended image. Now, let's look at another whole method of trying to do this, which will, intuitively, be a little simpler. And, actually, at the same time, allow you to build your own tools to do this kind of blending.

05 - Pyramid Representation.srt

So now let's look at the whole concept of how we can actually use a pyramid representation. And specifically in this form, we'll look at a Gaussian Pyramid. That'll allow us to encode the frequencies at different levels and let us do blending or other types of operations that look at different frequencies differently. So just to start off, let's start off with this simple eight by eight image. All right? So this is just a simple representation. I'll use this to showcase what we can do with this. But let's now go back the concept of doing simple filtering. I'm going to run a small Gaussian kernel over this. Let's say it's a three by three. And my goal is to, let's take the information from the filtering process here, for this one, and create a new image. Except that I will just do this, to kind of replace it, make a smaller image out of this. So my goal is to generate a four by four image out of this. , what we can do is, now look at this, for example, three by three neighborhood. Run a Gaussian kernel on it, get a new value that encodes all of the information there, and replace it in my newer image. So using this, I can now generate a four by four image. So in essence, what I've done is, I've taken every other value. These are pointed here. So I've taken these 16 values and put them here. Except what I'd rather do is run a small convolution with the Gaussian and take the value that's the resultant of all of the convolution here and replace it in this one. So in essence, that's what we've done here is, we've gone from a eight by eight and generated a four by four. We can, , cyclically repeat this and generate a two by two, which is shown here. And , at the end of it, what I can now do is run a kernel and find the value at this point here, which would be the, you know, the Gaussian kernel of these four values. And generate a value here. So if you notice, what we've done is, we've gone from eight by eight to four by four, two by two, and , one by one. We can do this for real images too. Here is my Einstein image. Let me refer to this as g0, the Gaussian at level zero. And , the first one, I'm running a Gaussian kernel, now actually generating an image like this. I refer to this as g0, to refer, , to the concept. This is a Gaussian at level 0, this is a Gaussian at level 1, here , run through the same process, and now this 512 by 512 image is converted into a 256 by 256 image. And we can keep doing this. Generate g2 with level two. This is, , now a half of 256 by 256, so 128 by 128. And then go on all the way to a small image like this. So each one of them is referred to now as a Gaussian. At different levels, two, three, and four. So let's see what this representation looks like. What I'm doing now is moving the smaller ones on top of the original image. And if I was to just draw, for example, this point here we can actually see that in essence, what we have is a pyramid with the bottom, the highest resolution, followed by the next higher resolution, going all the way up to this point here, and by connecting these lines, we can see that now we have is a pyramid representation. And that's what the pyramid looks like. So we've gone from information that actually was very high resolution to low resolution and by actually doing Gaussian filtering to kind of now scale these things up, we have different levels. This was, , the zero, original, one, two, and three.

06 - Pyramid Representation of Images.srt

Remember our concepts of using kernels, to help us do this. In this case, let's actually build two kernels, one in the vertical direction and the other one in the horizontal direction, and just for the sake of the completeness, here actually given you the values where a and the one fourth, number here, one fourth minus a over 2, in both directions, again, symmetry, in v and symmetry in h, using this, , we now should be able to compute this, this is a concept laid out in Berton Adelson, and I refer you to that paper for more details. In this case, a values are computed to be the following, by Berton Adelson, and , now using these, horizontal and vertical kernels we can actually create an h which would be the kernel we want to use to doing any kind of processing, and this is what was used, as an h function to compute all of this.

07 - Building a Pyramid REDUCE.srt

Let's look at it in a little bit more detail. So, what we're interested in, , is to compute the Gaussian at level k. We take the h, which was one we looked at previously, convolve it with the Gau, the original Gaussian from the previous level. So in essence, that's why it's a pyramid. So you take the previous level to compute the next one. You take the previous to compute the next one. And this, in essence, is known as the reduce function. takes the previous image to generate the new one. So for example, this is my image at level one. I can use this to generate an image at level 2. Here, while the samples of these images that go from g1 to g2 is half, I'm still scaling it up, all of them to be the same scale to kind of show what's happening as we go down. , g3 would be based on g2. So here, , recursion is going on. These are recursive functions. g4 is based, now, on g3. Again, I'm scaling them up, all to be of the same size, to showcase the fact that information is being blurred out. So, here, , you see a lot of pixilation, and even more. So we've gone from images that were 256 by 256 to 128 by 128 to 64 by 64, 32 by 32, and 16 by 16. Now we can actually start noticing an interesting fact. This is extremely blurry, much more crisp. This is encoding all of the high frequencies. This is encoding all of the low frequencies. Here you can see the blurry shape of Einstein. Here you can see all the details, including even, for example, the hair.

08 - Building a Pyramid EXPAND.srt

So in essence, what has happened in this process is we've generated a pyramid where we can do coarse computations here at the top of the pyramid and fine computations at the bottom of the pyramid. Again, by just doing this recursion using this to compute this one, using that and going up the pyramid. And, , we can move down and up the pyramid, do whatever kind of processing we want. So now we know that given any g0, I can compute g1 using the reduce function in a recursive manner. So replace this by g0, and I can get g1. Now the question is, if somebody gave me g1, could I actually compute something equivalent to a g0, or some application that could be used to generate something similar to it? , there is a function called expand which lets you take any value from here, or information that is from here, to generate this. Let's look at an example of that and we'll talk about in a bit. Now I'm going to show you what the output would be from this function if you were just given g1 to create a version of g0. , we will call that g0,1 and that's what the image looks like. Which implies that it's g0, but actually expanded once, from here to get there. Now for those of you looking carefully you may see, oh not much of a difference. Just to see what the difference is, what I'm going to do is I'm going to take the expanded version of this image, and subtract it from the original image. So this is what you see now here, and this is , , showing you where the differences are. You notice a lot of high frequency information is much more visible. The eye-lines and all that kind of stuff are much nicely defined. And in you can see that this is the output of which actually is showing you the error in reconstructing it backwards. Remember, g0 was the original image, g1 was the one we got by using reduce. Then we said let's come up with a function that takes the reduced image, and expands it back to the original which is g0,1. We did this from the original one, that's why this is g0,1. And , when we subtracted it, this is what the image we got. So in essence, the expand is the inverse of the reduce. As attempts to add new values between known ones. Again, remember the example when we went by an eight by eight simple one to generate a four by four. In essence, what we are trying to do is go from a four by four to an eight by eight, in essence meaning we will be adding new values. , there will be error, and we want to be able to measure it. In this case again just to get the notation correct. G, g0, 1, was attempting to keep the value from g0 just being able to expanding once, in essence, in general terms gj, n is gj expanded n times. Remember, we could even go from g2 to g1 all the way to g0. So, recursion is possible this way, where we go from g0 all the way to gn. Or, or, g, you know, a larger numbers. Or, we can go now from our gn, the level that we are at, back all the way here.

09 - Difference Between Pyramid Levels.srt

Let's look at some more details of this. What we looked at in the last one, example, was an error function between the expanded minus the original version of the image, and we called that a error image, and , it's referred to as a Laplacian. Here, , we will refer to as L1 for the first level. , using this same method we could computer L2. The expansion is in essence done by doing this equation here. We expand from one level to the other one. And then, , subtract it from the original to get a Laplacian. So that's how we computed L1, we can generalize this to computer L2, L3, L4, and so on. In essence, what these Ls or these Laplacian images are, again a pyramid. Because again, they can be stacked on each other and , they encode in different frequencies. But there are series of error images that form a Laplacian pyramid. And in essence, what they are doing is computing the difference between two levels of a Gaussian pyramid. At one level I got the Gaussian value and I got it at the other level because of, of how I computed it. And now I can difference it and actually starts giving me, what was the error between all of them? So now let's think about how we can use this information.

10 - Computing Gaussian and Laplacian Pyramids.srt

Let's look at the pipeline of how we're going to compute Gaussians and Laplacians. I start off with my first Gaussian at the first level. We know how to compute that using things like reduce. , now I can also compute a expanded version of the same image. So in essence that is, in looking at the blur function a little bit. And , I can compute a Laplacian, which is the subtraction of these two. Then I can, , sub sample to get a smaller version of the image. Which is where we go in for the pyramids and Gaussians here. So this is Gaussian pyramid, this would be G0, and this would be the expanded version of the same. And , the Laplacian. So notice that we can actually create and compute both the Gaussian and the Laplacian pyramid for each and every image that we want. So, here, I'm just showing you a Laplacian pyramid and also a Gaussian pyramid.

11 - Pyramid Blending.srt

So, how do we do pyramid blending? Again, my original image here, and what we have is this pyramid representation. , I can compute, just as I showed you before, the Gaussians for this and the Laplacians for this. So just to make it interesting in this example next to the Einstein, I'm going to use this Tigger image and we're going to do blending between those. Let's actually go ahead and showcase all of the, the Laplacian and the Gaussian pyramid levels for this image also. So now we have both the pyramids again we're going from coarse to fine. Now if you were going to do any blending between this image and that what we could do is. Again, learned something with it before. And just cross-fade at each and every level. Right? I can take parts of this, parts of this, and merge these together to generate a blended image. And remember when we talked about cross-fading images, we said let's do it at the different frequency bands. In essence, that's what we're doing here. That we're going to look at the different frequency bands and do cross-fading individually at those frequency bands, octaves and merge them together.

12 - Blending Example.srt

So let's look at an example. Again, Einstein meets Tigger. Now, our goal is to blend them that half of the image comes from here and half, the other half comes from here. And , just to doing a sharp blend like this, if we were just to put them next to each other, we would get a sharp line. What happens if we actually went through the process I discussed earlier? Notice there is no sharp line anymore. Because what we did was, we actually deconstructed into different pyramids. We blended, again, with this 90 this kind of half cut between both the images. Took pixels from both of them. Well when we reconstructed it, this frequency mixing going on, and kind of see the Tigger skin merging with the Einstein skin, the eyebrows and all of that kind of stuff are merged at least at these boundary lines. Now, this is, , a simple case. What happens if we wanted to do this kind of blending for a specific case where you wanted to actually have the blend happen just for specific regions of the image.

13 - Masked Blending.srt

So here, I'll showcase a much more detailed example, and this is something you will be experimenting with various programming assignments. Given an image A, and image B, we want to blend them, but actually we want to add one more constraint. So in the previous one we looked at, merge it at half from this and half from this. But now actually, I want to give it a mask, a region where I want most of the blending happening. Just to make it a little bit of fun, what I've done is in this example, the mask region is based on the hair and face of Einstein. because what we're interested in is taking the face from Tigger and merge it with just this region, and ignore all of the information that's black here. So this gives us a way of now kind of doing the blending in a specific region. How would we do this? Well, first we'll build our Laplacian pyramids. , that requires us to build Gaussian pyramids, too. Also, we want to do is build a Gaussian of just this mask region. We will actually use this because we want to be able to use that as a kind of a filter, a binary filter, with only positive values where it's one, we'll actually go through and zero where it won't. So form a combined pyramid using Gaussian for R. And, this would be the way we'd create each and every Laplacian. So Laplacian would combine the Gaussian of this region here with that of an A, because we want all the pixels from here. And then, , then subtract it. So then we get all of the other, the, the composite of this one from B. , this how we would actually use it to create a combined Laplacian pyramid, which would actually be blending information from one to the other. And then, , after this thing has all been done for each and every level, we would collapse the Laplacian pyramid to get the final blended image.

14 - Masked Blending Result.srt

So here again, my input. And that's my output. Notice now that the eyes are merged, right? And the hair is also kind of much more kind of a mixture here. We still see a little bit of the Tigger ears. But more of the hair of the Einstein image is now actually been put on the Tigger head. And you can see the frequency mixing is happening again much nicely. This shows where some of the laplacian errors were. I mean again, it accounted for a lot of the high frequency information where the eyebrows are. It did have a little bit of problems with the ears. That's what showed up here. And also near the edge of the mouth here which is showing up here. But overall hopefully you'll see this is a much nicer blend than we have actually see of these kinds of images so far. And the secret was we actually counted for all of the frequency variations, and put them again in the boat, the Laplacian, the Gaussian. Up the scales, we were able to use this information.

15 - Summary.srt

So, in summary, what we learned today was, we learned how to merge two different images, but we wanted to actually use the frequency domain in much more detail. We use the gaussian's and the Laplacian pyramids. We learned the mathematical formulations, how to compute a Laplacian and a Gaussian. And then we learned how to blend two images using these pyramid structures. Next class, we will actually go into deeper of trying to understand how to do merging of, blending of images, but this time rather than just fading from pixels, choosing which pixels are the best. So, for further information, I encourage you all to look at these papers in a book that I'm referring to. These will give you more details on how to do it. You will be doing an assignment on pyramids, and I encourage you to play around various types of coding to actually write big span and the reduced function that you've talked, we've talked about so far.

# 04-04 Cuts.txt

01 - Intro.srt

So if given two images, merging and blending by taking the pixels from those two signals tacking them on top of each other, and then blending the pixels together would be a good solution, it's not always the right solution. So in this lecture, I will introduce, how we can take these two images that are on top of each other, and cut one, and the other appropriately so you would get the exact pixels from one image and the other, and therefore, there is no blending, which means there is no ghosting or blurriness that comes in by doing things like blending.

02 - Lesson Objectives.srt

The objectives for this specific lesson are, I'm going to talk about the additional method that allows us to generate novel images but not, no longer using the blending, which takes two different images and figures out the best pixel merge between those two. In this case, we will actually figure out which pixel to use. And the method that we will actually come up with will be to find a seam in the two images. And I'll talk about the benefits, one versus the other.

03 - Recall Combine Merge Blend Images.srt

Just to recall again, let's take our favorite panorama image. We want to be able to take an image like this there are lots of images, in this case the panorama was scanned this way. And what we want to do is find, to generate a novel image the pixels from one image and the overlapping image beneath it and give you best possible pixel value within it. Today, we're going to look at another method to do that. This shows the example where blending can be done to generate a novel image like this. And with the right proper thing, you get a beautiful image like this. In a past lecture, we had looked at how to do this by cross fading different images, and looked at different sizes. If I looked at the whole size here by putting two images on top of each other, doing some sort of a cross-fade. I get a little bit of a ghostly artifact, I can do this for a smaller window and I get the artifacts right here. Or you want a smaller window. And we looked at various methods to actually figure out how to use the frequency information to look at the size of these types of windows. But when we do any type of pixel and pixel merging from one to the other, you will always have some ghostly artifacts.

04 - Cut Don’t Blend.srt

Let's look at another method to do mixing and merging of two images. Let's put these two images on top of each other as we've done before. And rather that in the previous times what we've done is take some sort of a weighted pixel value of the top one, or the left one, and the right one, and give you a new answer. Here is another way of looking in the same problem. What I've done now is found a boundary between these two images here and taken all the pixels from this side from this image and this side from this image. So here this boundary now kind of says is, I'm going to take all of the pixels from this image, from the left image here and all the pixels from the right image here. And this will allow me to now kind of create a new image that has the actual pixels themselves rather than a mixture of two pixels. Here you can see the boundary a little bit, . But we know by doing various types of simple feathering and stuff like that, we should be able to get rid of this artifact. And this would be a perfectly clean image. Let's look at a real example that would allow us to do this. Here I'm showing some of the work by Davis et al, here. And the idea is now let's imagine I took a camera and I rotated the camera, panned the camera over sight. And now I have images from all the way from the right to the left. And now I have the sequence of images, I want to merge them together to generate a panorama. This shows you the sequence of images. As the images panned across a construction site, you see a construction crane followed by a, you know, a dump truck, a person, and, all the way out we end up in porta-potties like this. Here is the video again. Panning across, a person is moving, lots of motion, and the camera is moving. Here are the images six of the eight think images. This sequence has about I think 12, so here are construction crane, dump truck, construction worker moving as the camera is moving to all the way towards the other end of the scene, we're interested in taking all of them and merging them together to generate a new image. Here is the process we can actually do to find a cut rather than blend. Here I take two images, and I align them and after I align them I look for a difference on these types of things, I'll talk about that in a bit. And this is going to start saying where all of the things are similar and where there is some sort of merging going on here because, the person moved, there will be a little bit ghostly artifacts here. But within there, there is a region which might actually be the best possible one to cut between them, it's also shown here. If I found this cut between these two images, now I can blend this, and the person will actually be not ghostly, because I've just taken this part from there. So in essence what would happen is for this right image and the left image we found all the pixels from here and all the pixels from the right image for this one. And this is where the seam is. To say that, which parts of the image I should be picking up information from. So, again, in that image if I had done blending as we’ve looked at in the past where I take overlapping images and find the best pixel and kind of had a weighted, you know, pixlar arithmetic between them to kind of come up with it, I would find there would be, you know, ghostly artifacts with the person in the dump track. So moving objects like with a moving camera will cause some sort of ghosting. So we want to do is find an optimal seam as opposed to blend between two images. Finding an optimal seam will allow me to now come up with an image like this where everything is crisp and clear, no merging of information. So this image shows you the boundaries that we can compute. And these boundaries allow you say that I'm going to take all the pixels from here, pixels from this one and allows me to now find the exact pixels that we want. So in essence, by doing this we're finding the exact pixels that were captured by the camera. And figuring out the best pixels from each and every one of the different images we caught, we captured. And therefore we get a crisper image. No blending or ghosting

05 - Find a Good Cut.srt

So to help us understand this, here's a simple quiz I want you to play around with. I have two different smaller six by six sample images here, and we want to blend them or merge them together. And here my question to you is, where would we find the best seam? So here is a simple example of two simple six by six images, and the question we want to look at is, which are the best possible seams where would we cut these two images, the left image and the right image, to generate a new image. The output we want is an image that's blended like this. Just mark out the pixel boundaries in this example where you think these seams should be.

06 - Find a Good Cut.srt

The answer, , is, this boundary between these pixels here, so I'll take all of the pixels from this image here, and all of this one, and they'll let me generate a file image.

07 - Minimizing Overlap Error.srt

So let's do this with a simple example here. I have two different images. There are oh, two smaller images like this. And if you notice, , there's a lot of similarity. We want to now find, within these two images, find the overlapping regions that would allow us to generate a smoother, novel image. If I put these images next to each other with a little bit of an overlap, I get a sharp boundary, a vertical boundary. , I want to get rid of this vertical boundary. So, let's start looking at the regions. So this is the overlapping region, between the two images. I take the two source images and look at the same overlapping region. This is the overlapping region between them that if I tried to do a simple cut, that's what it looks like. Let's look at these two overlapping regions. If I was to take these two overlapping regions, and my interest is to do looking at the difference between these. What we can do is look at the, the squared difference between them. So I take the difference and square it. So take all of the pixel values from this region, they're the same size small images, and compute a difference between them. This would give me a, an image like this. And within this, if I find the minimum values in this direction the horizontal direction, I will get a region like this. These are where the differences are the lowest. So this kind of says, is now I can take this, and generate a region like this. And if this was the cut I put in through this overlapping region, I would get as smooth as possible image like this. Now if you notice, these two images, you can't see the seam. So in essence, what we did is we took these two overlapping patches, create a difference, squared it. That started giving me a value of where things are. With in this, I found the minimum error boundary. Use that minimum error boundary to get enough pixels on the left and the right, and the overlapping region, and generate a new image. Okay? A new novel image that has the actual pixels but, more importantly, now you cannot see the seam.

08 - Finding Seams.srt

So here I'm going to actually look at a specific example. Here the goal is, I want to take a small image, let's say this is a 120 by 240 image of a strawberry, but I want to generate from this is something twice as big, so 480 by 240 image. What I'm going to do is now take this original image of strawberries and replicate this across the whole region. Except the intention is now to do this. And as soon as I replicate one to the other, I also keep on finding the seams between them that would allow me to generate a newer, larger image. So here, , I copy the overall image over. And when I copy it on top, I also find a seam. Our copies would be overlapping. But then, , I do this by randomly putting them in different regions in this manner, and now the yellow lines show that we have found the best possible seams. And now this allows us to generate a much bigger, and a larger image which has more strawberries. This example can be applied to many different ways. I can use this to generate a larger version of this. I can generate a larger version of this. Or even a larger version of this flower image. I'll talk about this next, in a bit. But more importantly, notice what happened here. I extended the Machu Picchu image, except that when I extended it, because of a variety of reasons, it actually added a hill, image is smooth. this is now no longer the real image, because the real Machu Picchu scene would not have these three different hills. But now we actually generate images like this. So in essence, this method that I just talked about, finding seams, can be used to generate much larger novel images, fake images this way, too.

09 - Extending Images.srt

Here the example was this is my smaller images and I want to generate a larger images like this. Now, here we do a couple of additional things. The idea here is you want to generate a larger image, but we also want to capture the perspective. There are only two flowers in this image which you can see them replicated here. Replicated different ways here, but this allows us to just like the strawberry image, we can actually do this across the thing. But how do you make them smaller? Well, what you can do is you can actually make smaller images of this. Remember the pyramid representation that allowed us to look at different aspects of different scales of an image. Here we can actually now find smaller images and add them into the representation and add a function that says, as you go away on top, make the shorter one show up there, because the four short mingle perspective is visible in that axis. And that way now I can generate a much larger image, and as I go further into the scene, the flowers get smaller. And , you do see they are the same flowers, just merged differently.

10 - Editing Images.srt

Another set of examples are this. So here, , what I want to do is I want to now merge two images. I have a background image, and a foreground image, and this is showing you the UI, that by taking pixels from the background and the foreground, I can now find a best possible seam that lets me merge these two images together. Let's look at this example again. I have a original source image and a target image. this, , is a blurry, you know, stream. And then I want to, on top of it, put this raft. What I do is, , take the source image. And we have a nice UI for doing this. And I say, take all the pixels from this image here. But now, take this image. But in this case, take the pixels from just the region I'm touching. Find all the similar ones from there. And , what it does, it finds the best possible seam and blends it together. Now, for those of you who are very good at this, you may notice that yes, this is not the best possible photographic output, because there's blur in the water and sharp water here, but this is just to prove the point. It can really, just by doing simple two clicks here, generate a new image by merging, and but it can find the best possible seam between one image to the other. So just to show this in detail, I have two images. I generate this image, but what has best happened is that we have actually found a seam between these two images that lets me merge them. you see my problem here with blurry and sharp water.

11 - Seam Finding Using Graph Cuts.srt

How is this done? Well, simply put, what we want to do is, we have a target image and a source image. What we want to do is create a small graphical representation. And now we want to figure out from this one is, that, which pixels come from which image. So in essence, what we can do is take this small three by three grid representation and create a node structure out of it. You will add constraints to it, and this linked structure is there, but we want to make sure that there is a node on this one and another one. So this node is related to target image, this node is related to the source image, and you can represent that here also. So now we laid this out here, these are all the nodes, and between each and every node is a cost function. A cost function says how similar it is to go from here to there, and this could be dependent on, , how similar these pixels are. If these two pixels are really similar versus this one, I want to keep these two and cut this one. if these two pixels are similar over this one, I want to create a cut here. So looking at it, I find the most similar ones between those two within this neighborhood. And I choose which one to cut, this or that, this or that, or this or that. So this allows me to now come up with a way to kind of take a target image, and, , this target image is constrained to this node here, and this source image is constrained to this, so they have to come up with one or the other. The rest of them, I have to choose, and I have to look at, again, similarity between these two to chose where to cut between them. And in essence, find a path of cut through this that gives me the results. So here, for example, a cut is through this, where I cut through these two different connections and I find that these pixels are coming from the source and these pixels are coming from the target. Such minimum cost cut is, can be computed by a variety of algorithms, and the more popular one is the max-flow/min-cut algorithm that's used widely. Again, I recommend you to look at these two papers that I've cited here. This allows me to now come up with a solution that says, I'm taking pixels from target here and source here. And that allows me to generate an image like the one I showed you. Another approach is to use this kind of stuff dynamic programming, and that's also something that's available for you to look at. It's one of the cited papers in the list.

12 - Seam Carving.srt

Here I want to show you another example, where we can do same kinds of things. Except, we want to use this approach to take the scenes that are the most similar, either out or put them back in to extend the size of this image. So here the goal is, I have this image. What happens if I want to shrink it, so I want to get rid of some information in the horizontal, or I want to increase the size so the aspect ratio is much wider here. And I want to add frame. This is a method called scene carving, you can look at more details on the papers listed below. Here's how the dissolve looks like. So here I'm shrinking them, what's really happening is some of the, you know, seam is being removed. Or, when I'm stretching it out, more seams are added based on the sample space between them. And this is allow used, allows you to kind of generate a much larger aspect ratio or a small aspect ratio. These kinds of approaches are very common and also available in standard Photoshop tools like content aware fills and stuff like that. They are widely available in a variety of photo apps these days.

13 - Summary.srt

So to quickly summarize. I introduced a concept of how we can actually use cuts where we find the seams to allow us to blend images or merge images that does not require us to do blending or fading between two pixels. , the concept of seam is very valuable as it can be used to kind of generate original images or new images that actually use all of the original pixels from the source and target images. And we looked at various advantages of each one of the methods. Both of them are powerful methods and should be used, again, depending on what you want to accomplish. The effects of ghosting and blurring are much more popular with blending. Less so with any kind of seam calculations, because you're actually giving you the same exact pixels from both views. As we get into more advanced topics, I will get in the habit of also show, sharing with you a variety of papers. Here's a list of papers that I talked about a little bit. Again, they will be made available off the course website for you to look at too. And what I'm going to do next is, I'm going to get into the whole topic of detecting and finding features in images, because those are the ones that'll actually allow us to do more advanced things. And here are just some cri, you know, sources that I've been picking up information from for a variety of the topics I've discussed so far.

# 04-05 Features.txt

01 - Intro.srt

Welcome back. So far, we've been concentrating on looking at how we are going to merge a few images together by doing blending and cuts and all such things. Now let's actually focus on some of the basics. In this lecture, I'm going to introduce to you the concept of doing feature detection and matching. We're going to talk about how we're going to look for things like corners in an image, and use that to extract information that will allow us to find the same corner in two different images, to allow us to do things like matching and alignment.

02 - Lesson Objectives.srt

The specific objectives, this lesson are for you to learn about, the benefits of feature detection and matching in two images, again, given two images I want to be able to find, commonalities in these two images for a variety of applications. I'm also going to discuss, what makes a good feature. An example of a good feature in this context would be a corner, some sort of an information that defines a corner in an image that would allow us to find a repeatable match across multiple images. And we'll also talk about briefly the Harris Corner Detector Algorithm, and I will just introduce the basic concept of a SIFT detector, both the Harris Corner and SIFT detector we'll discuss in detail, in a future lecture.

03 - Recall Detection and Matching.srt

Just briefly recall again the purpose of why we want to do this kind of matching. Again, I have been given two images. And here are actually the images that you've seen throughout these lectures. And here you can actually see some of the points labeled. But you can't see them because they're not in detail. So let's actually focus on them. In this image I found one, two, three, four, and five different features and I want to be able to find these same features in another image. Again, remember, these are two different images and you can notice that again, the sign is different across these two images. The same features, again, same pixel types of information and neighborhoods that are common between the left image and the right image are now labeled here in red. I've ma, manually labeled them here, but they're actually also exactly the features that are seen across the two images. Now, exact is a definition here that we need to discuss a little bit. That would be the best possible matches of these features across from one to the other. Because, in some sense, they may actually look different now because of the view has changed. But they should be as close as possible, similar to each other. That will allow us to do matching. The matching pipeline, now, , is finding that this feature is the same as this one, and the next feature is also matched. Another one. Another one. And another one. Again, now this starts telling me that these features are both detected in these two images. And are matched. Noticing this feature one, for example, is exactly feature one here. Feature two, feature two. Feature three, feature three. Feature four, feature four. Feature five, feature five. , to do real matching you would have to have a lot of these similar features. And if you notice that in this image there are a lot more features like this also marked and indexed across the thing. Even though, , some of them may not exactly the same. Because of our idea of reasons, but you can see these four, one, two, three, four, one, two, three, four are also visible here.

04 - Image Matching.srt

Before we go on, let me talk about some of the basics of image matching here. Imagine I have an image, small image, right here, and I'm just going to put them on a axis of x and y because we want to look at how various types of transformations on this small example image could be matched or could be actually all, also determined by looking at an image. So imagine in this X and Y plane, this image now has moved over here. In a simple matter, what I'll say is that image has now translated. In essence what has changed, is it's value of this image in X and Y. And I can actually use this coordinate axis here. And in essence everything in the image is the same except all values in this image have been changed by change in x and y. And some times depending on how I want to do it I may also actually want to also use a frame of reference that is the middle of the image. But again this is just a simple transformation. Another way of looking at these images. Again, this image now could appear and again in the x y plane that we are looking at except now its also got a little bit of an additional transformation which is that now it's rotated. So it means that this image now if we were to have these two axes is now appearing here at a different location x and y, but in addition the change in x and y. There is also a additional change, which would be this angle theta. So , now we can refer to this with three different degrees of freedom, x, y, and theta. So this is a rotation of the image from here to there, with a translation. What about this third transformation of an image? Here if you notice this image is translated, moved to a different x and y, but also in the same time it has been rotated. So theta is different, but it's also smaller. We refer to this transformation as scale. So now we have four parameters x, y, theta, and scale radiation. Here is another transformation. I've moved this again in x and y. And, if you notice, the shape of this image is also kind of different. So, this is usually referred to as an affine transformation, where in addition to x, y, and the angle, and the scale, we have actually kind of looked at some sort of a simple warp that makes this image look different. Yet another transformation is shown here. This is referred to as a perspective transformation. That is, now we have taken the image and also kind of given it a little bit of a perspective warp. So, if you notice now, is things are kind of looking shorter as I go into the image this way, and this is again a transformation. So, now we have actually added another set of degrees of freedom on this image. Now, in essence, to be able to get the image matching going on. We need to also look for all of these transformations.

05 - Finding Features.srt

Now, what we are really interested in is finding within these two images with all the transformations some features that are common that would allow us to do any kind of matching. So our goal is to find points in an image that can be found in other images. Again, I've just showed you an example of this, point being found in another one. They need to found precisely, that means they need to be well localized exactly where they would be across the two images. But also, they need to be reliably found. That they need to be, again, quite well matched. If I was to take all of this neighborhood around the circle here, they need to have the same neighborhood here. So if I was to run different types of operations on both of them we'd get similar answers. So again, they need to be reliable, they need to be easily detectable.

06 - Characteristics of Good Features.srt

So what makes a good feature? Again I'm showing you two different examples of the two friends we've looked at throughout lecture so far, Einstein and Darwin. So, one more important thing we have to look for is that these features have to be repeatable and precise. We need to be able to find them again and again. So if I ran the same feature detector on this image. Today or tomorrow, or every frame between it, I should be able to get the same set of features for both of them. And this has to be repeatable, even with geometric variations in image for photometric ones, that is geometric if the image is warped or deformed a little bit, I should be able to find these features. And, also, if the lighting conditions are different. So, this repeatable, precise depiction of features needs to have invariability to both geometric and photometric variations of images. A hard task, but these are the kinds of things we need to look for. Another measure we have to also account for is the need to salient that is very specific and also be matchable. Again, we are partly building on the repeatable part. They need to actually have characteristics that I can actually match them from one image to the other. , if you look at these two images, while there are lots of features here between both of them. If you notice, the commonality of these two features is this pattern here. They were taken next to each other. These two characters were sitting next to each other. And, again, we will look at this later. the, kind of, the matchable pattern between these two images is this region and this region. So in essence, for matching on saliency and matchability, these need to be features that have distinctive description mathematically. That makes a unique aspect of these features that I can actually find in an image. Another characteristic of good features is they should be compact and easy and efficient to compute. Compact as in I can't really imagine taking each and every pixel of this image and saying, I'm going to just use each and every pixel. I need to find a few specific points and neighborhoods that I can actually now match across those two different images and I need to be able to do this efficiently. Again, efficiency could be both computational, the number of features I want to find. Another characteristic, locality. Locality here means that each feature should have a small region of support, which means that with each feature should just occupy a small part of the image, not the entire image. Which means that if I actually have small features like this here, this here, I, I can match them much more reliably and more importantly, these features would be robust to clutter and occlusion. So again, if this was occluded, and there are some features behind it because of various types of things, I would not be able to match them. If, , I had taken the whole flower as a feature, I want to be able to find smaller parts of this feature, which still allowed me to match to this. And , even the fact, that if I had taken the whole flower as a feature, it's occluded and therefore would not look good. Or at least not match across images.

07 - Identify Good Features.srt

So to help you understand more about what makes a good feature, remember what we discussed just previously, which is the various characteristics of what makes it a good feature, repeatable, compact, and having kinds of information that would allow us to match reliably across the image what are the features. So, just for us to play around with this, here's a simple example that we will play with, the three musicians image from Pablo Picasso. I'm going to, in this one, put up different boxes, and the center of those boxes is what I'm going to suggest is what you want to say are good features are not. So total there are eight boxes. One, two, three, four, five, six, seven, eight. All right, the question that I have for you is, so right in the middle of these boxes, for example here, or here, or here, or here, or here, you can consider them to be lot, rather big, to include various types of things here. Or here and here and here. Are those good features? Okay? So please, select the ones, on this image, which you can make for good features. Again, remember, there are one, two, three, four, five, six, seven, eight. Eight regions I'm asking you to look at.

08 - Identify Good Features.srt

So to answer this question, let's look at each and one, each and every one of them separately. For example this one, if you notice there is no distinguishing mark here anywhere that would allow me to kind of find this repeatable, any point here, here, here would actually have the same characteristics but this is definitely not a good feature. Similarly, here if this was the feature anywhere along this line, and with rotation anywhere along this line or this line would also match this. So this is not a very good feature. It's not easily matchable. This one, the feature is here, and any point along this line, or for example, even with rotation, anywhere along this point, and many other points on this image, this would match exactly. Not a good feature. Same problem here. it's got a line here, and anywhere on this point and with rotation around here and here, this would of match as a feature, not a good feature. This one, again, a line anywhere across this one or on the other side here, would match all of this point here, not a good feature. So, so far, we've gone ahead and negated out one, two, three, four, five of these features. Now, let's look for what would be good features. Here is a point. If you notice, this is a corner. Which is rather unique and yes, exactly this corner with blue and the kind of the beige color doesn't really repeat anywhere else. This actually would make for a good feature. This one also has kind of a corner information here. And in fact maybe would be one of the ideal ones again because just by looking at it, I don’t find this picture again, in this locality here, a good feature. This one also has a bit of repeating pattern. a very unique corner makes for a good feature. So these three are good. These, one, two, three, four, five are bad features. Let’s look into, in more detail of what’s going on here

09 - Find Corners.srt

So, find corners is one of the biggest things we want to do. Here we have two images, and I want to be able to find features in this one that are kind of, have repeatable nature. And that would actually mean, let's start looking for corners. So the key property that we want to look for is in the region and around a corner, image gradient has two or more dominant directions. I'll explain what that means in a minute. And also, the basic thing is, corners are repeatable and distinctive within a local region of an image. Why is that the case? Let's demonstrate by a simple set of examples. Let's look at this simple image. white with a corner and a black region here. Let's just look at different differential information just in the region right here. I moved this region by a local neighborhood just by a bit. And since it's black, by looking at the differential in the window that I just moved here, there is no change in direction of the gradient intensity in those windows that I just moved in. So a flat region, there is no corner. There is no feature and therefore there are no change in directions. It's something which I cannot use as a feature. Let's look at this. Now I'm going to do the same thing which I did with this one. Except now I'm going to do it at an edge between the white and black. If I just move this window briefly, and as you notice I just did that. Again, I'm moving this window, repeating it again. In this case, yes, there is a gradient change, but there is only change when I move from white to black. But there is no change when I take this window and move it up and down the edge. So, no change along the edge direction. But, yes, a lot of change in this direction but not in this direction. So that doesn't allow me to kind of have the characteristic I want. We looked at in the previous example of the Picasso image and we noticed that, too. In this case, let me do the same, but now I'm going to move this detector. And move it in both in x and y around the corner. If you notice, in this corner, changes are in all directions. Just looking at it again, the changes, any x change, any y change the entire, the gradient changes would happen. So in essence, we recognize a point by looking through a small window, which is what we're doing here, and shifting the window in any direction causes a large change in intensity as I move it around. And it's that large change in intensity is what I really what to be able to look at.

10 - Corner Detection The Basics.srt

So in essence, if you were to look at a simple example like this, if I have a signal that goes in, changes directions like this, I'm going to be able to pick up these changes, so this is yet another way of looking at the same example. If I had a flat region, if I move my window around in this flat region, there is no change in any direction. If I was to do this at an edge. This changes, in direction this way, but not along the edge, but, even a slight change, in any direction around this corner, will actually give me information of changes in all directions, and therefore, this makes this point a very, very significantly important feature that has the characteristics we discussed, while, there's nothing here nor here that we can use.

11 - Corner Detection Mathematics - I.srt

Let's start looking at this with a little bit of an eye for the mathematics. What we're interested in is computing the change in appearance by shifting the window. And we're shifting the window in two different directions, u and v. So this is my measure that I want to look at. This is my image. what I do with this image is I move it by u and v in different directions. And subtracted by the original and look over some squared differences or the difference in the square of it. , to help us look at this carefully, we'll also create a small function, a box function like this. Which , any time I'm with in this, I get the answer one. Outside I get 0. Or I can put a Gaussian curve like this. So, in essence this is my intensity function. This is my shifted intensity. Again, recall all the kinds of stuff we've done with intensities. And how we've looked at the neighborhoods of it in all of our work on convolutions and correlations and filtering. And this is the window function, how big the window is and how actually want to keep the size of the window impacting what I'm looking at. And this is my function, which captures the change in appearance. Let me actually showcase a little bit of how we can compute this by the simple example. What we want to do, is compute the change in appearance by shifting a window by you know, it's small u and small v. Here's a simple image, and here I've shown you a small window within this that I won't actually compute information on. So here is the response function E u, v for this one. Computed by using an equation like this. just assume that we picked up one w x y to allow us to do this. And the value, , here is E 0, 0. And this is the window that I have. And , I can now compute the whole E u, v, for that window u, v. I can also compute the E three, two for this win, pixel here. Which is three down and two up, or three sideways and two up. Which allows me to kind of look at the whole function which would be this part here. And , that results in an additional window, like this. This allows me to do the computation I want. remember the tools we looked at again, that lets you do neighborhood calculations on images. And in essence, this allows us to do that kind of stuff. Looking again this corner mathematics, let's build on this concept a bit more. So let's take this equation. And take a quadratic approximation of it using Taylor expansion. And the result is something like this. An approximation where E u v now is written in a matrix form with u and v, M u and v transpose. So what is M? M is the second moment matrix computed from the image derivatives, Ix and Iy. Remember how to compute I x and I y? We have done this before when we looked at derivatives of images. So a second normal matrix is a matrix derived from the gradient of a function. And in essence, it summarizes the predominant directions of the gradient in a specified neighborhood point. Then again, repeating, it summarizes the predominant directions of the gradient in a specified neighborhood of the point. And the degree of this, what happens with this, is the degree to which the directions are coherent. Allow you to say something specific about an image or a feature. In essence what that means is M, again with the weighted window here, is the x derivative twice squared of that one, x and y derivatives and x and y. Well the, the forms of the off diagonal terms, and , the square of y would be here. Visually what this means is that if I had an image like this, for any point here, I'm looking for, two axes. And in essence, what I'm looking for is the maximal directions that allow me to kind of encode the information that makes this feature unique. So in essence, at the gradient of the function, it gives you the predominant directions of the gradient in a specific point. And we compute this using this methodology here, where we compute the Ix and Iy's. And use this to create a matrix. And it starts giving me information about how the gradient directions are dominant for this image at this point. Remember . We'll come back to this in a second.

12 - Corner Detection Mathematics - II.srt

So let's continue to build on the corner detection mathematics. We looked at this M matrix. , we now put this M matrix in our E u,v term and expand it out and we come up with this nice quadratic form here. So, , the surface UV is a locally approximated by a quadratic form here with a window here. Let's look at this car, equation a little bit more carefully. So as I said, this surface is locally approximated by a quadratic form, and have just shown you, am showing you here, the surface. Now, this quadratic here is a slice through this surface, which is this equation. And this equation is that of an ellipse. And you can notice this ellipse is right here. So this ellipse are these ellipses here, and these are the slices to the surface. Drawing them here, it's hard to see them because they get complicated. But you'll notice each one of them, these slices, is an ellipse. So now we can actually look at each one of these slices as a single ellipse. So again, this is my equation of the ellipse and in the surface and these lines here on my ellipses. Let's draw one ellipse out to just kind of see what we can measure out of this. Remember where we were looking at, at any point the directions of changes at any point and this direction and the magnitude of these change that will allow us to kind of quantify at any point the strength of this feature that I can actually use as a repeatable measure across two images. Right? So, this is my center. In this direction there is a change going on, from here to there. It's a rather fast change. I mean, the same magnitude is covered in over a short distance. So in essence, we can start labeling this as maximum change. So direction of fastest change is this direction in this ellipse. And another one would be in this axis. And we can refer to this as lambda min. And this is direction of the slowest change. Now, this is a very standard linear algebra trick. Where when we have surfaces like this, and we have ellipses. We can actually now compute the lambdas and also the orientation. Notice again that while these are the values of lambdas, this ellipse is rotated by certain angle from the original image at this point. So, this is where we refer back to eigenvalue analysis where axis lengths, the lambda values, are eigenvalues. And the orientation is when you take these two vectors and compare them to the normal axis or frame of reference you know actually the angles of these two vectors. So this allows us to now start quantifying the direction and the quantity of change that goes on at any feature point in an image. Help us simplify this, now let's actually still continue to come up with newer ways. Remember, we did come up with an M matrix here. We can actually still go back and compute the M matrix this way. This was the quadratic form we had looked at before. And M here is a diagonal matrix which has nothing else but R here and the two diagonals are the two eigenvalues, lambda1 and lambda2. Let's talk about Rs in a bit because these are all methods we've come up with to help us make an efficient set of calculations to compute what we are interested in.

13 - Eigenvalues.srt

So let's look at this matrix and try to expand on it. This is R, which is another matrix which is the determinant of M minus alpha trace of M and actually is this set of values here of lambda1, lambda2 minus alpha lambda1 and 2 and alpha is a constant that we go from 0.04 to 0.06. By varying lambda one and lambda two, just to help us kind of do this. This is an example of what we can come up with. And this will start telling us what kind of feature we have. R here now depends entirely on the eigenvalues of M. Okay? If you notice that's the case here. What we interested in is looking at different things. R is large for a corner. For example, right here. Anytime the right values of lambda one and lambda two are large, the magnitude of this would be larger. R is negative with large mag, magnitude for an edge. So if the edge is like this, you're going to get more information like this. Magnitude of R is small flat region shown here. So in essence I'm showing you the whole landscape. R is large when you are there, R is small here in the flat region. R has got different variations of lambda one and lambda two, our ellipses are narrow, that means you most probably have a niche. And in essence, what is important to note here is the way we did this we do not have to even do explicit computation of Eigenvalues. This is great because the kind of computation we want to do we want to do very fast. And actually even in the early days of computer vision types of techniques from late 80s and early 90s this was something you could do very quickly on a small computer. And that's what you want to do. You want to be able to do corner detection quite efficiently.

14 - Harris Detector Algorithm.srt

Let me just quickly now preview Harris Detector Algorithm. And I'll also talk about the feature the other type of featured detection methods, we'll cover them in detail in the next set of lectures. So what you do for Harris Detector is we want to compute the Gaussian derivatives at each pixel. Remember again, we have learned how to do Gaussian derivatives. Then we want to compute the second moment matrix M using the ix and iys in a Gaussian window around each pixel. Then we want to compute the corner response function R. Again, we know how to compute R once we know M. We want to threshold R a little bit because you want to kind of have a, you know, a lead as to how many corners we want to protect and then we want to find a local maxima of response function. I have a doc doing any kind of mon, maximum suppression. Now what that means is now I can take an image like this, and I would find a lot of features which are Harris detector features. Again, we will cover this in detail in a future lecture.

15 - Properties of the Harris Detector.srt

So what are the good properties of a Harris Detector? One, a Harris Detector is rotation invariant. In this case, what that means is that ellipses rotate, but the shape and the eigenvalues remain the same, if this image is rotated. And therefore, corner response function R is invariant to rotations. So in essence, I could take the same thing, rotate it, it'll find the same Harris Corners as it did when, you know, the image was not rotated. Intensity invariant which means is that if the intensity of the image changes because of photometric variations and stuff it's actually not sensitive to it. So it's partially invariance additive and melt, multiplicative intensity changes because of thresholding that we can do with this. And it actually supports us being able to deal with highly intense images. Primarily also again, because we're only looking for derivative information, that is the derivative change in intensity not just the absolute one. So in essence, again, you notice, same image. Now I've increased the intensity but find the same features. Finally, scaling invariant. it's dependent on the size of the window we use for matching. , we can also use different types of frequency domain analysis using pyramids to help us do matching of that type.

16 - Scale Invariant Detectors.srt

So let's look into a little bit more carefully the scale invariant features that we are interested in. The Harris-Laplacian is one type of scale invariant feature. , it finds local maximum of Harris corner detection in space and Laplacian in scale. What that means is, I have x and y, we've got the translations or transformations scale. In essence, if I have something in space here, this feature could also be measured in different scales. Laplacians, again, remember all of our work on Laplacian pyramids and Gaussian pyramids that suggests how we could do this kind of stuff. So, in essence, we can actually find the feature in different scales. Which means that, now, the same feature, if you zoom into an image, the feature got bigger, we should be able to detect it with this approach. So, in essence, the Harris space is here, and the Laplacian is in scale. So Harris is x and y, and Laplacian is in scale. Another scale invariant detector is SIFT, and we'll cover this again in a lot more detail, but I just want to introduce this very briefly here. It attempts to find local maximum of difference of Gaussians in space and scale. And the difference of Gaussian is simply a pyramid of differences of Gaussian within each octave. So this accounts for our idea of different things that are in the frequency domain, and, again, we're looking for these difference of Gaussians in both space and scale. Again, looking at our simple example, our simple schematic here, looking at variations in space. But when we go up in scale, we can actually start looking for information. The difference of Gaussians, both in x and y, and also difference in Gaussians in scale. Again, we'll cover this a little bit more in detail in a later lecture. So SIFT in essence refers to a scale-invariant feature transform, which allows us to look for changes in orientation, and allows us to find features within it. So it allows for us to compute the best orientation for each keypoint region in an image. And also captures keypoint description, where we use local image gradients and selected scale, and also look for rotation to describe each keypoint region.

17 - Invariant Local Features.srt

Here's an example of what SIFT would do, where we're looking for different types of features. Here is a car, again, in a completely different location, a different image. And if you look at it, it found this region, which looks like this, which is again found again with the rotation of the image and viewpoint scale and everything here. Similarly, you can see that these are the five features it found in this one, and they were matchable in both of them. So, in essence, this allows us to look for features, even with a lot of different transformations, of translation rotation scale and other imaging parameters. if I find this, I can match them across the two images.

18 - Results.srt

We started off this whole lecture by showing you two examples of my two friends, let's look at what happens with those two images. Here, with the two images, while they found a lot of features, we detected a lot of features in these two images. But, we really wanted to find matchable features between these two, so, the red here points out the features that are common between both of them, and once I have those, what I can do is, match them, and that allows me to now, bring those two images together, and here I'm showing you that both those images now match, and, , I could use this, to now create a panorama between those two images, because what I , do is, align those two pixels, and do a little bit of warping that we've talked about before to allow us to do this kind of stuff.

19 - Summary.srt

So, to summarize, I've introduced the whole concept of feature detection that could be used for matching between two images. I discussed variety of characteristics, four specifically, of what makes a good feature. Briefly introduced Harris Corner Detector and SIFT, again, some things we will cover in detail a lot more, in the future. Further reading is available on the website. But you can look at them here. And these are a variety of papers that have existed. These are a variety of papers that I've come across in the area of computer vision, for doing feature detection. Includes Sift, and also for example, Harris Corners. Also I encourage you to look at descriptions like features and open CV or Matlab sites. Further lectures will cover more details on SIFT and how we can apply them to various types of image alignment and image matching approaches. Again, thanks to a variety sources I picked up information from. And we'll continue discussing these types of concepts in future lectures. Thank you.

# 04-06 Feature Detection and Matching.txt

01 - Introduction.srt

In the last lecture we spent a lot of time looking at how we're going to reliably find corners in multiple images that'll allow us to do feature detection. We also introduced the concept of SIFT and Harris detection algorithms. In this lecture I want to actually dive deeper. Try to get to understanding of how does SIFT, and also Harris detector algorithms work, and how are they actually made, both illumination scale and rotation invariant?

02 - Lesson Objectives.srt

So the specific objectives of this lesson are for us to dive deeper into the Harris corner detector algorithm. Now, we have already looked at it briefly. Here we are going to talk about the various steps that go into making the Harris corner detector algorithm, also look at a variety of differences and variants on the basic algorithm, that'll help us understand how we can do feature detection. In addition, I will also describe the more widely used SIFT algorithm, which is actually pretty much used on most software that actually do any kind of feature deduction and matching. Just to recall, our goal is to start off with two images and our interest is to find features in these images, like these. And then also, find features in another image in an image pair. And then match them, saying this is the same feature as that one. So finding the similar features across two images is our goal. And to do this, we have been spending a lot of time, actually from the earliest lectures, trying to learn how we can get inside and find repeatable features that actually can be detected more reliably across images, even though the images might be changed. In this case, the change is subtle. Just a pan from one view to the other.

03 - Corner Detection Mathematics.srt

Now, recall in the previous lecture, I talked about the whole process of how we think corners are an important type of a feature. And we actually looked at this entire equation, that gave us a way of looking over an image by shifting it and using that shifting to be able to find a corner. these were shifting the gradients. Being able to move an image or move a region over and seeing if there's a change in different directions, and we computed all kinds of information to support it. So use it in this equation for corner detection. Just leverage the quadratic approximation, using the Taylor Expansion which gives us another equation. Which results in this simpler form, and again it's not in a matrix and vector form. And M is the matrix that we now actually want to compute, because that'll allow us to do the computations that we wanted to do here. Just recall again, M is the second moment matrix, computed from an image using derivatives Ix and Iy. So again, we compute the derivatives of an image and using that we can actually now compute a moment matrix. And that allows us to now do this computation in a much simpler manner.

04 - M Matrix.srt

So, now, let's spend a little bit of time trying to get an intuition of what this M matrix is. So, the best way to think about what an image moment is, an image moment is a particularly weighted average of image pixel intensities, and it could also be a function of those intensities, and it's used to give us certain specific property at a point within an image. So, any image, shown by this rectangle here, at any point I want to be able to find some properties that are kind of descriptive and something I can actually find about an image. Now , what we will leverage here is information driven by the derivatives at that point. So the derivatives, , can be measured using various types of things we've looked at and from that, I can find a gradient vector, the magnitude of that, and the direction. The second moment matrix is specifically a matrix derived from the gradient of this function that could be actually at this point. And it summarizes the predominant directions of the gradient of a specified neighborhood of that point. So around that point, now, gives you more information about the direction and the magnitude. One additional thing that's important about the second moment matrix is it also gives you a little bit more information about the coherency of the directions in that neighborhood. So let's look at the equation of how we would compute the second moment matrix. again, it is the intensities, sorry, the derivative of the intensities in X and Y direction, and then the mixtures of these X and Y's on the diagonals or off-diagonals, and computed over any neighborhood point. And the neighborhood search is this region here. So again, simply put, if I had an image like this I can use a second Loman matrix to at any point in the neighborhood around that point find description using the gradients that is unique to this point, and actually I want to use this to understand more about what's happening in the region and that could be used to now compute the information about if there's a corner and what kinds of parameters associated with that corner in that image. Think the note is, and we've talked about this, is the importance of the ellipse and the argon values. Again, something that we've looked at before. , when you have certain bits of information, if you think about it this way, with the moments it's kind of giving you information what's the best possible rotation around these two different axes. And, , we use this to define two major axes. Or, sorry, one, for this ellipse a minor and the major axes. And this kind of gives us what's the most dominant direction for this region. And that's what we're interested in here, is at any point in an image, finding these types of ellipsoids, and using these ellipsoids to find the dominant directions. And, , recall that item values provide us with this information.

05 - Recall Harris Corner Response Function.srt

So using this, we came up with this whole description of how to look for corner or responses of corners in an image. And we looked at this diagram. this is showing you in the two different eigenvalues, that if the eigenvalue 1 is larger, the eigenvalue 2 is smaller. This is the kind of shape you get. If both of them are smaller this is what you get. If eigenvalue second one is larger, this is the shape you get. And , when both of them are larger, this is the shape again. this is the corner. So we're looking for mostly corners. This would be an edge, this would be an edge, and this would just be a threshold point. We also looked at details of how we can actually now just if we know the second moment matrix, we can actually compute all of this from the straight from the second moment matrix. And we actually did look at the determinant of this matrix and trace and actually just comes after the simple calculation. And we use a small weight factor, a constant to help us do this calculation. So if you notice here where R, which is giving us this information about what is a corner or not, is only dependent on the eigenvalues of the second moment matrix, making it larger when it's a corner. Negative with large magnitude for an edge and small. The magnitude of the R would be small and that’s where this region would be. So again, we can do all of this without any explicit computations of the eigenvalues because we know how to compute the M straight from the image, by just doing the derivatives and then using the gradient computation to help us get there. Which actually is an efficient way of doing these kind of things, and again this is one of the reasons this kind of stuff can be done quite efficiently in most simple computers. Again, remember, this was the equation that now we just have to compute, and we can actually do this on a neighborhood with just image intensities, and the derivatives of image intensities.

06 - Harris Detector Step by Step.srt

So let's look at what the Harris Detector algorithm looks like in each and every step. So, first step, we want to do for Harris Corner Detection is compute the horizontal and vertical derivatives of the image. And, again, if you recall, we have learned how to do this in an earlier lecture by using, , how to compute derivatives, and also doing to get towards computations towards things like gradient images. And another thing we want to do is we want to actually now compute information from the gradients to compute M. Then, we, actually, kind of want to smooth it out a little bit, and that's why we take a larger Gaussian image and take this image that we get and smooth it out a little bit, because we want to kind of get more information out of the signal here. Then, we actually start measuring scalar interest measure. And I encourage you to look at more details on where are the books that could be available and the material that are make available on this one. And then, we want to find maximum value above some threshold, and those would be the features. So, this is a step by step method on how we can actually do Harris Corner Detection. Let's look at more details on this one.

07 - Harris Detector Workflow.srt

So here, let's actually build this by simply using the two image pairs. Now, if you notice in these two image pairs, the object of interest has significantly changed, but some of the local details are, , the same. Notice the same toy giraffe, and we can see some of the details, even though, , some of the details are significantly different. Also, in this case, just to prove a point, that the intensities are also different. First, , by running through it, we now want to come and compute the color response function as we've looked at this image. This starts giving us all of these responses, and I've just colored, I mapped them a little bit to point out where the higher cost-functions are, which are all of the bright red ones, that's where most of the interesting corners have been found in this image. they are found in both these image pairs. , now we can actually take those bright spots, and find points with the largest corner response. In essence, now that says, I know what the R value is, I'm going to threshold out and get rid of everything else that's below a certain threshold. That allows me to find these regions again in both of them that are just a higher rate, or higher intensity, of Rs. This image should be hard for you to see because the points are now very small in here. Let's start to zoom in. But , this is finding the local maximum of each and every point that we have looked at. at this scale, they get to be really small. Let's zoom in. And if you zoom in, you should be able to see some simple white spots all over this image. Don't fret if you don't see them, because what I've done now is just use contrast announcement and now you can see these points here. , they're still kind of hard to see. So let's look at them overlaid. So, in this case now I've taken the same image again, pair, and here you can see small red spots here. , some are found here. That region is not there, so, , they're not found here. But you can see, for example, there are some features at the nose. There are some features at the tips up there. And you can see similar features, and actually, now you can start seeing that it found features here and here, and also found features, for example, here and there, even though the image is a little darker. So this is the output we're looking at. We want to find features like this. That's what we want to do with Harris detection.

08 - Harris Detector Algorithm.srt

So again now, let's look at the whole approach again. And the same steps but different details. Compute the Gaussian derivatives at each pixel, for each and every image, that's what we want to do. Second thing we want to do is compute the second moment matrix in a Gaussian window around each pixel. Again, find a pixel and going through this whole image, find a small region and compute the second moment matrix, because now we have the derivatives. We know how to do this. And again we want to compute this within a window. Then using that, remember how we can actually take R, or compute R from M, by looking at it's trace and it's determinant, we can compute R. We want to do a little bit of thresholding on R, because we don't want to just completely live off the details that we had. Remember we showed you that with the example image two. The bright white spots will convert into small, white dots which were hard to see. Then finally find a local maxima of the response function by using a non-maximum suppression. And that now starts giving us features. Now we have found features in two images, which we now can be using to match between those two images.

09 - Harris Detector Quiz.srt

So now let me actually introduce you to a fun quiz we're going to do with Harris Detectors. And you'll get experience of actually using some code online to be able to kind of see how we can actually do Harris Detectors on an image. To do this quiz, you'll have to work back to the browser where, , you've been getting to see all of your lectures, and go to, and this site, and go to Sandbox. , whenever you are in the sandbox some code should be existing there that's been working on before. And in this case I have something for feature matching. So, for this quiz, what I want you to do is look at this code and run it. Let me show you what's happening here, this is the code that actually does Harris corner detection. And our goal at this quiz is to take piece of code that is there and find this image. And in this image, find the corners. , all of the processing steps are shown, and here you can see that it did find corners, but not all of them. So one, two, three, four, five. It missed these two. So what I want you folks to do is, first, go ahead and find the documentation for this function. And here, , in this function in the code, change the parameters that all eight corners are detected. Then you hit the Submit. The code will be evaluated against two different images and you'll get an answer. So that's what I want you to for this assignment, or this quiz, please.

10 - Harris Detector Quiz.srt

Well hopefully you had luck trying to do this. Let's see what we need to do to make this work. So this is the function that we need to do some things with. So if you were to play around with this. Let's just change this value to four and five, and run. And now let's look at the results. Once again it shows you all of the debugging parts, and all of the corners of this hexagon are now computed and actually highlighted by these nice visuals. So this is just to kind of get you familiar a bit, even the coding part of it and actually try something interesting in the space.

11 - Harris Detector Some Properties 1.srt

Now, I want to cover some basic properties of the Harris Detector. One of the bigger questions is, can we actually build these types of detectors, the invariant to rotation? That means is, let's say, if I have one image and then I have another image, and then this image , let's say I have something like a plus sign here. I want to be able to detect the features of this even though, in this image, this plus sign might be rotated a bit. Right. So being able to deal with rotation invariants is an important part, because I still want to find the same corners. I have a simple example like this, but you can actually see that this can get to be very complicated, depending on objects. And , I sometimes want to be able to find objects. Even with variations across the board, on lighting and stuff like that. So that's the next part. Can we actually also find edges if there is variance in image intensity? So again, I might have two images, and these could be two pairs next to each other. This one is much brighter, and this one is much darker. We saw examples of that when we looked at the image pairs of the toy giraffe, for example. Third part that we are actually be invariant to is Emmet scale. What that means is, I have an image and I have an object that is here and it is of this scale, but on the second image I have the same object but it is, in this case, much bigger. So, how do we kind of deal with the fact that the same object might be occurring in an image pair, but appear to be very large? So, in essence, now we need to start seeing, does the Harris Detector provide support for these type of invariances?

12 - Harris Detector Some Properties 2.srt

Let's first look at Invariant to Rotation. Here's a simple example again. Let's look at this. And I have the same corner in this instance is appearing in another image with a rotation. Now in this case, if we were to go through the whole process, and computing the details that would actually let us do the ellipse, which would have the eigenvalues and stuff like that. Again, we're not doing this directly, but we actually look at second movements, which allows us to get similar information. For this point and this point, the eigenvalues should be the same. The size of the ellipse that we're looking at should be the same, but what should not be, is , the rotation. So it's orientation has changed from this to this, but the eigenvalues remain the same. Since r is only dependent on the eigenvalues of the ellipse, it suggests that the corner response function, r, is invariant to image rotation. So, this object can be rotated completely, as long as these values are the same. We should still get the same r function, and with threshold and stuff, we'll still get the same point. And in fact, that's what would suggest that these two corners of this image would be identified in both the pairs.

13 - Harris Detector Some Properties 3.srt

Let's look at the example of image intensity. Now most of the time, again, if you remember our earlier work on looking at image intensities in images, mostly, if in fact, an image intensity changes because we have more bright pixels or less bright pixels. So in essence, mostly this would require us to be invariant to additive and multiplicative changes in image intensity. And most of the time the competition we're looking at, since it's dependent on the gradient information, would be invariant as long as at any point within the neighborhood, irrespective of the brightness at that point, the invariance or the gradients at that point are the same. So again, essentially what happens at any point in an image, as long as we are working with derivatives, we should be able to kind of compute the information, the response function, response function's purely being dependent on the derivatives, would not care if the image intensity of the region that we're looking at is much larger from one to the other. So in essence, what we're just repeating here is, it would be invariance to any kind of intensity shift, if you add a constant derivatives actually still won't care. And similarly, if there was a scale again the derivatives would not be impacted, and all of that actually still same results would appear. And this is demonstrated by again, our example. I look at the response function r, and depends on where our threshold, right. Because if my threshold at that level, if these two points are the ones that I'm looking for, I'll still detect them. But if I choose a threshold and now somebody has scaled my intensity variations, which will impact this. Then it might actually cause a problem. So then we would have to have an adaptive threshold that knows more about the scaling of the intensities in an image. But otherwise we still should be able to find, because the shape of the response one, response over the whole sequence should still be the same.

14 - Harris Detector Some Properties 4.srt

Now let's look at a third property. Is it invariant to image scale? That is, now the object has different scales in the multiple images. This can be looked at by again just taking a simple example like this. So I have two objects here or two elements. As you can notice, both of them are the same except this is much smaller and this is much larger. Now, here what becomes interesting is what resolution am I looking at information. So, if for example, if I'm looking, this is the size of the window I'm looking at and this is the size of the window I'm looking at, well nothing directly would compare to those. And, depending again what aspects of these regions am I kind of scaling and observing, I would have problems with this. This, , would be a perfect match at just this size. And there is no similarity at this window for this whole region. But as I said, this is the same object just scaled up. So what I would need is a different size. This in this detector will be a corner. This will not be at this window size detected as a corner. In fact, they will all be listed as edges, as you would expect. So this suggests that Harris Detector is not invariant to image scale. So what can we do about it?

15 - Scale Invariant Detection.srt

So, now let's look at what methods we can come up that actually makes our future detection invariant to scale. Let's take the same examples of these two shapes. Again, you can tell that they are the same, just scaled up. And , I kind of look at this window. It says that this is an edge. , when I look at this window, it comes across as a corner. So one thing I can do now is start looking at different regions. So, , consider this region a circle, and they start looking at different sizes around the same point. So, in essence, what I'm doing now is I'm zooming out. I'm kind of wanting to look at the same region but from farther away, which actually would start making this object start smaller. So I try out another region, a circle. Well no, that doesn't work. Let me try out, at the same point, another larger one. And if I keep going and let's say I get to this scale, all of a sudden, this detector is going to fire this to be a point. That's a corner, just like this. So in essence, by scaling up my detector regions, I can actually accomplish exactly what this one is doing and in essence what that comes out to is now I would actually have scaling variant matching. So, in this basic idea here is the region of corresponding sizes will look the same in both images, as long as I can come up with that scale. So this does require a little bit of, kind of coming up with the process that lets me find the right size of the new region that would let me compare this scale to this one.

16 - Scale Invariant Detection 2.srt

, the hard part now is I do have two different regions. How do I actually figure out what scale to look at? This looks good for finding an edge here, doesn't work for this one. I can keep growing it. Oh, at this point, it does. Plus, , it's possible, and a computationally and interesting challenge if I have to do this a lot. The problem is how do we choose a corresponding circles for both of them? And that could be computationally a challenging task. We could do brute force search methods and stuff like that. But remember there's another problem associated with it. Not just the fact that I need to do this, but the fact is that this, the region could have other bits of information that is not there. So in essence me matching this to this, I don't have the full picture there. I just have the full picture at this point. So we need to kind of start thinking about what kind of a match it would be also. Again, if I now start kind of zooming out of this one let's see what happens. , all of a sudden it was only a match at this scale, not at the other scale when I zoom out. So again, it's not guaranteed. I can scale out on this one. It matches. But when I scale out, no this doesn't match anymore. So , you have to choose the scale of the best corner and stick with it.

17 - Scale Invariant Detection 3.srt

So, we'll need to identify our region. Again, I'm using these regions of circles, which is scale invariant. A property that this region would have to have is that it should not be affected by the size and will be the same for all corresponding regions. To build on this, let's actually kind of think about a simple example. This simple example says that average intensities are interesting things to observe. For any corresponding regions, even if the object is of different size, the average intensity, the image pixel intensity in that region, will be the same. So while we zoom in and out, we need to start looking at the fact that if I do have a region in both of those parts, the average intensity of that region, if it's the same object, should be the same. Because as I said, it's the same object, just viewed at different scales. We resolve this, by again computing the intensities at different scales at a point. Compute the scale invariant function at different sizes and for different neighborhoods. And then choose a scale for each image at which the function is maximum. Let's see what that can be. So again, this is the function that I'm looking at. This is my intensity. And I'm going to scale up the region size and see where I actually find a certain peak. this could be done for Image 1 this way. And now I have an image that's half the size of the object and an image that's half the size. Again, if I was to take the region size and measure the function that changes, it'll actually have the same peak, but it'll actually be doing it much faster just because the image is at half scale. So just now start seeing is that the peak and stuff like that for this function are the same, just how it impacts with the region size. So we need to start thinking about where we can find functions like this. , this kind of starts telling us that we can find, again the scale here is important and the scale for this one would be, these are the two scales. If I know that, I should be able to now compute where in which region, which size of the circle do I want to use to look at an image for detection of features.

18 - Scale Invariant Detection 4.srt

Couple of other things are looking at the previous example that I just showed you that we should make concrete. A good function for steel detection has one stable, sharp beat. For example, again the same function that we looked at, f. And if I was actually to trace this out, this is not a peak here. This would be problematic. Here it has multiple peaks, would be problematic because I wouldn't know which scale to choose. This is the kind of behavior you want. You want to be able to kind of see as the region size gets bigger, this function gives me one peak. Again, some of the ideas here suggest that we can actually do this for most normal images. A good function would be the one that responds to the contrast or intensity changes in an image. I'm aware all of this still sounds a little abstract, so let's look at a few examples to make this a bit concrete. Here is an example of the same building at two different scales. In the case, this is an example of a zoomed up image, and now zoomed out. Here look at the scale and we look at the function. This a nice peak here. This is the one we want to look at. And at the same scale level, this is visible at this point. So in essence, now we need the two scales for these two images. This allows us to look at scale, and actually model the scale that will allow us to make our detector invariant to scale

19 - Key Point Localization in Space.srt

So, how do we do this? Now, this will relate back to the concepts we had looked at when we looked at the frequency domain of an image. That is, how do we actually start representing things at a different frequency spectrum and , we played around with things like the pyramid. So, to achieve this, we would have to find in the scale and the pyramids of images, the most amount of information in both space and in scale. Space in this case would be the image itself. Scale is, again, the pyramid levels for example. Being able to find information at different frequency spectrum differently. So one of the more important methods that's actually widely used is, as I said, SIFT, Scale Invariant Feature Transform. For this process, what is suggested is that use the pyramid to find the maximum values. Again, we use this kind of stuff to do things like edge detection. And in this case, then eliminate the edges and pick only the corners. So, run the same process that we've been doing before, except that now find all edges using different frequencies and then eliminate the edges and pick only the corners. So again, the same concept that we've been doing for corner detection, but except we do it at multiple scales. We find edges at multiple scales. And then we kind of say, look here. Which one of them are corners at each and every one of the scales? So in essence, what happens now is that each point is compared to its eight neighbors, shown here. So each point is compared to its eight neighbors. And then we can also scale it up and down. And in this case, each point is also compared to its nine neighbors above and below in scale. So eight local neighbors at this point, then above, nine points, below, nine points. And these are different scales which could be computed using, again, things like pyramids and stuff like that we've looked at.

20 - Scale Space Processed One Octave at a Time.srt

So what we now do in these types of processes is kind of start looking at scales at different octaves. And this is one of the reasons we looked at the whole concept of doing this kind of analysis at different pyramid levels, because each one of them is a different octave. And now we can actually start modeling the signal, and now we can actually start doing all of the processing and what comes out is the different scales we can actually have these Gaussians, and stuff like that, and using these Gaussians, we can also start looking at differences of Gaussians. Again, very similar concept to what we have done when we did some of the blending using the frequency domain. , look at these types of things, you can see is that we now actually can do the pyramids and start looking at information at different scales. This shows us how we can actually do the different Gaussians that I was showing you before between different scales. And using that, we can find the extrema points, which are all these examples here. , what we want to do now is find a few specific extrema points. So, , we would now kind of find all the points that have high contrast and that starts giving me more features. There's a lot more features here and, , now I've started reducing them. And then I get rid of all of that are not edges, just the corner features and I get even less number of points. And this starts kind of giving me the features that I want.

21 - Scale Invariant Detectors.srt

So, once again let's go back and look and detail the scale invariant detectors. Let's first look at the Harris-Laplacian. Again, I've given a reference for those of you who want to read more about it. , what is does is it finds a local maximum of Harris corner detector in space, in a Laplacian scale. Remember, we've looked at how to do Laplacians already. So here is the best way to kind of visually see it. Scale is shown up this way, space is x, y, that's where the images are. Scale kind of captures the frequency in the octaves. First what I do, is I run a corner detector in space in the image coordinates, but then I kind of move up in scale by looking for the same features. Using the Laplacians this time, as I go from one scale to the other. And that allows me to find now at these different scales, the feature does have the information I want, if one scale or the other. So just to reiterate, in the scale which is up here and then space which is down here. In space, we are actually just running a standard Harris corner detector, but then we're using a Laplacian in the scale space here. So Laplacian in this, Harris in this gives us the Harris-Laplacian detector. Again, a widely used detector and image processing computer vision and computation photography.

22 - Scale Invariant Detectors 2.srt

So now let me bring up the whole famous approach of using SIFT for doing feature detection, scaling the features, by David Lowe. Reference is also there if you want to look it up. It's also on the website. what we're interested in is finding the local maximum, both in scale and in space. the measurements we want to do is difference of Gaussians. I already showed you before. And difference of Gaussians creates a pyramid of the difference of Gaussians within each object. So it captures the scale. By looking at the derivative information of the difference in variation of Gaussians across scale and also space. So here, coming up again we have space, x y, scale going up. We again do the same processing, but again for each and every aspect of it we're going to be now doing difference of Gaussians. So in this case the difference of Gaussians in x and y for each and every image, but also doing the same for difference of Gaussians in scale.

23 - SIFT.srt

So I just want to quickly also now just review the whole process of how scale-invariant features are computed SIFTs. First we want to find the orientation, compute the test orientation for each keypoint region. And then we want to actually use the method to kind of find the keypoints itself and we use local image gradients at selected scale, and rotation to describe each keypoint region. Here's an example of invariant local features. I mean, here are two examples of the same car. So here for example, this region here is matched to this region here. It's the same car, but , we notice this patch is exactly the same even with orientation and all that kind of stuff. And it finds others regions like the similar. This region here is matched to this. This part here is matched, well it's available here but there's no match here because these two here are either too small or not visible. The bigger patch here again, has the same characteristics on both. So one of the biggest advantage of SIFT is that image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters. And therefore allows us to take two pairs of images with similar appearances of objects and stuff like that. And it starts finding the features that might have seen and do different things, and that allows us to do the matching. And that becomes an important part. So let's say I have these two images. First, I can run the whole process again to detect all the features. Here it found all the features in these two images. This is the common regions. These are two images next to each other. Now , what I after I have found the features, I want to match them. So here , it find and matches that these are similar between these two. And once I match them, I can also then register them together. And if I registered, then I can use that to align the two images together, which I'm not actually showing. I'm just showing that these two features are now matches between these two images.

24 - Feature Matching Demo.srt

Now, let me show you a simple feature, matching demo, to kind of demonstrate how we can do this kind of feature detection. To show you this demo, let's go to the Sandbox. Again, remember, it's always going to have the last code that you had played around with, and here, I'm putting in the code to help us do feature detection. Again, setting up numpy and cv2, and all of that stuff, some code to draw things. Look for things in there and read an image. Now, one thing I want to point out, we're using a special variant on SIFT called orb. That's what we're going to be using. You should be able to, in your own setup space, if you have all configured correctly with open CVN python, be able to actually use a Sift, also. I encourage you to look at documentation of Sift on open CV to try that. So again, in this process, we come up with the whole process to find the key points. We draw them, which is shown here, and then we actually run a matching algorithm to be able to match these types of things. And once these matches are appearing, we will print them out, and then , show the matches. So let’s run. Okay, it’s pointing out the image sizes. It found 500 key points in the two images, and found 257 matches. These are the two images. This one and this one. This is the box cover itself directly, and this box, again here, with conclusions and all shown there. , now, if you notice that I actually did a lot of computation to find all of those features on both of these images, lots of colors here, and once it found those features, it matched them. So now, I was starting to put this here, and if you notice, all of the visible parts that were there, here, were matched between the image here and this one image here. So this showcases how this could be done, and this showcased the whole code is right above here. I will even share with you this code so you can actually play around with it yourself.

25 - Summary.srt

So now let me quickly summarize. , I dug a little bit deeper into the whole concept of feature detection, and we discussed the different types of invariants, to scale, to size, and to intensity and rotation. went through in detail to the Harris Corner detector framework and the SIFT detector, specifically talking about issues with scale. And also shared with you some example demos of how we can make this work. Again just for your reference there are additional readings I encourage you to look at them. And also just spend a little bit of time look at information on features. The Open CV side this is one of the harder areas in computer vision and many people are writing specifically and making it available on Open CV to various applications of using things like feature detection. You'll get to play around with this a little bit more and again we'll be using this approach extensively for doing other things in the next set of assignments. Now we will actually move to a few additional examples, like for example, image warping, morphing, high dynamic range imaging and panoramic imaging, in the next module.

# 05-01 Image Transformation.txt

01 - Intro.srt

So far, we have looked at a variety of ways, how we are going to find information in images, to do things like alignment. Let's also step back a little bit and now ask the question how does image transformation happen? How do we actually figure out how to transform or warp an image, translate it, rotate it. Also, look for things like projective and affine warps and transformations. In this lecture, we're going to talk about how we can do this mathematically.

02 - Lesson Objectives.srt

The specific objectives of this lesson are for us to look at how are we going to transform the image? specifically transformations like Rigid Transformations. Specifically Translation and Rotation. Again, the emphasis is that we are trying to do now is look inside the image but not at the intensities, but the number of rows and columns. The number of pixels in the image and transform those. An additional part of it, where we will also look at non-rigid transformations, where we would actually look at our final projective transformations. But more importantly, what we want to do now is start looking at, how many numbers, how many parameters, degrees of freedom we want to be able to model in mathematics of how to do these transformations. So, let's get started.

03 - Image Transformations.srt

To help me situate this let's look at image transformations in general. So far we have looked at image filtering. Now what we're going to talk about is image warping or image transformation. So here assume we have an image, f, and what we're interested in doing is some sort of a transformation. So for example, what we have looked at is we have transformed this image to a new one, and here, hopefully you can see that this image is now much brighter. If you remember where we've looked at before, one of the best ways of doing this would be to look, loop over the entire image, and look at each and every intensity, and perhaps do some calculations to it. In this one, I just added, let's say, 20 more intensity points to it. Now this image is brighter. Here now, let's do another form of transformation to get the output image g. Here, if you notice what we've done, is we've actually now stretched out the image. So if you look at this number of columns here, there are more number of columns here. , so far the same number of rows. So in essence, we've changed the width of the image, keeping the height the same. So in the case of doing filtering, what we're doing is we're changing the range of the image, the inside values of the function itself. So to get the output image, what we have done is done put in a function and transformed the function itself to get the new image. In case of changing the size of the image and such, what we're changing is the number of things like rows and columns, the insight. Not the content, but the indexes of the matrix that we're changing. To do this, , we are now growing a transformation of the insides. That is the width and the height and all of the information associated with it. And that's changing the function. So the transformation in this one image is the change of domain of the image as opposed to change of range of the image. Those of you curious about these terms, I encourage you to look back to your old calculus or algebra material on defining what is a range and domain of a function. And, , here our function is always at image.

04 - Parametric Global Warping.srt

So now my goal is to start thinking about how we going to warp an image to a different size. This is a global transformation, or a global warping, and to help us, we will need to start looking at if we can come up with a set of parameters that on easy adjustment of these parameters. We can take a set of equations and generate a new output from a given input. So now we want to actually start looking at how to get an output image G if the F is given and we have a certain set of parameters to do this transformation. To help us, lets create just a simple coordinate axis here. Usually if you remember I always put the coordinates axis going down in y and x this way. Just for simplicity and consistency. I'm going to start using the traditional set of axis here. this would be the x-axis, the y-axis, and this would be the (o,o). All right? Let's find another coordinate frame. And here, so assuming this is again the (o,o) and this is x and y, you notice this image has been moved a little bit. Let's claim that this has been translated. By tx, and ty. The translation's in x and y. So this in essence kind of says, that now I've translated the image. So this is , warping, or transformation of translation. Another similar example would be when I actually take the image, again, the x and y axis, that we've been paying attention to. Origin except that now we notice. This image has been rotated. And if you draw this axis here, you might say that this has been rotated by amount theta. And that's rotation actually now means that the same pixel values that I have here, now appear in a different setup, and , thinking hard about it, you might also thinking, this is , change the way. Rows and columns look, and how my pixel values are looking. So this is looking at simple rotations. Let's look at a few additional examples of this kind of, parametric warping or transformation of images. Again, my input image. In this one I've stretched the image. Right. In essence, what I've done is I've scaled it. I've added basic additional dimension in the. X axis then I have here, y in this one is the same. Result in my vi is larger and in this case the height is the same. This where we just change one scale parameter is the change of the aspect of the image, and we could do this in both directions x and y, and that's the change in the scale of the image. Very simple stuff so far. Another example where a have the same image. So this time around if you notice that we've actually done an interesting thing. What we've done here is, we've added a little bit of perspective projector warp. Now the top row is much top numbers or top part is much smaller. Bottom is larger, and in essence, the image seems to have flipped on that side a little bit. This is, referred to as Perspective, and this, , is an example where we have created an affine deformation of the image, where now, the image seems to have, kind of have a lot more sheer. Again different types of warps are transformation applied to the same pixel values. And as you noticed the domain, the number, how the x and y's change depend on how, what kind of function we could apply. So let's now talk about what kind of functions we could apply.

05 - Parametric Global Warping Functions.srt

So now what we're interested in asking the question, given an image like this, how do we transform it to an image like this? Again, here I'm showing a little bit of scaling in one direction only. Let's start off by finding two points on my original image, f. This is my original image f. And I'm trying to get my output image g. Here, I have two points that I've kind of marked. , what we're now interested is seeing where these two points would be on this image. Where would they transform to? So, call this one P1. And we call this image, or point here, P2. This P1 and P2 have now moved to different points here. So this is now all these two points. And just for sake of parity we will call this P2 prime, and then we will call this P1 prime. In essence, now what we have to figure out is how to transform the point P to P prime. So in essence, what we need now is the simple function that takes any point, P, in this case, in this case, they would have values of x and y to generate a new point, P prime. In this case, we want to find one function, T, which has a set of parameters that actually applies this thing entirely to the whole image. So in this case, I'm talking about one function that directly applies to each and every pixel. This means that this would be a global warp. Every, the same function, the same parameter function would be applied to the entire image. So in essence, what we're talking about is ,a global function, that given a P, we would always get a P prime. And just to reiterate, what I'm talking about is, I want to come up with a few set of simple parameters. So by now, you've noticed I like to convert everything to matrices because that's a great representation for us to be playing around with. And, , this is true here. What we really want to do is find a matrix that encodes all of the transformation or the parameters, and then when applied it to point p, remember, this would be just simple x and y, to generate a new x and y. , this is the simple two-dimensional transformation we're looking at. Given a matrix M, that has certain set of parameters, when applied to point x and y, any variable in this f function, which is my first one, I would get an output, g, which would have all x primes, and y primes. So, by just looking at this, you may note that this M should be two, a two by two matrix, and now we, let's start to figure out what would be in this two by two matrix.

06 - Image Scaling 2D.srt

First, let's look at the simple example of scaling an image in two dimensions, which means I have an image here, and I want to generate another image. And in this case, this image is twice as big. Again, we have simple set of things here in terms of how I want to get x prime and y prime from x and y. Now let's think about what would be in the matrix M. , the way I want to do scaling would be a simple, right? I would just multiply each component by a fixed scaler, and uniform scaling would be when the same scaler's applied to both x and y. The difference, , is if they're not the same, you would get a different aspect ratio. So let's say if I multiply the, all of the x's by 1 I would get the same width. But if I multiply all the y's by 2, I would get a different height. So though similarly, you know, that different aspect ratio would come in. So uniform scaling would only be when I apply the same constant both ways. If I apply different ones in x and y directions, that would actually allow me to now have different aspect ratios. So what does that mean in terms of our matrix here? , that mean is now we would replace the M matrix by nothing else but two scalers on the diagonal, a, which would actually impact the scaling in the x direction, and b, which would impact the scaling in the y direction.

07 - 2D Image Transformations.srt

Let's look at other 2D image transformations. Again, let's look at this simple equation that we're looking at. Let's say I create the simple equation here, which takes a scale in x and y directions, Sx and Sy. And what that means is that this image now would be scaled. But the parameters Sx and Sy and they would be scaled around the origin point of the image, 0 and 0. I mean, there would be a linear scaling in that direction. Here is an interesting transformation, M matrix. What would happen when I applied this? If you think about it, what it's going to do, it's going to flip the image in one direction. So , what it's going to do is, when I have the y-axis going this way, it's going to flip all of the values on the, this side. That is the right side of the image, to the left and all the left ones to the right. So , that'll be a mirror operation. If I put minus 1 on both of them, that would be a mirror over the origin, so it will flip both an x, axis and the y-axis to generate a mirror flipped image. Here is another interesting one which now I put the terms on the two off diagonal terms, and we would refer to this as shear, so shear in x and shear in y would give us and image that would be sheared. That would kind of have the impact of kind of having top row of the image, kind of move towards the left. And, I'm sorry, top part of the image moving towards to the right and bottom moving to the left. We'll show examples of that in a bit, too.

08 - 2D Rotation.srt

What about rotation? So here, I'm showing a simple example. I have two points, x and, this would be the original points here, and x prime, y prime, where I want the transformation to happen. , in this case, these two vectors kind of show how the transformation would be, and , there would be an angle associated with this. Let's call that theta. So in case now, just looking at the simple 2D example, let me ask the question. So in this case of the simple 2D rotation by a theta, where I have transformed points x and y to x prime, y prime. Question comes up is, how would I now figure out the values of x prime, y prime, given x and y and this angle theta? Now, if you may remember this from your old trigonometry classes, this can be done, and I encourage you to look it up again if you haven't, looked at such things recently. And this would say that x prime would be computed by taking the cosine of the angle theta here, with x the original x, minus y sine theta. And similarly y prime would be given by x sine theta plus y sine theta. So this now gives us an input, interesting way. Now I can write this in a matrix form, which would mean I would move x and y into a column, which would now give me the elements of the M matrix. So M in this case would be cosine theta minus sine theta, sine theta, cosine theta. And if , knowing this angle I can apply this as an image transformation. Again, I encourage you to look this up or try to kind of do the derivation. Just as a hint, to do this derivation, you'd also need to represent this angle and then come up with equations to kind of compare all of these.

09 - 2D Linear Transformations.srt

So, now we've looked at 2D linear transformations. Let's look at that in a little bit of an interesting way. Here's my equation. We know how to look at that. we talked that now a simple way to do would be is, in this case, I can come up with four parameters. And sometimes, for example, a could be the scale and y, and d could be the scale. The for example, the numbers negative kind of give it a flip not. And similarly, things like c and b kind of give it the share information. mixtures of this with sines and cosines tell us how to do rotations. So one interesting to note is all that this means is we are only look at linear combinations of these parameters to generate a newer set of x and ys, given an x and y. So x and y, x prime, y prime are computed knowing what x and y are, and again if I know the parameters a, b, c, d. Let's look at some of the features or the kind of the details of what these types of things for y. They can be used to generate scaling of images, rotation of images, shear and also mirroring. We've looked at all of them so far. Now let's look at some properties of these linear transformations. One thing to note is all of the time the origin still remains where it is at 0, 0. We haven't actually done so far any translation, for example, right? I gave you this example of looking at these matrices where we just played around with how to change them to do scaling, rotation, shear, scale. We haven't actually done translation, so origin remains where it is, always. Second thing to note in all of this would be that lines would actually generate lines again. So if the image had a line, same straight line would actually be generated in the new image, even with scaling. , it might look different in terms of it might be stretched because we may have stretched the image more in y and less in x or something like that, or it might get flipped when I do the mirror. But, it still appear as a straight line. All parallel lines, because of the fact that everything is straight line, will also appear. As a parallel in the new image. And all ratios would actually be preserved, as much as possible, in that image.

10 - 2D Translation.srt

So now let's actually look at translation, something I had ignored before. This is the way, the best way, I can actually translate an image or pixel points on an image. Right? Take any x and y value, and actually have to give a translation, tx and ty. This simple addition would actually give me newer x and ys, x prime, y prime. Notice that this matrix representation does not allow me to reconstruct this. Right? Because in this matrix representation, shown by this, when I do a multi, a matrix multiply becomes ax plus by for x prime. X prime would become ax plus by. And similarly, y prime would become cx plus dy. Notice almost impossible in this linear combination formulation that we've looked at that we can generate an equation that matches this. In essence, these terms are additive. As opposed, these are linear combinations. So then, question to all of you, how do we resolve this? How do we come up with a way, in this form of a matrix representation, that will allow us to encode things like translation?

11 - Homogeneous Coordinates.srt

So the answer is that we need to consider a newer coordinate frame and we would refer to this as the homogeneous coordinate system. what we want to do is actually take the two dimensions that we were looking at before, the x, y, and the x prime and the y prime, and a two dimensional matrix M. Let's start seeing if we can actually represent this as a three vector. So so far we've looked at x and y. So we want to be able to take this two dimensional x y and generate a new three vector. We can refer to this as x y and the third vector being 1, and for just sake of completeness and what we will do with it next let's call this a w. So what we doing is we adding a third coordinate to every two dimensional point. So and what we did is now we're coming up with x, y, w. And the thing we want to remember is what w implies is again, my two dimensional vector, except that now we are dividing both x and y with that third point here, w. Now there are certain subtle things we need to pay attention to. Here for example is my simple two dimensional x and y. I can look at this point here 2 and 1, just looking at x 1, y 1, the values would be 2 1. Now just keeping this convention in mind, if I had omega, or w to be 1 this still makes unreasonable sense 2, 1, 1 applies, but then , imagine I could work w to 2, then 4, 2, 2 applies and if I make w be 3, 6, 3, 3 applies. So, this point now can be represented by this three vector in all three of these values. One thing to note, w cannot be 0 because if you make this 0 x and y also would actually go to infinities, so when w is 0 we can refer to that as an infinite point. This point, 0, 0, 0 is not allowed because we cannot have 0 over 0. That would be an indeterminate point. But all of a sudden, now we have a lot of strength in our hands, when we actually now create this new coordinate system, the homogeneous coordinate system, with x, y, and w. Let's see what that buys us.

12 - Basic 2D Transformation.srt

Now remember, we started off by saying we want 2D transformations where we have a matrix M to give us values of any point p to give a point p prime. Let's look at what we would do to get something simple as translation. So what we want to do for translation is , we would have the axis for x and y. And we want to translate it into a different location again from this and x and y, so this has been translated. Let's say for a lack of better words by an amount, tx and ty. Okay? How we represent this in this new homogenous coordinate system? Well, again, this is what we interested in x, y prime, x prime, y prime, 1. X, y and , now let's think about what M would be. By just looking at this, you notice that we can actually construct it. Diagonal terms are all 1's, and these, these two axis could just be tx and ty. And if you actually do the math, you'll notice that what we will come up with would be is x prime is going to be equal to x plus tx and y prime would be equal to y plus ty. That gives us what we wanted. So translation is easily modelled this way. What about scale? Again, here, the goal is to go from this image, to a larger image. In this case, again, what we did before, the same two by two we had looked at before, with sx and sy, can go into these two values here, and, , the rest of it will work out, because what will happen is x prime times sx, x would be the result. And that's exactly we want. So this gives me a nice three by three matrix for being able to doing things like scaling. What about rotation? In rotation, I want to take an image and being able to kind of rotate it around where how I need to know things like theta. And here the theta would be this value here. All right. And we know how to do this. Again, we know when if, somebody gave me a theta, I can compute using this cosine theta minus sine theta, just in this two by two. these are 0, 0, 1. What gives me x prime, y prime. And we know how to do this too. The good thing with homogenous coordinate system is, these two parameters help us do things like model translations. We can also do shear the same way. Start with an image. We want to shear it. Remember, this was when shear happened like this. And again, we would put the shears in the off diagonal terms. All the diagonals would be run. You can play around with this yourself in these matrix multiplies and you'll see it works.

13 - Basic 2D Transformation p2.srt

One of the things that's important now to notice, this these types of transformations can be combined. To achieve a transformation we are taking single image and then use that to generate a translated image. This has been translated. Then I rotate this image by amount theta. And then I actually add to it things like shear. So in essence, this shows that to get these set of parameters here which now show translation, this is where this comes in. This kind of rotates the image theta that we looked at. And this , also shears it by a certain amount and shears it in this amount again noted by these ratios here. So x y w here can be combined into this and we can simplify this by looking at the now there are nine parameters here in this matrix and if we knew these and combinations of these. , noting here that these combinations can be done priory and saved to and applied to imagery one. Now one thing I may want to talk about right here is most of the time this i value would be 1. Because remember, the current condition we had with the ws. W always wants to come out to be a w on both sides. And that would be the case here. So most of the time, we're looking for these 8 parameters. Okay? So let's look at what that means.

14 - Affine Transformations.srt

So let's start off by looking at affine transformation as an example of something we want to understand and figure out all the parameters for. This would be an equation, we would have these eight parameters that would change to create an affine warp which means is now have an image like this. Which after transformation or warping would appear to be this way. So an affine transformation we are combining linear transformations, the ones we looked at, that were the you know the rotation scaling and all of that stuff, with translations. So in essence, this this image is now moved over here and it's also been warped. [SOUND] The properties of an affine transformation are that origin does not necessarily map to origin. For example, this could be the original origin of this image, we've translated it and then morphed it. Lines map to lines if you notice, all lines are still lines here. They've just changed a little bit but they're still lines. And similarly, parallel lines all remain parallel. These are the basic properties of affine transformations. And one thing, if you noticed, this could be achieved by actually modeling these six parameters. So, parameters a, b, c, d, e, f, I have six parameters. So in essence this is a six degree of freedom on our presentation and once we do this now, we can actually figure out the transformation going from here to there.

15 - Projective Transformations.srt

Let's look at projective transformation. In a projective transformation, what we are interested in is taking an image and warping it in this manner. So , a projective transformation is a combination of an affine transformation we've looked at, but added to that, a projective warp. Properties of that form of transformation is. The origin does not again necessarily map to the origin. You can see that this could've transformed or move translated over to this point. But the line are still straight lines, right? Lines map to straight lines here. But now parallel lines do not necessarily remain parallel. An example of this you may actually see. That for example, if t and h, if they were parallel here, they're no longer parallel if I was to draw an h here and a t here. Let's just do that. And they would actually intersect somewhere. While t and h, unlikely to intersect. And ratios are not preserved in this one either. Here, , we have. What do you think? Nine parameters? Nah. Remember, I always said this will convert back to 1. So in essence, we have 8 parameters. Which means we have 8 degree of freedom.

16 - Recovering Transformations.srt

So now, move down all of this. Let's ask the question, what can we do about trying to now be able to recover transformations from images? So what that means for recovering is given f of x, y, can I generate g of x, y? But also, can I also kind of learn the transformation itself? So what it means is given an image f and given the transformed image, if I know the axis for both of them, this would be x and y, and this would be x prime, y prime. If we know what f and g are, can we recover the transformation, t itself? To achieve this, one question would be is, how many points on both these images do I need to know? For example, would I need to know a point here that would correspond to this point here. I would need a point here that would need to correspond to this point here. I would need to know this point corresponding to this, and this point corresponding to this. All right? So, if I know these correspondences, [SOUND] I would be able to figure out the inverse, by all this transformation function. But how many do I need to know is the question, do I also need to know some inside? Those are important questions, so let's get to that. And we would do this in forms of quizzes.

17 - Translation.srt

So look at this framework here. Again, what we're interested in is simple translation. Have an image. This image, in this case, has been translated by certain tx and ty. Question for you to think about is, how many points do I need to have to be able to now model this transformation? So please put in, how many points correspondences we need? So in the previous case I showed you before, do we need all four of them here? That's one question. And also, how does, how many degrees of freedom can be used to model this? Please fill out these boxes. And then, in this matrix, tell me which parts of this three by three matrix do we think that the values would have to be, related to the parameters that we need to change. Just for simplicity sake, know that we already know this will be one. So, fill out the others, please.

18 - Translation.srt

The solution to this one is pretty trivial. All I really need is, if I know where this point is, and I know this point here, that's the number of correspondence I need. 1, because if I get these points, I can get Tx and Ty. And that's all I need. Number of freedom, , is, degrees of freedom is tx, ty. That's 2. And, , the rest of this matrix should simply be 1, 1, 0, 0, 0, 0. Correct?

19 - Rotation.srt

Let's look at another example. In this case, let's look at rotation. So in essence, the image moves and is also rotated. And let's call the rotation be, the rotation amount to be theta. Okay. So just simplicity's sake, let's say that we can claim c theta is equal to cosine theta. And s theta is sine theta, okay? How many degrees of freedom do we need which would be the answer here. How many points or correspondences do we need between both these images and what would this matrix approximately look like. I'm just giving you this coach so you can now use c theta, with sines here, here, here, here, here, wherever they need to show up. And, , also remember, this would be 1. And the question comes up, how many degrees of freedom? And, also more appropriately, how many points of correspondence do we need?

20 - Rotation.srt

All right, the answer for this one. Let's look at the correspondences. I need this point. I need that again also. That will get me the translation. But this time around, I have to also model this. So I'll actually also need another point. So , we would need 2 points for correspondences. And how many degrees of freedom? Well I need to know tx, ty. And if I know theta, I should be able to compute all of these, right? So the answers would , here would be, I would need to know tx, ty, and then the cosines and sines would show up with the theta. So I need to know theta, tx and ty and I can actually fill this out, to be able to get the answers I want. And these would be 0, 0.

21 - Affine.srt

The same question here this time for an affine and I mean here, warp or transformation. Same drill as before. Please enter the number of points of correspondence I need, the number of degrees of freedom, and what would this matrix look like.

22 - Affine.srt

So , we would again need this point here because this we need for translation. Like in the case last time, we would need this one. [SOUND] And that would give us theta, and here I would need one more point. That would kind of give me more information about how this transformation happened or the warp happened. So I would need three points of correspondences, 3. Number of degrees of freedom, we have looked at this example before and if you remember four, the answer is I would need to know a, b, c, d, e, f, I already know this, these would be 0. So , I need 6 degrees of freedom.

23 - Projective.srt

Last one is, , projective. Same drill for this one. Now, you want to look for, , projector transformation. How many correspondences do we need, how many degrees of freedom? And fill out this this three by three matrix.

24 - Projective.srt

So for this case, we actually need all 4. This point, and that's the map to this one, this to this one, this to this one, and this to this one. Number of degrees of freedom. Well, if you remember right, remember this is still 1. , we need a, b, c, d, e f, g, h. So, 8 degrees of freedom is what we need here. [SOUND]

25 - 2D Image Transformations.srt

Now let me actually just recap for a little bit. So this is the 2D image transformations we are looking at. We have an object that could be translated, this could be the image itself. We could scale it, in this case I'm showing the scaling. We could rotate it, an affine warp, and a projective or perspective warp would be this one. So let's summarize all of them. I'm going to show this with a simple kind of a table here and we look at how we actually want to do the transformation, what the three by three looks like, and what kinds of things does it preserve. , simple translation two degrees of freedom, and we know kind of how to model this and these are the two parameters we would actually be kind of modeling. And in this case you only get translation, orientation is preserved. Case if you clicked in where there's a rigid transformation, three degrees of freedom, the object is rotated. Here we would change if there's translation involved in these two values, but also just the cosine, theta, and stuff like that would change these four values here too. This would still remain one as it is here. And, , zero, zero, zero, zero. I'm not implying that this would be a zero, it just means that the cosines always based on theta would be coming in. In this case, the lens would be preserved. Third case similarity, where now we have scaled things out four degrees of freedom. what that means is now we have the two parameters for translation, assuming there's translation going on. And scale parameters would be here and the rest would be the same. For affine, we've looked at this just in the quiz before. Everything that would preserve, a parallelism would be preserved. Lines would be straight and everything else. We also know that the six parameters here would be the ones we would need to model and that's the six degrees of freedom. Projective eight degrees of freedom, all of these. This would be still one. Straight lines are preserved, parallelism is not preserved. So if you notice as we down this preserves orientation, because it's only translation. This doesn't preserve orientation, but it preserves lens, but next time all of the angles are preserved. Parallel lines and lines are preserved and only straight lines are preserved. And if you notice this is how we can go through and look at different types of images from starting here looking at translation, rotation, scale, affine, and projective.

26 - Translation Demo.srt

Let me now show you an example of a simple translation using our browser code here. again, we start off by just simply doing things like importing a computer vision two kit and numpy and there from there on, we're going to look at doing things like read the image. How just reading the tech image itself, get more information from it. Again we can show it and here just see me creating a simple translation or transformation matrix. If you notice it's transform by 100 pixels and 50 pixels and the diagonal terms are one and one in my matrix. , using this now, this translation matrix we can print the translation matrix, and then, , apply the transformation using this piece of this co, function here which actually takes the transformation matrix and applies this to this image. Let's see what it looks like when we run this. So here, , you see the translation matrix being printed out. Here is my image. This is the original image. And , this is the final image that has been translated by, , 100 pixels and 50 pixels.

27 - Rotation Demo.srt

So in this code example, I just want to show you how we can do transformation f, a specific form that is rotation. First two lines, , are just loading in computer vision and numpy, then just load in the image figure out the height and width of the image, show the original image. Here we want to actually showcase a rotation around the origin at zero and zero, that is the point of the image right at the top corner. And , what we do is we apply a rotation of 45 degrees. So using that, , we have now computed the rotation matrix. We can print the rotation matrix here. And that's shown here. And , then we can apply the transformation by this function cv 2 warpAffine. Again, rotation matrix. The image itself. And that way we now can show the image here with this line of code. In this part of the code here, now we apply the same transformation except now that we are applying at the center of the image. to achieve this we have to transform the point to the half the width and half the height of the image, and again we are rotating this in the other direction, minus 45 degrees, and here one is still the scale. We don't want to change the scale. [SOUND] So, again, now we print out the rotation matrix and apply it with new images transformation using the same function above and display the image. Let's see what this looks like when we run it. This is the rotation matrix when we have the image at the origin. That is the point top here, of the image. And the next one here is after we've actually figured out the center of the image. That is we've moved to the width half and half height of the image. And then applied the same transformation of rotating by 45 degrees. Notice again this is minus 45 degrees. This is plus 45 degrees and therefore the signs are different. This is our original image. This is the image rotated around the origin point. And again as earlier stated I rotated this image by 45 degrees, so , now it's truncated or cut at this top here, but you can see that the image has been rotated by 45 degrees. This is the final one where again this time around I've done the 45 degree rotation at the center of the image. That's why we actually have put in this for different terms in the transformation matrix. So actually our rotation would be at this point. And here you notice, , the tech sign now has been rotated this way 45 degrees and shows a transformation of rotation by 45 degrees at the center of the image.

28 - Shear Demo.srt

Now let me show you a bit of code for the scale and shear transformations applied to an image. The usual preamble of loading computer vision and numpy, and wrote reading the image. And here what we do now is we want to actually be able to apply a resize scale, by what we're applying is, transformations in x and y, 1.5. These are various types of, additional information we can to our resize function to be able to scale it. And that allows it to kind of change the image and scale the image here. And I can, , just show the image, by just scaling it. This one if you notice, I didn't spend time building a transformation matrix because this piece of code already takes care of this kind of stuff. The next example is where we can apply a shear or a skew in the horizontal axis only. So now, for this, we will create a matrix, the diagonals are still ones. We don't want to do any kind of scaling here. But now I'm applying in the off diagonal terms, a small scale in this case, just in the x direction. And I've given it a 0.5. And using that now you've actually computed a or come up with a new transformation matrix. You can print it and then apply it using again our affine, warpAffine function, take the image, and now here, we just do some different types of transformations, apply it, and , can see the image. Let's see what this looks like. Here is just the printout of the shear matrix. This is the original image. This is the original image scaled by 1, 1.5, that is, we've just added a little bit more size to it. And this is the output of a shear transformation where we have applied a 0.5 shear just in the horizontal direction.

29 - Affine Warp Demo.srt

Let me now demonstrate a bit of code to do an affine transformation of an image. The usual preamble of loading an image, an out, open cv and numpy, reading the image. And here, rather than do other types of things with transformations, we're going to take much of a, approach where we can identify the points of transformation. So I now come up with first user points. And if you notice here, I'm giving it three different points, in first image, and three different points in the second image. Using these two points, I can now create a, affine, apply the affine transformation to compute a transformation matrix. And I'll, once we have the transformation matrix, we can apply it to the image that we already know all the other information of like, for example, width and height. And after, , we have applied it, we can display the image. Just run this code here. So, from those three points that we used, we were able to compute an, transformation matrix which is actually printed out here. This was my original image. And this is the final output image after the transformation matrix applied of giving it an affine warp. Now you can see the image have been warped, but again, notice straight lines are straddle straight lines as we talked about earlier in the lecture. It just has more of an affect of being able to be, create a warp, or a kind of a shear in two different directions here. All lines are still straight as you can see here. The straight line still remains straight in this transformation.

30 - Perspective Warp Demo.srt

So for my final example, let me now showcase the perspective or projective transformation of an image. Again, the usual things. Here just to be different we're going to play around with a different image, the Berlin Wall image. We can actually compute the, the height and width, and all of that kind of stuff here from the image itself. And again it should be no surprise to you so far that now we need four points. So, for example, so in the first image I'm going to find four different points, so I've given them those coordinates here. And for the second image I've found four other points, and we need those for perspective transformation using these two points, points one and point two. In this code I'm going to compute the matrix, transformation matrix that actually uses these four points to compute the perspective transformation. We're going to print it out and then , as we have done before, we're going to just apply this transformation here. Let me just run this code. Here you see the perspective transform. Again, this should be no surprise. This value is still 1. But we have other values in the rest of the matrix. So this is the original Berlin Wall image, and you'll see why actually this image was chosen to showcase this effect of perspective warp. This is the perspective corrected image, now just being able to apply the perspective warp, and again the points were correctly chosen. You notice now all of a sudden you get a warp of actually seeing this image right in front as opposed to, in the previous case, where you saw an effect of foreshortening. Again, notice here straight lines remain straight, which is what we talked about as one of the values of these types of transformations.

31 - Warping.srt

Now let me actually talk a little bit about warping and we are going to get into lot more detail about this in the next lesson. So here , I'm just showing you two images. Right, so I have a point here and I want to generate a larger point which is being rotated in the new word image space where the domain is x prime, y prime so this may coordinate xs, and , I have my transformation. So I take this pixel and I warp it to this location here. So in essence, what we're doing is sending each pixel from f(x,y) to its corresponding location with a transformation T(x,y) in the second image. What happens if the new pixel lands between two pixels? Remember, this could be much bigger in this open space that we coming up with. In that sense, we're taking a bigger image and filling information from there. In the forward mapping, what's really done is that we would distribute the color among the neighborhood pixels. So, if this is the pixel I have there, and it shows up there, I would kind of distribute the color in the ones around it to generate a new pixel. And that's what I would actually do. This is the forward warping process. Another well-known process is when we actually go inwards, backward warping. Again, I take a pixel from here, and I want to go and figure out the inverse transformation to find where would it actually end up, and what would I do with it. So again in this, what I will do is take each pixel from the warped image and find its corresponding location, and move it to a new image as long as I know the inverse. Again, in this case, what happens if it shows up in between pixels? In this case, what we would do is we will interpolate the color values. Those where we were redistributing. Here we would interpolate the color values and fill it in here from the neighbors. Again, how do we do this interpolation? Remember how we did things like filtering images and stuff like that. We could use those types of methods to help us do this. Just to do a quick comparison, forward versus inverse warping. Which one do you think is better? Well usually, inverse mapping is a better map that, because it eliminates holes. We're always going from something we know how to get to, to much more of an original image. And that allows us to fill in all the color information. If you sometimes go from one to the other, we might run into places where we don't have, we'll have to do some sort of hole fill. I'll talk a little a bit about that when we talk about morphing. But, the important part is to do inverse warping, we need an invertible warp function. Now I want you all to think about how we would actually create an invertible warp function based on what we've talked about before. And see that in some instances, especially for rigid warps and stuff like that, that's the easily computable inverse functions. Especially when you have rotations and translations and you're doing scales. It does get harder when you do a bunch of other things and not all the time especially when you do have, not a global warp. It gets harder and harder to compute those.

32 - Summary.srt

So to quickly summarize, we learned about image transformations, not just about image filtering. Remember, we talked a lot about how to do warping in things like even when we played around with panoramas. This is some of the foundations that we're going to use. We're going to talk about morphing, but we're going to come back and use these, not just image filtering, but image transformations and warping to help us do the kinds of computational photography that we are interested in. We looked at all kinds of transformations, rigid, projective transformations of images. And we actually kind of looked at what parameters and how to do this simple types of things using matrices. If anybody's curious, more, more detail exists on chapter 2 of the Rick Szeliski book. I look it up. And also, I just, , as usual, relied on other people's slides to generate the slides that you saw. More information will be available on the website.

# 05-02 Image Morphing.txt

01 - Intro.srt

So in the last lecture we spent time trying to learn about image transformations. We looked at how would we rotate, translate, or even do projective transformations on images, and learn about image warping. Now lets spend more time trying to understand what happens when we do these warps that are non-ridgid. Specifically what I'd like to introduce is the concept of image morphing. That is how would we take one image that we know some features in and transform it and morph it to look like another one. This is a widely used technique used in films a lot, and actually is one of the stronger techniques in competition photography, something we'll be leveraging a lot in future lessons.

02 - Lesson Objectives.srt

The objectives of this lesson are for you to learn about image warping. We will specifically talk about forward and inverse warping, the kinds of image warping techniques that are widely in use. Then we'll talk about how we can actually do the warping using a mesh. With this, I'm going to introduce the concept of image morphing. And specifically, the feature-based image morphing technology that's widely in use, and you've seen it many of times in different types of movies.

03 - Recall Image Transformations.srt

In past lectures, we have talked about the whole concept of image transformation. That is taking an image and being able to transform it. One standard way of doing that was by doing filtering and what we talked about was image warping, or transformation in the last lecture. Image filtering was aimed at taking the information of the intensity values and changing it. While image warping or image transformation was changing the number of rows or columns off the image, that means the range of the function f, in case of filtering, we were changing the domain of the function. So, just a recap, image filtering was when we took an image, and changed the range of the image, that is, the intensity values to be able to kind of change it. In this case, I've made it much more whiter. And image warping or transformation, , we applied a functions actually would change the, the size of the image itself, the range of the domain of the image would, in this instance, means the number of rows, the number of columns. Domain , is inside the size of the image itself, while the range was the values of the intensity. So, , now we're actually applying this to insides, while in this case, we're applying it to the function of the image itself.

04 - Image Transformations vs Warping.srt

Now let me talk a little bit about image transformations versus image warping. Specifically what we were interested when we talked about image transformation or the related warping that came with that kind of image transformation was, we were able to convert an image like this to another image and if you notice in this one all of the lines remain straight. So you look at the capitol t it's still the same here. In essence, all of the shapes remain consistent. , there's stretching going on. The H is a little bigger. But in essence, if you notice is all the lines remain lines. Warping comes down to is when we want to take points in an image like this and map it to another set of points and not stick to the constraint that lines remain lines. For example, in this case now, if you notice, the T is curved, E is curved a little bit, C and H. In essence you can see a swirl coming in, to the T. And in fact you notice this is no longer a line either. And you see a lot of different types of warps going on. So here essentially what we mean is, we need to now find a mathematical function for warping from a plane to another plane. And in this case , there are two planes, but we're doing a lot of non-rigid warping in-between. So, in essence, that's what we're going to talk about now is how we go from image transformations like the one we looked at. Most of them were rigid, but they were projective and -line warps and stuff like that. In this case, we're really going to start talking about non-rigid warps.

05 - Image Warping.srt

Now in the last lecture, we have started talking about image warping already. So now let's actually try to make it much more foundational. Again, let's take this simple image here. And what we want to do is we want to distort this image. And one of the ways we can do this is distort it by simulating some sort of an optical change in aberration of some sort. A typical one of that type would be something like a fish eye lens here, right? If you notice, if you had a fish eye lens in a camera, everything in the front would pop out. And , if you notice, farther things are going back here. And in this case, there's a bulging in the middle here, right? And this bulging is showcased by E is much bigger here than it was there and , there's all kinds of curvature information. Another operation could be, we can project this onto a curved or a mirrored surface. So in this case, I projected this image onto an arch. So if you notice, it's curved and , it now has a shape. In essence, this almost could be looked at as taking this image and warping it, and putting it on some sort of an arch so now it actually looks like, curved. This could also be referred to as texture mapping where I made this image be a texture on a surface. And in this case, it could be a curved surface like this one. It could be a cylinder or a sphere also. Another method is, we can take polygons and kind of discretize this into small regions of polygons. And , each polygon would be distorted. So for example, this is my original image. This is my output image. Let me show you some polygons on top. So here I've , in the original image, shown a small quadrilateral. What I want to do in this one is I'm going to take this quadrilateral, and it will be warped in this one. And any information that's in this quadrilateral, rectangle in this instance, would now be transformed in this one. , you see that warping here, right? The T is there, you notice the T is still there. But the E is curved. And , there are other polygons next to it too, I'm just showing you this. For example, you can see the other polygons. This shape here, and another one is also down here. And there is another one here. And these are again polygons. In this case, I'm showing the polygons not to just be linear quadrilaterals but to be various types of deformations. In essence, what's happened in this instance has been, this has been converted into a warped polygon here. And , there's another whole concept of distorting using morphing, and we'll talk more about that too.

06 - Two Methods.srt

So the two traditional methods widely in used for doing any kind of warping are forward and inverse. We briefly touched on them in the last lecture, I'm going to now just give you more details on them. So, to help us understand this, let's start off by just having two images. We'll call the first image the source and the T, the target. We want to be able to take information pixels from the source, and create a new image or a target image, that would have the deformations or the warps that we are interested in. For simplicity's sake, let's imagine S has a coordinates base u and v, and the target has x and y. Now this is slightly a different notation than we've used so far. In the previous ones, we talked about S being x and y, and the target being x prime and y prime. Just bear with me for a second, we'll come back to that notation in a bit but this is just to kind of help us understand how we can do this simply. So forward warping is taking, generating a new x and y by creating a warp on X, which is applied to u and v here. And warp and Y, which is also taking u and v from this one. And generating a new set of coordinates, x and y. So here, , the warp is X and Y, apply to the values in here, and generating this. Backward or inverse is trying to do the prediction of where things are in the target based on what are things or pixel values in the source. So here, , we generating, in essence, the u and v, by doing a deformation or a transformation, or a warp, U and V, which takes values from here and generates this image. So in essence, while we have this, we're trying to now do the opposite of trying to figure out how to go from pixel values in all of the range values here into this one. To help understand this, let's take a simple example. Now this was, , what we want to accomplish for forward. And this is what we want to accomplish for inverse. Here I'm showing an input and output, for, , the forward, and the input and output for the backwards, or the inverse. And the input and output for the inverse warp. Now, to help us along, what we can actually start imaging is, that this could be, u, v could be completely integer values, will become clear in a second. While, the output has to be in real values. While in the case of the inverse warp, the opposite is true. If I have all of this in integer values, I could actually now have real values input. Let me start off by finding a few pixels in the original input here and see what they would look like in the output. Here is one, and here's the other one. When this pixel is moved over, the information from this pixel is moved over to the output, , because of the warping, it won't be just in the regular grid pattern of this image anymore. In fact, the intensity value should be distributed in the raster scan of this image, and , what we now need to do is generate a new image that would have the values that came in from here. the warp would be based on these two X and Y. And imagine this, is now, , moving here. So, in essence, what we've got is from here to there actually we warped the image, and from here to there, we warped this part of the image, this region, is now moved here. So this is the forward warping process. In the case of the inverse, the opposite is true. We would have this region, we want to move it here, and we would have another one. So again, let's take these two pixels again, these are integer values and all of the values that come from it and we want to try to move into this range. Now, let's talk about the problems. The big problem when we do a forward warp is, in this instance, I moved all of the information to a warped region here. And , imagine if there is another pixel or region here, then I want to do it here. , if it moves here, there might be a cause for an overlap. , it also is possible that this region here, that this region here next to it, moves here. And , all of a sudden, I have no information connecting these two parts here. So , this would be a hole. So overlap would, would mean they go next to each other overlapping in the information or, in this case, when they're far away there would be a hole. So this is one of the problems when I go forward. In the other end, when I actually am going backwards, that is doing the inverse warp, I know that these pixels belong to here. But the problem that comes in is, when I actually start going for something that becomes smaller or the minification process. I'll show that in a second. That would result in all kinds of artifacts. One of the bigger artifacts in these processes is aliasing or blocking.

07 - Minification.srt

Now let me talk about the problems of magnification and minification that applied to, in both forward and inverse mapping. So let's take this region, it's being copied over a warped in this region and be much, bigger in this and so and so. So this is the issue of magnification. Where this, all of the information from here, is now much larger here. , the problem that you can expect is going to be because of aliasing, there's now lot of information here, much more information here, so it will be blocky and choppy. In case of the inverse, we'll have a region and what's happening really is this part is going there so now I have to figure out how to do computation of this from here and this is a problem of minification. To solve this, what I would do is kind of even discretize this region here and copy out the discretized information, from here to there. So this allows me to now generate a much larger thing. And I would take all of the values from here and inject in here. And similarly, this one would be here. And that's how I can actually start doing, and we can do a variety of other things that allow you to , anti-alias this. However, in this case, we would actually have problems because we don't exactly know how to do the inverse warp. Again, we'll talk about that briefly again in the next few slides. But one thing I want to point out is that, in essence, what we want to do is, we want to be able to sometimes figure out the best computational mechanisms to support it. There's a lot of literature of this kind of stuff in texture mapping and stuff like that in computer graphics, that's aimed at addressing this problem with both forwards and inverse mapping. One specific method I want you to look at is a Two-pass transform that does the geometric transforms that we had looked at in the previous slides. Which was the whole concept of doing rigid map line warps and then being able to do smaller warps to be able to get smaller differences, and by combinations of those. And the fact that these current processors can actually deal with these geometric transformations much better. You would actually have much quicker ways of deforming regions and images. You'll be able to generate nicer artifact free visuals and videos

08 - Recall Forward Warping.srt

So now let's look at forward warping again. And this time around, I've actually changed the notation back where x prime, y prime are the coordinates in the target. And x and y are the coordinates and the source. And we want to create a transformation that takes this function f and generates a new function, g. With, , the inside values having information that show the warps. So we have to now take this in pixel value here, transform it to this one. So in essence, what we are trying to do is send each pixel from f(x, y) to its corresponding location in g of x and y and the transformation is T(x, y). So now the question comes up is, what happens if the pixel lands between the two pixels? So now I'm gotten rid of the images, and kind of just pointed out what I mean by the fact, that the pixel line. So this is one pixel here, and , now because of the fact, we have changed the domain of this image. We now have pixel values here, but this one is now falling in between all of them. To achieve this, we have to distribute the color among the neighboring pixel to generate this new pixel, and the technique widely used to do this called splatting.

09 - Recall Inverse Warping.srt

Let's take the Inverse Warping example. We have pixels going this way and now we have the inverse of the transformation. We want to create or get every pixel from g x prime y prime to its corresponding location in this image. Transformation is again the inverse and now the question is, what if a pixel comes from between those two pixels, in this range? Looking at it again, without the images. , see this pixel now is falling in green, and in this instance, what we want to do is interpolate the color value from the neighbors and fill this in. This will not have any holes and stuff, but it will have minification problems. Forward warp, that we talked about, the forward warp will have more problems with holes and overlaps

10 - Forward vs Inverse Warping.srt

So, which of the two methods Forward or Inverse warping is better. Usually, the inverse is much better because it eliminates holes but it does require an invertible warp function which is not always possible. Again we're not going to talk much about this anymore here, but I do encourage you all to start looking up this in more detail. This kind of stuff is covered in much more detail in computer graphics classes.

11 - Mesh Based Warping.srt

Now let me introduce to you an additional concept on how we can actually do this warping that actually tries to avoid some of these granularity and aliasing and minification, magnification problems. And that is using a mesh on an image and deforming that mesh to generate the warps. To demonstrate this let me use these two images. We will refer to this as a source and this as a target. Some of you may recognize this person, actually these are images from my PhD thesis from many years ago. And the big difference between these two images is one neutral expression, the other one is a smile expression. Now we are interested in computing the warp between these. Now, one way to compute a warp would be finding out a corresponding set of points that are common in between these two. To achieve this, many different methods could be used, I'll talk about one in a second, and , once we find these corresponding set of points, we can interpolate between them using a displacement field. So first, we want to do is take this image and this or actually represent this as a mesh, which each element is a set of triangles. So here I'm showing that I can actually now take this, put it on top of this. And this would be rectangles but what we're interested in is triangles, each of every one of these squares would actually be two triangles. And you, you see what I mean by this, and now , I have regional pixel the values intensities, and stuff like that, of an image. Put on each every one of these triangles. But we can then do is use an affine model to transform each triangle from one to the other. , we will also want to generate a similar triangle mesh with the deformations that kind of show all of this. This is my displacement field here, if you notice that there is a little bit of, asymmetry. The eyes get a little smaller, and, this showing most of the motion is here, and most of the motion is here, you see a lot of deformation. Again, deformation is 3D, but in 2D you can start noticing this is where the changes are. So using this now, actually we can come up with a model of how we would transform our region points from here to here to get to this. And , to achieve this, one will be able you affine model for each and every one of the triangles then we'll use inverse mapping. To be able to go from there to here, and being able to then generate a warp field. So now you notice the interpolation resulted in going from a neutral expression, to a small expression. And what we did, was we created a warp from one to the other, and in this case I showed you all of the in between frames to kind of make it look like a person went from neutral to a smile. Again remember, we did not have any of those frames, they were generated because of the warp field. Let's look at it one more time

12 - Image Morphing.srt

Let me use this concept to introduce the basic premise of what is called Image Morphing. Here, you see, now a sequence of images generated, to be able to go from one frame to the other, which we actually captured separately the rest of them were generated, on their own. So, this created and animation that changes all morphs. One image, and it could be shaped also into another through a seamless transition. This is widely used in movies and is actually a very well known concept in even trying to understand shapes. And in fact this very famous book I recommend anybody who's interested in shape and form to look at this, that talks about biological phenomenon that could be tested to this. And talks about we can, , take a simple surface like this. And by changing the regions and the warp itself, generate other types of fish that actually, you know, have the same kind of structure but , now have different shapes and stuff like that.

13 - Image Morphing Approaches.srt

Let me talk a little bit about image morphing. Here I'm going to demonstrate this. just to point out, this may not be exactly the images. But, we're just going to learn about how we're going to do image morphing. In the previous slide, I showed you how we can use triangles and deform them with the displacement map to be able to generate a morph field between two frames. , we don't have to stick to just triangle. It could be a quadrilateral mesh that's also displaced by various types of interpolation techniques. Minimum energy methods are widely used with this to compute the best possibly way of deforming from one surface to the other or one mesh to the other. So for example, in that case, we would take a like this, have another one that would be deformed appropriately for this and that would allow us to generate in between images. Another method we can use is find corresponding features between these two different images. So in this case, , it's an image we can actually relate to quite well and start identifying common features. Both of them are faces, so what I can do is now mark out the, you know, the corners of the eyes, corners of the nose, tip of the nose, and also the three points on the lips. And , I can do the same for the leopard, or a cheetah image. Again, if you notice, these eyes have different forms, but now I have these corresponding points. And I know this point is that one, this point is that one. And remember, all of the work we've done with feature detection, some of these will come out off using things like algorithms. Then they could be matched. We can actually do more. We can actually come up with corresponding oriented line segments. Which, in essence, also defines details like how would the translation rotation and scaling happen on these region. , what that kind of mean is, now the nose is this direction. I want to keep that, that nose may have shrunk in size if I draw a line like this. And if the eye's were smaller, if I connect these two lines, the eye's will also get smaller. So in essence, to achieve this, we will now create oriented line segments that are connecting these regions and giving it a lot more structure. So now if the eyes were small and these two types of things, you would notice the difference here. Here, you see the line segment is smaller for the target. And , the lips have moved down, the nose is approximately the same, but the tip has moved up. Now, , all of this detail is important, because now we can actually do the image transformations like, translation, rotation, and scaling, and allows us to have more control how we want to actually see these things.

14 - Feature Based Morphing.srt

So let me show you now, complete feature based morphing approach here with these two images. First we're going to put features, here these kind of show off the eye features, eyeball, lips, and everything. And we can apply the same to this image also. By using this I'm now doing the morphing. We get. So let me now actually show perhaps one of the most widely used methods for trying to do feature-based morphing, and this was actually in the Michael Jackson's black or white video. This is a method actually done by Byron Neely. Very early in 1992 and actually just for those of you interested even the kinds of techniques we've looked at like cross fading and stuff like that were actually widely used in the pre-computer era, in 1940s, of trying to do fading between things to kind of show the morphing examples, I'm about to show you. And the effect really was kind of do fading between two different types of images and kind of changing the fade level to show you a morph. Now , with the, now with the kind of the geometry and all of that stuff available, with great patterns we can do a much better job. So again, this is the video that you may have seen. Here you notice a couple of interesting things happen, right? And this is where feature based morphing comes in and notice the hair. They can control how the hair comes in. The eyes are all aligned, , this is very nicely choreographed. Everybody was doing the same exact steps to music. An, lot of things about features if you notice this range and shape of that is much more nicely adapted. Notice another example, where you'll see the hair grow again. these are because of the corresponding feature lines help you do this. I think quite an impressive piece of work, you know, and this kind of image morphing is widely used in the special effects industry. Just for fun, let me actually show you another interesting example, I wanted to showcase we can do this very easily, so I went ahead and from our College of Computing web page just took pictures of. Our deans legally and the three of us who are the associate deans. Ron Arkin, he's an associate dean, and Charles S Bell and myself. And I'd just thought I'd generate a nice morph of all of us. So, kind of fun and interesting.

15 - Jaguar Morph Demo.srt

Now let me just show you a simple demonstration. And I'll actually be sharing the code for this demonstration that you can actually use for a variety of things. you'll have an opportunity to play around with this on your own. Here I want to demonstrate how we can morph between the two different types of jaguars, the jaguar the car, and the real jaguar here. Now recall, to accomplish any kind of morphing, we need to find feature points to create a mesh. In this case, I've actually created a mesh that covers the whole image, but I've found specific feature points that I want to be able to identify in this image here. And this is, , now the same mesh with different feature points applied to the real jaguar. So, again, you should see that this feature point here is matching this one. This feature point is matching this one. So once I have these two meshes and I'll now figure out how to do this alignment of these types of meshes across the two images, I can use that to generate a morph sequence between the two. So here you see the animation sequence that is morphing from this jaguar to this jaguar, and you notice all the key points deforming, and these mesh triangles deforming with the right kinds of pixels, and warping them appropriately to generate a new sequence. Let's look at this again without the triangles. So there you see the original image sequence now without any of the triangles and see the morphing. We will share this code with the slides for you to play around with this on your own with your open cv package that you've been playing around with.

16 - Summary.srt

So to conclude this lecture, let me summarize what we've learned. In this lecture, I've given you more details about image warping. We've covered a variety of topics on how we can actually do much more of the non-rigid effects like spherical, and also defamation of images. Specifically talked about the basics of forward and inverse warping applied to image transformations. Then actually extended the whole approach to take a mesh and use that to warp an image. We looked at both triangular and quadrilateral meshes, used that to introduce the concept of image morphing and then discussed variety of methods, including how we can do feature-based image morphing and different applications of image morphing. For full details, I recommend you look at the classic paper on feature-based image metamorphosis. This is the paper that actually was used as a foundation of some of the effects that we've seen in this class. And , the chapter three of Ri, Szeliski's book also has more details on all forms of image transformations.

# 05-03 Panorama.txt

01 - Intro.srt

In the beginning of this class, one of the applications of computational photography that I had introduced was how to build a panorama from a series of images. In this lesson, let's actually figure out how to do it based on the kinds of things that we have learned so far. So, I'm going to introduce the whole pipeline of how we can actually take a series of images, find the overlaps in these images, and use that to generate the panorama that we have talked about in detail. But now, we're going to learn the technical details of how to do it.

02 - Lesson Objectives.srt

The specific objectives of this lesson are, one, we will really learn how to generate a panorama, building on the ingredients that we have actually spent a lot of time on until now. We will actually revisit the concepts of image re-projection. We will learn about homography between the two images. , looking at two images and how we can compute the homography. They'll let us do both the alignment of images and registration and also kind of learn about how we combine those images. We will also learn about things like how do we actually find points and reliably detect them as inliers versus outliers. And then finally, I'll actually discuss some of the, specific aspects of how we construct panoramas and our ideas of different types of things. Again, something we've touched on. So again, the goal here is to bring all of those concepts together and show you how we can actually do something like panorama building.

03 - Review 5 Steps.srt

Now recall, that we have actually talked about five different steps that are important in creating a panorama. First, , is being able to capture images. We've looked at how cameras work and how we should be able to use them to capture images. We'll talk a little bit about it again here specific to panorama building. Then we have actually discussed how to detect and match features in images, a pair of images or a sequence of images, that will allow us to do alignment. Then, , we talk about how we can warp an image to align the images to kind of have no kinds of ghostly artifacts. Then also, we actually looked at how we can do blending, both at the frequency level to be able to kind of merge images, we can fade images, and cut images. The final part, cropping, you know how to do, it's rather easy. It's changing, , the domain of the image itself, that is finding the range at the, at the number of columns and number of pixels that we want to actually use to do cropping. Cropping, , is the last step and it's an optional step to be able to kind of now just create a rectangular image out of a series of images that we use to build a panorama and in essence, it's really about finding the right number of rows and columns and the pixels to create a rectangular image.

04 - Align Images Translate Warp.srt

Let's actually look at the whole concept of aligning images, right. First thing I can do is if I have a pair of images left and right, I can just translate the image on top of each other, find similar features. Just showcased here as long as I can do a decent job of aligning some of these things, I might be fine but will I really be? I mean, again, the options that I may have in this kind of relationship is I may be able to put a left image on top and the right image on the bottom and do a best possible alignment that way. Another option is I can put r, the right image on top and the left on bottom and you can see that now we can actually start figuring out a little bit more between these two images but in reality, we need to be able to do a lot more. In reality, what we need to do is be able to kind of blend these images together but before we do that, we also have to warp the left and right images this warp means now, if you'll notice, is that the image is no longer rectangular. And, , what that means is now we've been able to kind of merge the regions that similar across both images. So warping, in this instance, is a better solution now we know how to do that kind of stuff

05 - Bundle of Rays Contains All Views.srt

Let me introduce to you a concept of a bundle of rays. Okay, , again, remember, what we are interested in is rays of light that we want to capture in a camera. Bundle of rays implies is that, at any point in a scene, it's a concentric set of rays of light hitting that scene or this point. So in essence all rays of light as converging to this specific point. In my world what I have is a camera. At this point I can now do a variety of things. I can actually rotate this camera just at one point. So we can generalize this motion by this curved arrow here. So what I mean by a bundle of rays now is nothing else, but at this point, we are going to get a concentric set of arrays of light. And here I've just put them equidistant. We just kind of show that all light is converging at this point here. And what I did by rotating the camera at this point was try to capture all of them. , you know, these rays of light could be coming in all directions. Let's now look at a specific type of views to help us understand this. For example, this is my first view. I have a subject here and I use the camera here. And now I get these bundle of rays coming into my camera. , I could also have another view. Right? The camera was pointing this way. This is my view. This is my normal from both sides here. And I now have two views. So now question comes up is, if I have these two views can I create a synthetic wheel that is in between both of them, which exactly does have the same points that I have here? So this point here is also visible in this image here. And these points are also visible here. So by just combining these two can I now generate a new view, a synthetic view? So the belief is that it's possible to generate any synthetic view of a camera as long as it has some center of projection. This one. So as long as I keep my camera at this center of projection and if I do get this view and that view, I can synthesize this one. And this is something that's important to note and that's one of the reasons when we talk about panoramas we want to actually rotate them around one single point. Either actually have them have a path like this or just at this point rotate our camera like this.

06 - Image reprojection.srt

So let's look at the concept of image re-projection. Our interest is being able to take a scene like this, and capture it on two different images and more being able to then, , project the information between those two images to be able to reconstruct a panorama or a mosaic. So for the capturing this scene, imagine I have the first viewpoint, we'll call it projection plane one. And this showcases the second projection plane. Again, this is the scene we're capturing. So what we're interested in is relating these two images, which have been taken from the same camera, and map a pixel that is in this scene from both PP1 and PP2. So it means is we now need to cast a ray through each pixel in PP1, and draw the pixel where a ray intersects PP2. So that means is, this point here, this is a ray of light. This is a point here in PP2. And, , this is a point in PP1. Now in essence, this is the kind of stuff we learn, we, we could do with things like, feature detection, right? The same feature is visible in between them, and now we need to detect the same feature. And , this implies that this ray of light is not going through this. So, rather than to make this into a 3D re-projection problem, we will think of it as a 2D image warp from one image to the other. What that means is, we're going to take these two different projection planes, and think about what would be a work between these two images that would allow us to align these two points, even though these cameras would have moved as I moved this you know, took the mosaic or a panorama. And remember, we're taking multiple pictures of the same scene and we're moving the camera a little bit. So while this ray of light is going through the same thing, these cameras, or these viewpoints, are shifted a little bit, or moved a little bit. we want to be able to do this without knowing whole 3D geometry of the scene, because if you were to make it into 3D projection, it would be a little bit more complicated.

07 - Recall Image Warping.srt

Now recall, we have looked at Image Warping, which is an attempt of trying to figure out how to transform one image to the other, which actually says that maybe now we should be looking at as a way of actually doing a warping from Projection Plane 1 to Projection Plane 2. , we had looked at things like translating an image, scaling, or Euclidean things like rotation, which includes full translation and rotating the image. But, we also looked at things like rotation and affine projective types of things. If you recall, translation required us to model two unknowns. Euclidean, which had the true translations when one rotation had three unknowns, affine, which I counted from. I had six unknowns and projective had eight unknowns.

08 - Introducing Homography.srt

So again, this is what we're interested in. A scene, and we have two projection planes. And we know from our work that we did in modeling or being able to kind of look at warps, that we need to find a new p, prime, that is kind of a warp our projection from the original p. And these are the two images we have. if you recall, we knew how to do this with an equation like this. And , this was the homogenous coordinate system, and these are trying to get to the new pixels. This is used in order to introduce a concept of homography between two images. So the idea really is, how can we relate two images, which have the same camera center? Again remember, the camera was rotating and getting newer points. And again this rotation might actually create a different plane. I refer you back to some of the earlier visuals I showed you on that one. So the basic idea and this is, this is a rectangle. And again remembering the properties of these types of equations. A rectangle should make, should map to any arbitrary quadrilateral. Again lines should remain lines. They do not have to be parallel. So the shape of this para, this region might change, but overall it will remain a quadrilateral. It'll have straight lines. So again, parallel lines won't be parallel, but lines will remain straight. As we know again, this is something we have looked at before when we looked at this whole equation. Now, you may remember, that i in this one is always equal to 1, and we have eight parameters that we need to now model.

09 - Computing Homography.srt

So let's think about how we can compute homography. Let's start off, my two images from my scene again. Again, this is the Lord's Cricket Ground and what I'm going to do is I'm going to focus in on a specific region. Let's say this region here and this region here. The reason I'm actually picking these is because there's a nice planar rectangle in that region and we can actually use that as an example. Let's zoom these regions up. So we'll take this as one of our images and the second one would be this one and let's look at them a little bit more carefully. So here is my two regions, zoom and zoomed in the little bit of the left panel. It's done to kind of find this one. This is my equation. We know everything about it by now. What we're really interested in computing is a new P-prime from using the transformation from the original piece. Let's find four points in this one and I did say there was a reason I found this region because now I can actually find four points at the corners of this sign that was on the grass. I can find the same four points here. So, P-prime would be here. P is here, all right? So these are all xys and these would be, , in my new coordinate system, x prime, y prime, using just this equation, homogenous coordinates. So, again, all ps and all P-primes. So to compute the homography, H here, given pairs of corresponding points in the two images, we need to set up a set of equations, where the parameters of H are unknown. So, in essence, these are my two sets. I can most probably get the information about where these locations are. What I actually don't know is what H would be between those two images, right? So that's what we want to compute. We want to actually model the transformation that goes from this to this because if I know this transformation, I can use this for a variety of things.

10 - Solving for a Homography.srt

So let's see how we can actually compute that. Again remember, we are looking for eight parameters. This is equal to 1 which is a scale factor and always known to be equal to one. We need to set up a system of linear equations like so where Is equal to b. And how would we do this? Well, h is our unknowns. So what we would need to know is, where is the vector of Nodes h is all of the values a, b, c, d, e, f, g, h, the eight things, and that's my vector. What would be the other two terms? Well, we need eight equations. But more the better. So these are eight equations now. But if you had more of them, what, how would we get more of them? We actually can sample information from the images ourselves. So, can actually compute all of this, but what we want to do is, if we had more of them, we can actually come up with better solution. So, we need to solve for h here. I need to know A, and I need to know b. And again those things are available because I have the pair of images. And of, , if it's an over-constraint, I can actually get more samples. What I can do is makes this into a least squares solution, where I'll solve it for a lot of information and look for the minimum of the least squares solution.

11 - Warping into a shared coordinate space.srt

So this allows us to now start knowing what H is, and then using that for being able to now find a warp between the two different images. So again let's take these two different images here. These two points that I want to actually be able to model, and , these two points are visible here. To achieve this first I need to create a bigger space, coordinate space like this one. Then I can put this one here. Now I know these two points are aligned. You can put this on top of each other that way. And, , this means now actually I'm enable to align these two images. because I know where these two points are. Course, we need to do a little bit more. We need to also do a little bit of warping and interpolation, something we have looked at before, to be able to align this image with the other one. , I need to find the fact that this is the original image that I actually have the transformations for. And, , knowing this warp and interpolation parameters, I can now generate a much cleaner image between those two regions. And that will allow me to kind of create a much smoother type of an image between those two.

12 - Dealing with BAD matches.srt

Now, let's talk about the fact this is not that trivial all the time. We are dealing with feature detection which does have some times problems and sometimes you know why there are problems with feature imaging. Let's look at two images. I'm going to zoom in again just to kind of show you more details. So this is my right image, and my left image. Now, what I can start doing is , finding features. So I found one here and, , this also here and I can match it. I can keep doing this for a variety of features. Another one here. And , if we look at it, decent feature detection will do fine for some of these types of things. Another feature match between these two points also works. Now, what I'm doing is I'm just, kind of, running a feature matching algorithm, finding features, and seeing if they match between two images. Notice each one of them, right now, looks pretty good. And the transformation looks to be correct. All of them seem to be moving approximately by the same amount. Remember? These are cameras moving. And these are images moving because we have kept the point the same. So we expect these row, these arrows that kind of show the distance between these features to be the same, right? And the same thing is true for this one. But if we keep doing this, we will find, and there will be instances, there will be features like this matches to this. Now this looks like a bad match to anyone. Especially if you consider the fact that all of the other ones were you know, the distance between them was this much. Another bad match here. So we want to be able to discount these matches. And here you, you can see it, this is a bad match, so we want to be able to kind of look for good matches, we refer to them as inliers, and ignore the bad matches, referred to as outliers. So, this can be done by a process called RANSAC, or random sample consensus, which means randomly we're going to sample things. But at the same time we're going to see which one of them are actually creating a consensus. Which one of them are more popular. Just looking at this you can see that the yellow ones are more popular, there are more of them. They will actually be the inliers, and they will actually kind of out vote I'll popularize the red ones, which are outliers. So, we want to build consensus, but we want to do this randomly, not to have a bias. So, in essence, what it means is, start with one match and then find all of the other ones that match it. These are the inliers. Then find the average translation vector, of these inliers, which in this case would be this line here. Notice this line is shorter. Much more similar to all of the yellow ones. Not anywhere close to the red one. And now this gives me an algorithm to be able to kind of compute which matches inliers. Lets look at that algorithm for RANSAC next.

13 - RANSAC.srt

So first we want to do is compute H, but loop until we get the most popular H. How to we start? Well, select four feature points at random. Compute the homography between all of them, exactly. But then start looking for the sum of squared differences between the inlier for the new one. And, , the one that's computed, with respect to some threshold. Keep the largest set of inliers. And then re-compute the H estimate within all of the inliers. And the basic idea here is that there are more inliers than outliers. Well, that doesn't really help. The idea really is, is which outliers are wrong from a set that is actually much more correct. So, giving more consensus or more power, more popularity to the ones that actually are matching each other is what the key is, not that there are a number of samples more. But there could be a lot of samples, but it's going to go choose the one that are actually more similar to each other and that'll be the one that we want to look for.

14 - Panorama Demo.srt

So now let me show you a simple example of combining two images that have a different perspective, that is, they're actually taken at two different viewpoints, but with some overlap. And how we can use those two images and go through the whole pipeline of finding the right features and matching them and using that to align and create a simple panorama, just from two images. , we start off by first loading in numpy and copy, I'll open CV. I'm going to load two images. I'm going to load the Einstein image and then I'm going to load the da Vinci image I'm going to print some information about these two images. And , also show the original images. So to help the computation, I'm going to convert these images, which are in color into grey scale. That , helps with your computation time. And , I should be able to now find the features in a much more efficient manner here. So we are going to do that. We applied this to both the first image and the second image. So here is this line of code here is initializing the ability to find features in images. Here we're actually going to use the ORB function to be able to find images features and images. There is another method available in a different version for OpenCV, which actually uses SIFT, the algorithm that we have discussed in detail. That's available in a different version of OpenCV, that's not available in this version that I'm using. Both of them actually give you feature matching that actually can be used for this application that we're interested in. So, after we've initialized the feature detector, let's start using that approach or this method to identify and locate keypoints and then use those to match them. So here in these two lines , using orb. We have detected and computed features for image one and image two, we put them in these two different data structures. Here, I can actually also go ahead and print them out to see what these things look like in terms of what feature points are extracted. These two lines of code are used to actually now draw circles around those keypoints. This is a good debugging tool, it'll tell us more about how we can actually visualize where those keypoints are. And , then we think to display them. So by using a BFMatcher function, I'm going to now start get ready to do some simple matching of these features. We're going to create two sequences of corresponding match points and then look through the whole process and find matching feature points that we've identified using the orb feature detector. Here , now those are accumulated into the two different points arrays. And then using that, we will compute the homography for both points two and point t. And , here, we're using the RANSAC algorithm, which I've also talked about in this lecture. After this we get the homography matrix, we'll print that out to see what it looks like. Finally, in this piece of code, we are going to create a panorama. We are going to take the homography matrix and actually use that to generate a shape of the panorama which is where we can actually project all of the small images into. And using that, warp the images apply the transformations that we have actually now computed. Remember, the homography matrix is here, which can be applied to the whole size of the image. And using that, we will actually now generate a new image and copy information into that image that actually has all the panoramic pixels. And , showcase this result. Let's look at the solutions. Couple of things to just help us debug things. We have two images, both of them are the same size. We actually used the orb detector to find 500 keypoints each in one of the images. Then we went ahead and found matches, 305 matches were found of the 500 keypoints in both of them. Using those matches we actually computed The homography of the transformation matrix. This is now just displaying the images, so this was the first image. The second image. The circles here show where the keypoints were found using the orb detector in the first image. And the circles here, showcase all of the interesting points or keypoints found in the second image. Notice again, this is the overlapping part and this is the final panorama, which is actually now, if you notice combining these two images here. In this image or in this code, we haven't done anything about accurate blending of this kinds of stuff. So, if you pay attention and look hard, you might be able to find a seam. And , it's the one seam here. Again, remember all the stuff we've talked about with blending and cuts that could be used to make this better.

15 - Warp example.srt

Now this you've seen before. I'm just going to show this again. This is how we can actually make all of this wonderful stuff happen. Here, you see the multiple images, they've been all warped together, based on the kind of stuff we've talked about. Again, if you notice, in this camera, they were all taken from the same projection point rotating the camera this way. And now we have all of these images. Each one of them has been warped. And registered on top of each other. Here we kind of, see them on top of each other, but as I move them around, you can see that they're all perfectly aligned. This is the this is the software I used to do this. There are many different types of software , you can use this, use your cells for a variety of things.

16 - Not just a plane.srt

One thing to point out. We did start off by saying that this is a plane, and we put a camera here. I rotate a camera about a projection point, we create a planar surface or a mosaic. , this projection plane can also be a cylinder. Or a sphere which would have curvatures on both sides. by knowing this kind of stuff we would actually be able to then also do the similar kinds of projections. So here I'm just rotating around a plane. But as I said, I could rotate in this way, capture a cylinder, which has this thing. And also, in many instances, move the camera up and down and also represent a sphere. So in essence be able go up and down. And this allows us to be able to generate different types of panoramas. And we've looked at this kind of stuff before. This showcases three different types of panoramas. A planar panorama, which is trying to make straight lines be straight lines. , you notice that all the lines are straight here. But this is more of a spherical panorama, where the lines in the middle are straight, but if you look them, they curve around pretty badly at the edges, and stuff like that. And, cylindrical panorama is like this, where, again, we can actually model these types of things correctly.

17 - Finding Panoramas.srt

Now let's look at this whole additional concept of finding panoramas. One of the restrictions of the way we've talked about panoramas so far has been that you have to take a sequence of pictures, panning left to right and in different formats. >> Well, that sound rather restrictive. So, what we want to be able to do is kind of find these panoramas from a collection of pictures. This actually also has been studied, basic idea is that we want to be able to find similar you know, find images from, and run RANSAC and find the most similar types of these patches, and say okay, these two images are the most closest to each other. Let's actually build a panorama on this one. Similarly, find another region that might have you know, similarities and use that, and keep going that way, and third one, , would be again, similar ones, and in essence, what we do is we will use RANSAC and other types of matching techniques, we can find images that are next to each other and we use that to form a panorama. So we don't really have to worry about taking pictures in a sequence. This was a method proposed by at Brown and Lowe and actually it's one of the papers I would like you to look at on your own and use that as a method of trying to understand how we can actually do this kind of stuff with taking pictures that are not actually just in sequence but can be taken in any order.

18 - Summary.srt

So to quickly summarize this part of how to build panoramas, we went into details what five steps are used to generate panorama. Talked about image re-projections, specifically with, for the cases of panorama building. Introduced the whole concept of homography and how it can be computed from two pairs of images. And then also introduced the whole concept of RANSAC, to allow to deal with good matches and bad matches and that could be used for a variety of things including assisting us with being able to find the most reliable features, and also, perhaps finding panoramas, and then we talked a little bit about additional things that we need to know about panoramas, including projection models and stuff like that. Again, this is just scraping the surface on the whole idea of panoramas but again, there are lots of other readings that I'm going to be asking you to look at that'll help you build a better understanding of these panoramas and we'll do a simpler assignment on panorama building but also know there are lots of software out there that you can use to learn how to do this kind of stuff. Additional reading that you should look at include these. There is a whole lot of literature out there I encourage you to look at. There's also publicly available software that you can play around with, and also commercially available software.

# 05-04 High Dynamic Range.txt

01 - Intro.srt

A classic application of computation photography is, , generating a high dynamic range image. In this lesson, first I'll introduce to you, the concept of, what is high dynamic range? And then, we'll actually go through the steps of building a high dynamic range image from a sequence of images.

02 - Lesson Objectives.srt

So the objectives of this lesson are for you to learn about what is Dynamic Range. We will specifically talk about Dynamic Range as a concept that applies to images, and how we want to be able to capture real scenes with the lighting and radiometric information of the scene to capture the best possible image. We'll also think about what makes or prevents a camera to be able to capture dynamic range correctly. Specifically talking about digital cameras, and how they encode information? Or do not encode the information of dynamic range of an image. We will talk about the Image Acquisition Pipeline. Which is aimed at capturing the scene brightness, the scene radiance, and covert them to pixel values. And , those are the pixel values. We can do various types of mathematics on to generate newer types of images. We will look at the variety of a linear and non-linear aspects that are inherent in an image acquisition pipeline. That will allow us to start thinking what, what kind of mapping we want to model and build upon. In this lesson, we will also cover aspects of camera calibration. That will allow us to calibrate a camera, so we can actually be able to get the right kinds of levels of images and colors, that are actually visible in the scene, into a real image. So the goal here is to calibrate a camera, so we will be able to look at the exact colors in a scene, and be able to capture all of those to replicate them in an image. We will then discuss how we can take values, the pixel values of intensities from different exposures of an image. Again, this is where a camera comes in, how we'll use the exposure? You know, to remember things like aperture and shutter, to then actually capture the light information, and then use that to generate a new image of the scene. And then we'll talk about Tone Mapping. Which is aimed at taking a high dynamic range image and then converting it in a form that would be made visible on the traditional display or perhaps even bring it out so you can actually see all the different types of dynamic range in that image.

03 - Dynamic Range in Real World.srt

So first let's talk about what a dynamic range is in the real world and what kind of images we're talking about. Let's use that to situate our problem. I'm going to show you a bunch of pictures that I took all around my house a while ago. it was a sunny day so I was able to capture images under the sun, but at the same time I was able to find different dark corners. And the goal here is to showcase to you a variety of pictures taken under different lighting conditions. So for example this is my first image where I took in a dark corner. I took it inside, there are no lights, you know, kind of, illuminating any part of that scene. And even with the long exposure you can see it's a dark image with nothing there. Second image I took was again inside, but with regular inside incandescent bulbs. You can see a little bit of the scene, it's a little bit kind of darkish orangish, reddish kind of stuff are visible, but you can see that there's not much detail here. Moving up, then we go, still remain inside, but this time I'm actually near a window, which means that the scene is now naturally lit. So there's a little bit of detail in the scene. Now you can actually see the character and the book and everything. Next one I now move the same two objects outside, but this time I'm under a shade. Again you can notice that from going here the amount of light is increasing. Finally move to the outside, now this time under the sun , I've intentionally made it still that there are no shadows or anything else like that. But you can see now the scene is much, much better lit. in none of these images I'm playing around with any external lighting source like flash or anything else like that. Another whole instance would be as I can now just take the camera and point it straight into the sun. the complete opposite of this dark image here, everything is just completely bright. So you notice now that, you know, there are lots of different types of lighting conditions, and the light actually plays a role in what kinds of images we capture. And we've discussed this when we talked about illumination. But now let's try to understand and quantify some of this, so we can understand, what are the different ranges of light intensity that's actually in the natural environment that we're trying to capture?

04 - Dynamic Range.srt

To help us understand dynamic range, let's first define the term luminance. Luminance is a measure of the intensity of light per unit area. And this is again, kind of accounting for also the light traveling in a given direction. So any time it, light is hitting any surface area, the direction it's coming in from, and the luminous intensity per unit square area of that region is what we're looking for, and that's what the measure of luminance is. And it's ca, measured in candela per meter squared, cd over m squared. To help us quantify this, let's look at the whole range of luminance. Now I know we've, in the past, always looked at a black and white image or a range of zero to one, or zero to 50, to 255 in a different set of numbers. But now, we're going to look at it in terms of luminance, on a surface, or an object, or a scene. And in this case, let's look at these values. And again , I'm showing you that this is a log plot, going from eight, six, four, two, zero, minus ten, minus four, and minus six. So, , that's why it kind of has a linear thing. But these are again, numbers that are much more in detail, kind of showing you things about what's happening in an image. And I've just now marked in this range in indoor, sunlight, sunlight is much brighter, and in fact, there's even something much even brighter than that. Indoor images would have these type of ranges here. So this point is showing sunlight. This would be indoor. These are the kind of images, we looked at it. If it, this scene was lit by just moonlight, directly captured, that's what we would have. And, , this is starlight. Now, one thing to note, the human vision system, can measure static contrast ratio. That is, being able to kind of see the range at any moment, when I'm looking at a scene, are from hundred to one, so ten raised two to one. the human eye, in a static case, can also adjust itself by about 6.5 f-stops. Remember the term of f-stops, when we looked at from the camera where, when we actually looked at what were the f-stops. And we looked at a variety of ways of what of happens. And that means is, how wide the aperture is opening up that allows us to capture more light, or less light, depending on the size of the aperture. Now, I do categorize this in two different ways. One, I'm talking about static contrast ratio. Now, another one is dynamic contrast ratio. That is meant for if the scene is dynamically changing. And in this case, our human vision system can actually do much better. And in fact, we see a range from 1 million, ten raised to six, to one. And , this can be captured with about 20 different f-stops. So you notice, the range goes from 6.5 to 20, when we go from static to dynamic. Now the big difference that I want to point out between static contrast and the static scene is, nothing is changing. With dynamic is when the, the illumination in the scene is changing. Now this could be because the scene is dynamically changing, or somebody is changing the lighting conditions, and this could be again, the sun is moving around, there's shadows and stuff like that. So that's the difference. , human eye system, as we've talked about before, is a pretty impressive bit of technology, and it can adapt quite quickly to a variety of shades. something to think about when you start thinking about the human saccade system, and I encourage you to look at that kind of stuff on the web, because it allows you to kind of adapt very quickly from one brightness to the other. Now partly what we are trying to do with this representation of dynamic range, is to create images that will allow us to actually capture dynamic range in one static image. That actually is somewhat similar to what our human vision can, system can do. By moving our eyes around in saccades and stuff like that, we can see a lot of dynamic range, and our eyes adapt very much to bright and low light situations. And we can see much more detail. , cameras at present cannot do this, and that's one of the questions for computational photography is, how do we kind of, bring in the computation to make it do the adaptation to what it's seeing in the light, and being able to capture images? And , at the end of it, we want to create sometimes a, just a static image that captures it. You'll see more examples of this as we continue talking more.

05 - Limited Dynamic Range of Current Cameras.srt

Let me give an example of the limited dynamic range of current cameras. Here's is an image I took again, in my home. Here, , a short exposure, remembering again the concepts of exposure that we looked at from exposure triangle and stuff when we talk about cameras, and here if you look at it, you should be able to see, it's a dark scene but outside you can see a little bit of snow. I took this picture on you know, one of the rare snowy days in Atlanta but it's a short exposure, you can't see any of the details inside but look outside, you can see a lot of snow and brightness. Same image or same scene, different exposure values. Long exposure, , all of the insides are nice and visible now but you might think that there's a lot of overexposure going on at this point and the details outside are not good. So this is an underexposed image, this is an overexposed image. Underexposed because lots of pixels did not get any detail, lots of black pixels here. Overexposed, a lot of white, bright pixels. So you, if you were to build histograms of these images, you would start seeing a lot more on the white pixels and for this one, you'd see a lot more black pixels. Again note, both of them are exactly the same scene. They are not even a different time of the day. They were taken seconds after each other and that's the kind of stuff that old cameras give you right now, and that is a limited dynamic range and now, , we're interested in its capturing the range from here to there. To capture this dynamic range, we need 5-10 million values to store all the brightness around us. Remember the luminosity stuff that I showed you. The scale information was significant. Here we want to able to capture all of this in one image. Problem is, and recall when we started talking about images, most images capture each of the three different channels in basic 8 bit images, values of 0 to 255, sometime just put in values of 0 to 1.0 but again, there are only 256 values as opposed to a much brighter range that would cover the range from here to there. Nowhere close to, , being able to capture 5 to 10 million values. Just to showcase this, let's look at this example again. Here, if you notice, I'm showing you the high dynamic range. In the real world, this was a dynamic range from 10 raised to minus 6 to 10 raised to 6, and that's what we're trying to look at. , on a photograph, what we would most likely do is just showcase this one and that's what is shown here. I'm showing all the values that are much more on the brighter side and focusing it here. The other end of the spectrum is the long exposure and , this is the long exposure, and mention again a lot more bright values and here what we'll be doing in this dynamic range is most probably just capture the region here and pack it into my 0 to 255 values. Again, a lot of detail would be lost. Here, I'm showing you more of the information on this side of the spectrum.

06 - Relationship Between Image and Scene Brightness.srt

Now, lets look at the Image Acquisition Pipeline. This is something again, we've discussed differently, when we talked about cameras, but now, lets talk about it, in detail in this context. , we start off with a 3D scene. And this scene is being captured, and the, , the capturing in, information is called the Scene Radiance referred to as L. And , what the units here describe is watts per steradian meters squared. Steradian is a measure of the solid angle, and because these are 3D scenes, and the light is coming out, in form of a cone from everywhere the basic idea we want to do is use the cone information, to capture the light. So in essence what that means is if I have a scene, what I will be doing is using a cone. And the steradian is , kind of, counting for all of the information that's in this cone, from any point. And that's , allows us to kind of have an area, from that point onwards. To just reiterate scene radiance, and referred to as L, is , watts, which is energy per steradian meters square. Steradian is the measure of the solid angle. Primarily because again, this is a 3D scene, and the light is coming out in form of a cone, from everywhere. , when we have a 3D scene, to capture an image from it, what do we need? , Optics. So here, now, we use the out Lens, and Optics, and that converts all of this 3D light information. That's why, we have the solid angle, and the cone , coming in. And now, , we have a 2D sensor. So we, , do not have the information from steradians, but we do have watt per meter square. And we know, that was something we referred to as Sensor Irradiance, and it's something we ip, in, labeling as E. This is a Linear mapping. Once I know this, and if have this information, when it hits the screen, I can actually now, or a sensor, I know, what this measure will be. Next stage in the pipeline , is the Shutter, because shutter is the amount of time, a light is allowed into my sensor. And we know that , E times delta t gives us the information towards getting, the sensor exposure. And you may recall , that we refer to the exposure as H, which was equal to the sensor irradiance, and amount of time, the shutter was open. And this is something again, we've looked at before, so in essence we'll be going from 3D scene, and now, we're getting sensor information of the exposure values. Continuing on, we now, will have a sensor, which would be a CCD, again, we've covered the details of how a CCD works. Light comes into a CCD, and what we , get is electrons collecting, and depletion layer kind of, collects it. In essence, what we're doing is computing the voltages, at different types of capacitors, within a CCD. Next step , is an analog to digital converter, , takes these voltages, and converts them into a digital values, to give you a raw image. And , then we want to do some sort of remapping, from the raw image, again, if you were doing camera roll, you would just use this information. But sometimes you , get compressed pixel information of intensities, and that's where we get the intensities artifact. We have covered each, and every aspect of this earlier, and you can refer back to the previous lectures on this one. But the bottom line is, from 3D scene, once we have the scene radiance, there's a linear mapping, to get sensor irradiance. There's another linear mapping based on again, the time opening to get the exposures. And then, there is a bunch of different operations that happen, at the sensor level. So, here is one thing we want to note here is, this whole process of the pipeline is linear. There are lots of linear mappings going on. This part could all be non-linear, right? I mean, there are no dir, direct linear equations that would actually, let us predict this analysis here. And they're also sometimes depends on the kinds of sensors, and stuff. But this is linear, and now, we need to kind of understand, how we can use this to understand, and gain information, from what's in the scene, to what we want on the sensor, in terms of an exposure. So far, we've studied the image acquisition pipeline, now, let's look at some of the mathematics associated with it. So, , this whole pipeline starts off with L which is the scene radiance, a linear mapping to E, a linear mapping to exposure, and then, , all the way down here, to give me intensity values. , we're interested in this pipeline, but we are also interested in the inverse of this pipeline, which lets me compute the inverse. Now, we are , what I am interested in, is I have an image, can I actually now, predict the model of the whole scene intensities here, in terms of scene radiance? So, I'm interested in both of those. And that's the kind of stuff now let's think about, to be able to model.

07 - Camera Calibration.srt

First let's talk a little bit about camera calibration. There are two different types of camera calibration. First is a geometric calibration, where we are interested when we look at a 3D scene is how each and every pixel in the real world is relating to directions, angles, shapes, you know, any kind of geometric information in the real world. So that requires us to now calibrate the camera and the location of the camera with respect to the scenes and objects in a scene, so we can actually get more 3D information. That's geometric. , we're also interested, remember from our earlier lectures, to be able to capture radiometric, photometric, information from a scene you know, like reflectances and scattering of light information. Here, , we're interested in each and every pixel related to the radiance amounts in the real world. You know, what's happening in a scene? What's the exact radiance value at any point in a scene? How can we capture it onto a pixel in an image? Again, this is the whole pipeline we just saw, where we want to be able to go from l all the way to i. Now, we study this a little bit when we did panoramas. Sometimes, we might not have ability to capture everything in a 3D world directly from the sensors so what we want to do now is take a little bit of a data-driven approach. , that suggests is let's actually do this by looking at a bunch of images, not just one image, and use that information as to how things change when I take a bunch of different images. So, for example, here, I may have a color pattern, which tells me everything else about it but I could take a color pattern, a checkered pattern, and move it around and as I move it around, if knowing a little bit more about the information in that scene, in this case, , the exact nature of the structure pattern associated here and if I look at it from the same camera, I should be able to kind of now get different data, at different points. That suggests is that now I can get how pixel relates to the geometry in a scene by just looking at a whole lot of other images. Well, the same thing is true in the case of photometric. This was relevant when we looked at the whole case of paranomas. This is more relevant in the kind of world of HDR, where now we're trying to relate is how the radiance of any point on the scene is accounting for a specific pixel intensity and we do this by calibrating a camera by getting a lot of data by the same scene by having different images. Remember, again, these were both issues that help us when we start thinking about concepts like epsilon photography.

08 - Radiometric Calibration.srt

So how do we do Radiometric Calibration? Again, we're interested in this pipeline of going from L towards I, but we also want to be able to model the inverse phenomenon of it. So let's start off with a Color Chart, that we know the reflectances of. What means is a color chart like this comes in predefined and here , just I'm showing you the gray values here. It shows what are the intensities, 90, 59.1%. So somebody had sat down and calibrated this color chart as a perfect color model or gray scale values that we're interested in. What we want to do is we want to take multiple exposures from this co, color chart. Which means is I'm going to take images of this. What I can do is I can take them at different exposure values. That means I can, you know, do things like ep, epsilon photography. I can take images at different exposures, shutter speeds, and all that kind of stuff. That would change how, what kind of intensities for the same image that is out there. , we're playing around on this parameter here to oh, different types of parameters to get to this value. I'm just showing you an example of this. So, again, my pixel values go from zero to 55. This is my scene irradiance. I'm going to look at irradiance, because we know scene radiance to ir, irradiance is a linear things and actually we're more interested in trying to get there. This mapping is something which is linear. , when done, we know what's going on. , this means is now everything's projected on a 2D surface, which the information is coming from. So, , if I know the scene irradiance I can plot a point, as I go at different scene irradiance. And I can figure out from this one what is the intensity at each and every pixel. So next part that we can do now, , is after I've drawn this curve, for a specific device or a sensor, and imaging device like the camera I've done this on by taking in this case, one image, two image, three, four, five, six, seven images and I can fill in the curve. Now I can do the opposite, right. If I know a pixel value, or I want to, curious about a pixel value, can I actually now compute the scene irradiance? And that's what this exercise shows. Once I can model this, I can always come back and I say, okay, what's here? I can figure out the scene irradiance at this point. Again, scene irradiance here is going from zero to one. Now, this will allow me to pick up any values I want. So, , there are few assumptions here, in, when we play around with images like this. It assumes constant lighting, and all patches, , are equally well lit. So it can't be a focused light source. The assumes of light source is really far away, and is equally lighting the whole scene. So, you know, for example, something like natural sunlight. When you light with this kind of stuff we're allowed to do this. A unique inverse exists for g because in this case g is monotonic and smooth for all types of cameras. And again we will model this for different cameras and this allows us to compute this.

09 - Exposure Example.srt

Let me a show you a series of examples of what it means to take these now images at different exposure values, to be able to generate something equivalent to a stack of images at different exposure values here, we're giving it a lot of different samples. Again, an underexposed dark image, you can see the outside. Now, , I've made it a little brighter, more, more, and now you can actually see in detail. This was always visible but now more of the insides are becoming visible and as we keep going, you'll notice that outside gets overexposed and some of the insides are also getting a little overexposed and detail is kind of getting lost. You're getting a little bit of a halo effect here and all that kind of stuff too. Even more now, actually, the boundaries are getting lost. So let's save this as my stack of images. What we did in the previous case was we calibrated but this would, , could be done for a real image. Again, in this case, what I did was I took different images and note that in this instance, I actually have all of these images aligned because I put them on a tripod. If they were not on a tripod, what you would've had to do is, , do image alignment. Remember the stuff that we looked at when we talked about panoramas, as we'd have to align these images together.

10 - Series of Images.srt

Let's take another example to now start building the kinds of curves that we're interested in from real data. Just to keep things simple, rather than use my stack of real images, I'm just going to showcase these small types of ramp gray scale images here. And I've have, kind of simulated them to have a shutter speed of 1/64, darker image, 1/16, 1/4, 1 sec and 4 secs. It gets from darker to brighter. Again, we're interested in getting to I, the intensities. G is my function, and the thing that we want to extract from is exposure H, right? I mean that is the relationship, because remember the rest of it we actually were able to get to from the camera itself. So now to get an exposure we would base, have to compute the irradiance, E, and with the time and we know all of this . We can also do this in log because that actually makes our curves and everything appear nice. And just, I'm showing these pixel values here. And I'm going to create an axis, the log of an exposure H. What we can do now is find, let's say, three points, one, two and three here. At three different points, I've colored them differently so we can see where they are. And I now in this instance can pick out what their values would be. Similarly now I can find three similar exact points in another image. And I'm actually just randomly walking around in these five images. And if you notice that I can find these three points in all of these images. You know how to get these images, feature tracking and feature detection work, or feature matching work that we looked at allows you to find these types of things in a real image. This is just a synthetic test case that I'm showing you. , after I find these points, I can actually use this to plot these values. , I'm showing this purple here. While I just have five images here, you can imagine I may have more. And this would allow me to get a bunch of different values here. So here you have one, two, three, four, five, six, seven, eight, nine, ten, eleven, and so on. This allows me to kind of, you know, blot these out. I can do the same for turquoise. And here is the range change that's happening in this one. Pixel values, log of Exposure. Based on this calculations again, pure data. And in case of the yellow, we see this kind of phenomenon. Where again we have points like this. So I took a lot of samples, over ten plus, and this is what it gave me.

11 - Response Curves.srt

We can use this to now start building response curves. So this was what we had in the last slide, which had the log of exposure, pixel values, and these were the three different points. And we were able to trace them out in equally, all three of them. Now assuming unit radiance for each pixel, we can kind of start making an interesting model here. So what we want to do now is take this and start kind of figuring out how to align them so then they would have a uniform shape. So in essence what I took. The purple values and I want to be able to shift the turquoise and the yellow to create a unique profile. And what that means is I'm adjusting the radiances to open, to obtain a smooth response curve. So I take all three of them, I adjust them out a little bit to create a nice, smooth response curve. Once I have this I have actually gone far in trying to be able to create an inverse G, purely from there. Again, remember, G is giving me this information. I want to be able to get intensity values and come the other way.

12 - How to Compute.srt

So how do we compute this? Well, first, we want to be able to create a discrete inverse response function from the data. And we refer to this as, let's say, say G of Z, and what we're interested in for each pixel site, and I just showed you three in the last image In image j. I just showed you five, and I showed the plot was, were about 15 of them. For each pixel site, and we can choose a few good number of them, all of the images, we will compute the values, and again we do this I'm in log here, of both the exposures, And, a time, to compute a function g(z). I mean, doing in this log allows us to do addition. Otherwise, remember we would be doing this as a multiplication. , what we have now is an overdetermined linear system for N pixels over P different exposure images. So again, we have P different exposure images and we have N pixels. We started off with I, side pixel I and image J. Now we have a number of pixels that we want to measure, and the number of exposures. Remember, last time we showed 5, we had 15, and I had 3 points. So N was 3, and P was 15 to create those plots. Now, , what we can do now is create a system of linear equations, log of exposure, log of the time itself. And then, , subtracting by this and this lets us do, you know, these squared difference. , what we now do is come up with a fitting term and a smoothness term. And this is the, kind of way we solve these types of linear optimization problems.

13 - Estimated Response Curves.srt

So this allows us to now do the computation. And now let's look at few curves. Now here I'm just going to show you some curves from a different set of images, not these, but just this is just to prove a point. Here you see the four curves. This one is for green. This is for red. For blue and, and then all three of them combined to rgb. This is to point out that we can actually do this analysis separately for all of the different channels and then combine it separately. Again just showing you six images. Number of images increases, we can actually do this. Most of the time for simple, smaller hand-held cameras. They only take about two or three images. Higher end cameras and using functions like bracketing you can actually capture a lot more of these images. So that is the capital P in my previous equation

14 - Radiance Map.srt

This is a radiance map now of the image we captured. What comes out now is we're no longer in the range of 0 to 255. This output shows that this image now has a range from 0.6215 to 12, 871 intensity. All of the dark blue colors that you see are close to this one. All of the bright red, and , when you're seeing the bright red here and you actually notice that there's a little bit of yellow because as you notice in the image, this was kind of getting very bright too, but you see a lot of detail and there's kind of really bright spots. They're out there. , now we need to study how we're going to take this radiance map and create an image. Before we go there, we also need to think a little bit about what kinds of file formats we can use to save images like this. Now that radiance image I showed you, right, needs to be stored somewhere. So far, we've been only as interested in trying to say, okay, I'm going to have 8 bits per red, 8 bits per green and 8 bits for blue. Well, now we need a newer form of an image, 32 bits per pixel, because now we want to actually create 8 bits of additional information. This is 24 bits, 8 bits per color, so we're going to add an exponent. Math works out the following way. , what it means is now, if I have an RGB value of 145, 215, 87, I'm going to actually add another exponent number here and what this exponent does is take the RGB values, multiplies with the 2, raise 2, the difference of 149 minus 128. One-twenty-eight was, again, remember the 20 to 256 or 255, that white part. , and this gives us values much bigger than 0 to 255 just by adding this 8 bit of information. , the other way is also there. Rather than take this value, I can subtract the 103, this information, from 128 and this is this value here and now I have much smaller numbers. So now this allows us go from very small numbers all the way up to larger numbers and, , this format can be used to save an image like this into a file. This is a proposed representation of a file format from Ward. There are many other similar formats. I do encourage you to look at radiance map formats on the web.

15 - Now to Display it.srt

Now, how do we display it? This is the image I got, captures a lot more gory detail. It's no longer from 0 to 255. What I want, , is an image that can be still displayed because displays also have the same constraints as cameras and printers do that. I still need to be able to put up colors between 0 to 255 for each and every one of the channel's RGB. So this is what we want out of it. Just to zoom in, here is the image that you should now actually have the details here, right? You can see details of the painting. They're all well lit. You can see the snow outside and it's well lit outside. You can even see, for example, the umbrella outside in detail but you can also see details of the floor and everything else like that. I personally love this image it's a nice, , background and backdrop. We'll be making these example images available for you to play around with too.

16 - Tone Mapping.srt

So, the process that now we want to leverage is called Tone Mapping. Tone mapping is an attempt to be able to take an image, and high radiance image, like the one we talked about the radiance map, and converting it to a space where we can now actually visualize it. So what basic thing we need to do is we want to map one set of colors to another in a reduced space. And we want to account for being able to display it on a medium that has a limited dynamic range. So again even displays actually are built the same way as sensors. They could, they should actually be able to go from 0 to 55. We don't want to be able to dynamically, and get the colors in the space that a display can show it, and again as a set-up printer. And a variety of things exist on this kind of technology. And , we want to be able to display these things on printers, monitors, and projectors. This is primarily to address the fact that printers and most of the displays right now are inadequate, in terms of how they can represent it. And , these days with 4k displays and all that kind of stuff. We can actually really display a lot of high dynamic range information. But, the content for those types of displays is hard. So Tone Mapping addresses the problem of being able to get this contrast reduction from the scene radiance image, which is captured in the radiance map to a displayable range. It preserves the image details and color appearance, because remember, what we did with the HDR process was really capture the radiance map from a scene. But we can't display it, so what we want to do is convert it into a form and that's what tone mapping does. Many, many well known algorithms are existing for this kind of an approach. We'll be discussing some of them in detail in this class also, and I just list a few of them. I encourage you to look at them. Now before we go on, there's one thing that I wanted to add. If you look at this image, I actually get bothered by both Tone Mapping and HDR on the web a lot. Sometimes, and this is one of the perhaps most overused imaging technology out there on the internet these days and people are actually generating images, that to me look ghostly. So this image , is capturing all of the detail. But with the clouds and all of that kind of na, natural lighting it actually feels unnatural. In fact this scene would have never been naturally done like this. So that is one of my problems with doing over you know, use it of things like HDR as we are actually generating a lot of images that don't look natural anymore. So my recommendation, use it carefully. Hopefully, you like the example I showed you. Here is another example of a similar type of an image. I mean yes, it's a dark image. It's a nice image. It looks artistic, but even in the most natural situations you would not have seen lighting like this in a square, a popular square, where ever you are. So what we're really doing with tone mapping for high dynamic range images is taking again this whole dynamic range, which is shown here in the real world, and what we want to do is squish the whole range into a 0 to 255, and in essence, that's what tone mapping does. It takes the whole range and, based on the display characteristics, and perhaps, a few things that I want to emphasize, it compacts it out into this range. So yes, you do lose information, but if you save the radiance map, you already have that information that you can use later. Things that does, it takes the limited contrast information and maps it to the medium, display medium that you're using and preserves details. Again, we'll be covering a little bit more of tone mapping online in the class.

17 - Summary.srt

So, to quickly summarize, I discussed a variety of issues of dynamic range, what it is, and relate it back to also things like how we can see some details of dynamic range, and how we can capture that kind of information in real scene. We talked about how image acquisition pipeline is aimed at capturing the scene radiance and we want to use that to convert to pixels. We talked about a variety of linear and non-linear aspects of the image acquisition pipeline. Again, remembering that really what we want to do is capture the scene radiance of a real scene, and convert it to the pixel values. And how the whole pipeline works and how we can represent and model this. Then we talked about the whole idea of camera calibration, which was aimed at , using a lot of data, a data-driven method to be able to map the information about what scene radiance was and what pixel values were. This allowed us to create a profile that can be used to then , do the opposite. Given a pixel value, what would be the best scenery radiance for it. Then we talked about to do this mapping. how do we go from pixel values to different exposure images. And then a, a new radiance map that captures a scene. Then we talked about how tone mapping could be used to take the radiance map to then display an image in the displays that we have available. There is a lot of work out there in the HDR area. I'm just listing a few of the papers here. Other papers I will make available also on the website. So you should be able to look at them carefully. These are the ones I referred to in this slide, sort of slide deck. And , there are lots of software out there. I encourage you to look at things like Exposure Fusion, that's an additional method. Builds on a whole lot of stuff that we talked about and builds on explorant photography. And these types of HDR images are now available on your cell phones.

# 05-05 Stereo.txt

01 - Intro.srt

In this lesson, I'm going to introduce the concept of stereo photography. So far, we've concentrated on individual images. We've learned a lot of techniques to process and merge these images together but now, we're going to take a specific instance of a pair of images like our two eyes that look at the same scene. What can we do with this paired, a stereo pair of images, to extract more information about a scene and then more importantly, how can we leverage it to do computational photography?

02 - Lesson Objectives.srt

The specific objectives of this lesson are for you for to learn about, the Geometry of the scene. Remember we talked about, that one of the interesting things we want to do with capturing images and photographs is to capture the geometry of the scene, that is the relative distance of objects in the scene. Here we're going to start getting into that kind of geometry and look at depth. That is how far away are pixels that are visible in the image or photograph. And also structure relays a lot of information of that scene. Going to look at specifically the concept of stereo. That is how to do this from two different viewpoints. By introducing the concept of the Parallax that could actually help us look at the whole concept of how we're going to extract depth from images. And then, talk about specifically, how can we compute depth from a stereo image pair? So let's get started.

03 - Depth of a Scene.srt

Before we go further, let me actually just talk about the concept of depth in a scene. Remember we started talking modern computation photography that we're interest in. Capturing the 3D scene that is lit by some sort of illumination sources. We want to use the optics, the sensor to capture the information that actually can be processed displayed for user to interact with. So essentially we're interested in capturing a photograph in computational photography that captures a 3D scene, and all of the related geometry. Look around yourself. You have a 3D environment, and but in essence that's what we want to capture in a photograph that shows the shape and size of things around it, and also relative information of, how they're related to each other in shape and size? So an image like this, in the real world actually has depth. That is parts of the image are in front, parts of them are in the back. Here there is some missing detail, but you can see the 3D environment or the 3D information of this scene. In this case, just shown from this image. Now missing information, we'll talk about that in a bit. But in essence, what we want to do is capture images that also can show the depth. Here is one example of subtypes of images, depth image or sometimes also referred to as a range image. All the black pixels here, show the fact that, these are all values of zero considering that's the background. And everything in front, and different intensity values show how close they are, to my viewpoint in this camera. The brightest points are the closest, and the farthest points are black. Missing information here is represented by black also. So , you really want to start capturing a 3D environment, we need depth, geometry, and 3D information. In this lecture we're going to talk about, how to capture that?

04 - Compute DepthStructure.srt

So now let's review something we had looked at before. We had started a lecture off talking about that images best represented in this coordinate frame, x and y, but an object in the real world. And I referred to them as capital X, Y, and Z. With Z is in the inside the camera lens itself, that is the depth. Y and X show the width and the height of an object. So if I had an object like this, I actually would actually capture an image and, , the coordinate frame X and Y here. We represented that scene in 3D this way, where now , I've turned it around. Z is going in this direction. Y and X are this way. And , the same object is here, shown by this arrow. And here, I'm just giving it, you know, values of X0, Y0, and Z0 in capitals. Let's build this to kind of see what the camera model would look like and how we can compute depth. These are the coordinate frames in real-world coordinates. , in a camera an image is formed. Here I show this image and the coordinates X and X and Y, sub i for image. , this image is being formed at focal length f. And if you remember when we looked at cameras we knew did all of the calculations. That is, the similar triangles to figure out that xy, xi over f is equal X0, Z0, because you know, focal length is this distance, and these are two similar triangles. And , I want to be able to use this to compute where the x would be in the image. And this would be my equation, similarly for y, Y sub i. That is the pixel information. In this image it's related to both the focal length, how far it is, and knowing information like where the Z is, how far away this object is from the scene. Notice, this information is not contained in this image, that is, where is Z. And that's the point that we want to actually now start thinking how to actually compute. So in essence, this equation would apply the same if this point was here. So this point here could actually be at this location. Same equations would apply. Also if, I can find any point in this line here, which is the ray of light that's going through this point. So, the point I wanted to emphasize here is there is a fundamental ambiguity here. That is, any point on this line in the same ray maps to the same point on the image plane, right? Any point here, , the values of Zs are different. The values of X and Y are different. But when it comes down to where there are, that is x sub i and y sub i. All of those values of any point in this ray of light is going to project exactly to this point. In essence, what that suggests is I can actually scale any of these three values, X naught, Y naught, or Z naught for the world coordinates. that means put the same scale, I mean, this, the same equation. So in essence, that suggest is there is a scaling factor here between both of these images that will actually produce the same x i, y i, and actually make sense, right? As I move closer, the object is get, getting smaller. And therefore, this would be the scaling factor. So, moving farther in, closer, would be the scaling factor, and , by multiplying and dividing by the same k, we can see that impact. So this is the ambiguity. This is the one we want to resolve in variety of different ways, and that resolution of that ambiguity will give us depth or structure of a scene.

05 - Depth Ambiguity.srt

Let me show you what this depth ambiguity how it manifests itself in real images. For example, here, you've seen lots of these types of images. Somebody being at a different distance that is, in the scale, which in this image means how far that person is from an object, and depending on where they are, you can actually see them to be the same size. , you know that Leaning of Tower of Pisa and this person are, , not the same size. You've seen other fun images like this or other images that show how big your pumpkin could be and this is actually what we also saw when we talked about cameras, right? Two different lenses actually showed different depth information. Here this person is looking bigger than this one they are exactly the same distance here but they look the same size, depending on where I am. The scaling factor that comes in from the lenses. So this is the depth ambiguity that is hard to decipher if somebody just gave me a single image and that's the point that we want to actually try to take to the next level on asking the question, how we can get depth?

06 - Depth Cues.srt

Let me actually introduce to you some concepts on, how we can get depth? If somebody gave me a single image like this, how would you compute that? Well, an interesting thing in this image is, there's perspective in this image that is, if I was to let's say, draw a line let's just draw a line connecting all these points here or I can draw a line connecting all these points here. And I can draw a line connecting all these points here. In essence, if you look at them, each and every one of the lines that I would draw is pointing that these are vanishing lines. They get closer and closer, and converging to a same point at the distance away. So, in real scenes you will always find that there are pre-determined, well defined vanishing lines and points that an image in a real scene will point to. And that start showing is there's depth. Things that are farther away are smaller, things that are closer up are larger, and by deconstructing it as you go farther into the scene they get smaller and smaller. So this is an important cue, the perspective and just looking at vanishing lines and points. Now one thing I want to remind you is, this is actually true for mostly real images and photographs. Just as an exercise I recommend trying to do the same exercise on paintings and seeing, you might actually find that sometimes vanishing lines do not converge. Here is another perspective to look at depth, right. Here I'm sharing an image. Again it's hard to, in this instance find vanishing lines. There is horizon and everything, but the vanishing lines are hard to determine. But, I know what humans usually are or what size. So by looking at this and noticing the fact that, as you go further, this same humans are, are different people. They're getting smaller and smaller. That also starts giving me the sense of the depth as I move further down this way. Right, so just objects of known sizes help me understand how I'm going to be able to perceive that. Another high level cue, and when I mean high level is it requires a lot more kind of analysis of a scene is something which could be called occlusions, right? These two people well, he's in front of this lady here. She's in front of all of the shopping stuff. She's in front of the materials and this market here. Which starts saying, is I can't see any of this because she's in front of it, I can't see anything behind her because she's occluding it. And there's a whole lot of information on how the occlusions are kind of manifesting it selves. And this kind of starts telling me more about the scene, so occlusions are also an important cue that could be used to analyze scenes. A really well known method in the computer vision is extracting information of shape of a object by just seeing how well or how differently it's lit? So here is an example of just by looking at the lighting variations on the surface, you can actually extract the shape and the depth. As I said depth as you go into the image of this face. This is an interesting example of how we can actually start looking at depth of different types of scenes, and just to showcase this, I'm going to show you something fun and interesting that one of our alums Grant Schindler worked on, and which is an app on the App Store now.

07 - Trimensional - 3D Scanner App.srt

So this is an app called Trimensional. Grant Schindler did PhD, with us at Georgia Tech and here we notice a couple of things he's doing. Just by an app like this, and he puts it in front of himself in a dark room and, , the app turns on different types of lights. And, based on how the face is lit, it actually can generate a, well, interesting 3D mesh. Now, it's not the most accurate mesh or anything like that, but you can see that it does a pretty good job. And all it does is changes how the face is lit. This is a whole concept that actually also sometimes is referred to as photometric stereo. That is, how do you photometrically change the properties of the environment and capture the variations that are coming in because of that, to extract the shape of something? So in this case , it was done for faces, just like I showed you in the previous example of shape from shading.

08 - Depth Cues Continued.srt

Another example is we can get depth cues by illuminating the scene with a structured light. In the past case, the light was the intensity of the light that will being put up and how it shaded the image was used to kind of reconstruct a shape. Here, just by figuring out the stripes and how well it's lit and using that information and, again, you notice that these stripes are very useful, because I can, now I can trace the curve to get the shape of this statue here. And this can be used to model 3D environments or 3D scenes like this. This is a widely used technique for doing range scanning of rooms and, or people and statues, and stuff like that and sometimes they even use laser lights to do this. Another method is shape from texture. Here, I'm showing an example where just looking at the texture as a change knowing that this is a regular texture. We can actually compute the shape of a cloth like this. Also something we've played around a little bit so far and talked about cameras. We can actually even get shape the focus that is by being are to focus and defocus an image. In this case focusing on this one and that one, you can actually compute depth. So here showing you the two images, focusing and defocusing. You can actually use this to compute a depth image like the one shown here. A very well known method is also using structure from motion in this case by moving the camera along, around this building and taking multiple pictures and then actually doing things like we've looked at already feature detection and stuff, making sure that these features are actually then matched across these images. I can actually construct the geometry of the scene. In this case, these cloud points are being used to used to create a model of the building that is there. It's a widely used technique, we'll actually see another example of this in a future lecture

09 - DepthShape from Single View.srt

So in essence, this sort of technique, sometimes referred to computing shape from x. We looked at examples of using perspective, shading, motion, focus, occlusions and objects. There are still many other methods that are possible to extract shape or depth or range imaging from sometimes single image or seas of images.

10 - Stereo Vision.srt

So now, let's talk about Stereo Vision. A specific form, driven by many biological systems. Humans , have two eyes. Many other animals also have two eyes. And our interest with this is, can we actually infer depth from images that are captured at the same time? But from more than one viewpoint, two different viewpoints. So here's simply an example that actually showcases this, right. I have two eyes, and inside is just kind of showing the brain. This eye is seeing a viewpoint which is kind of, you know, this view point here, and this eye is seeing another viewpoint which is shown by this cone. Both of them are different, right? One of them is seeing from this side, the other one is from this see, seeing it from this side. So just look at these cones. You can see that this eye is actually seeing, this viewpoint and actually, you're actually going to see things a little bit from this side also. Right? You can actually see the other pins of this bowling pins from this side. 'Kay, what happens from the right side? , the right side sees the right sides of these pins. The left is seeing the left sides of these pins. Actually, this is interesting so by fusing these two images one, , you're getting a little bit of sense of depth. Two, you're actually going to see more information about the scene. This is going to actually be the way we actually see 3D scenes.

11 - Why Stereo Vision.srt

So let me ask the simple question here. Why stereo vision? Here the question that really comes to mind is, why do we want to take a pair of two images taken at the same time? Pretty much model our human vision system of two eyes looking at a scene at the same time, how can that be used to generate a 3D geometric model of that scene? Again, let's start off with our simple camera model that we looked at. Here I'm going to start off again with a simple object. And in the real world, here I'm going to give it point P, capital P, with coordinates X,Y and Z in the real world. Let's first take first viewpoint, which is actually shown by this point here. And this is, , the world coordinates from this viewpoint here. And this is my image. So here, , at focal length f is an image. , it has its own coordinate axes. Just for simplicity, I'm putting the coordinate axes at the center of the image plane itself rather the corner. But you know that's a simple transformation. , , what we're trying to do is capture this whole object, which means this ray of light and really kind of coming down to small x, small y to point out this point in the image plane itself. So that's my image. And , this image parameterized with both the location and the image itself by x and y and, , the focal length, f. Now let's actually take the same instance that we had looked at before that was the ambiguity of all points that lie on this line actually being a problem because I think all of those points on this line will appear to be here on this image. So, for example this point will appear exactly here. So will this. So will this, and so will this one. So that's the problem we have. How can we mitigate this by another viewpoint? Well, let's take this as a left view and this is the right view. I have another image coming in and, , all of these points are coming to the point here this is the optical point of this image here, and converging here. But now I actually have one, two, three, four, five rays of light. This should start telling you that now we can actually resolve the ambiguity. Because all these five points actually now come in in the image plane here. But this is the only point that appears here and there. So, by just looking at these points, and doing things like triangulation, we now know exactly this is the same point that's visible from both. So no longer are we having the ambiguity that we had before. So hopefully this convinces you that having multiple viewpoints allows us to now get rid of the ambiguity and extract more information about exactly which point I'm looking at.

12 - Why Stereo Vision Continued.srt

Let's look at the whole concept of stereo vision a little bit more carefully. Here, , we have two eyes. Human vision system is driven by two eyes that are about 60 millimeter a, apart from each other. And as I've shown you before, what we're seeing is we see two different images from afterward which is slightly different vantage point. They just have a very slight difference.

13 - Parallax.srt

So let's look at the whole concept of parallax. So from these two eyes, if I was looking at a scene, and here I'm just going to create a synthetic scene here, and I'm going to put a object like a star. If the left eye was looking at the object, that would mean that a ray of light is coming out from here and going there. And , coming from this one would go here. So the ray of light actually going in to capture the scene from the right eye, sees the star and has got the red background. Which actually, would then actually show an image like this, right? Because there's a red background, there's a star in front of it, this is what you'd see from the right eye. The left eye on the other hand, would see, this. So one of the observations that are important for understanding the concept of parallaxes. That points at different depths, will displace differently. We'll get to that in a second, but you see that now both these eyes are seeing something different. And that's the kind of point that we want to make here to kind of understand what's going on. Parallax, which we'll talk about in a bit, kind of says is, because of this feature, that if this thing was further back here, it would display differently, or look differently as I would actually look at this object in relationship to it. And similarly, on the other hand, if objects were closer, they would look different to. So here, you see the relationship that because it's in front of it, it looks different because of the fact the background is different. In case both of them are the same distance, but if they were different you can actually see the difference that it'll have in our perception of depth.

14 - Depth Via Parallax.srt

Let's look at that in the context of this image. Very complicated, beautiful image here. Mountains at the back. Bunch of rocks and a small rock here. This rock is much nearer to us, much farther and then, , much farther at the back there. So, the parallax suggested that the motion of the scene features at, located at different distances will appear different. So, for example, any point or any object that's this close here is, any motion, this is going to move up here to move a lot. Right? While, all of these objects that are just a little bit would move a lot less than this. So, all of them. But, , the farthest point here is going to appear to move very, very little. So, all of the points in the mountain back here will have very small displacements because of simple motion. But the, these more distant rocks will have smaller displacement and this one will have a larger displacement. You can experiment this yourself by actually looking at a scene that has a lot of different types of elements and just looking at it and moving your head just a little bit and you will see all of the objects that are near to you seem to move a lot more than all of the objects that are farther. The same thing is, , something which you see because of perspective. Farther things are looking smaller and , they'll also have smaller displacements. So, in essence, this kind of starts telling us more about far, mid and near types of information from simple images.

15 - Stereo Photography and Stereo Viewers.srt

The concept of stereo photography has actually been around for a while. And when we talked about cameras, first we talked a lot about different types of cameras. Here is an initial type of a viewer, invented in 1838. you would put a stereo pair here. You would take two pictures of the same subject from two slightly different locations. And then use a viewer like this that would only show to each eye, the most appropriate view point from that image, one that of, of the pair. So this is a stereo pair image, we'll talk about that in a bit too. In this view where actually you'll start giving you sense of depth. Again, if these two images are captured from approximately the same distances as the two eyes would be. This is earliest forms of stereo viewers, and I'm sure by now there are loads of additional types of viewers out there in the market. just another example of importance of stereo or early 3D photographs, I mean here is case of a stereo pair of our president Abraham Lincoln he can again as a stereo pair.

16 - Anaglyph.srt

, one interesting way of visualizing these stereo pairs has been something referred to as an Anaglyph. Where you take two images of a stereo pair, and I'm showing you a very classic image here and it encode parallax just in a single picture. What it does, it take two slightly different perspectives of the same subject and superimposes them on top of each other in contrasting colors. It produces a three-dimensional affect which in mutual corresponding set of color filters gives you a perception of depth. So for example, these two merged images here can be now viewed through a red cyan glass set like this one, and it will allow you to capture or at least perceive from this stereo pair a depth image. , you've seen enough of these types of green cyan glasses. Here is one simple one, and if you just, were able to look at the scene from this, you would actually see depth. Let's look at a little bit more of these Anaglyph types of images. Here is one that you may have seen you know, there is a lot of these types of images came up after the Mars Mission where they generated a lot of these nice Anaglyph images. Another one again a very well known classic example of Anaglyph images.

17 - How Anaglyphs Work.srt

What's going on with Anaglyphs? Well , what we have is we have patterns like this, and if you have a glass like this, what's happening is we can close the one eye, and then close the left, and you see different images pop up, right? So in essence what this happens is when you put a filter, red filter selectively passes only the red color information. And the cyan filter only lets the cyan information go through. And , then the brain attempts to do fusion. So this is what actually is interesting and lets actually play around with this.

18 - Make an Anaglyph Image.srt

So what I want you to do now, in a simple piece of code given two gray scale images where we have two images a left image and a right image, we'll provide that to you. Can you actually now just generate a simple anaglyph image. Given again a left image, a right image can you generate an image that once when seen through this A anaglyph glass set here, which has again cyan and red, would let you see that image. Again important points to remember, this is a red cyan glass set, so you have to generate an image that works with this. So here is a simple code I want you to now ex, experiment with to create an anaglyph stereo image. Again, something that could be viewed by a small set of glasses like this. We've done the precursor stuff, we loaded np and computer vision open cv2 kit. if you have this function which takes a left image and a right image and now you want to be able to create a red-cyan anaglyph using two images. To help you do this, we, , have set up this two different images. There's a left flowers and a right flowers both of them are grayscale. We can actually just do this to show them and , when you run this function it should be able to now create an anaglyph image. So here is just a simple code that you have to write taking the left image and the right image to generate an anaglyph image, which would then, , when you test run, will give you the result. Please write that code and then test it and submit it.

19 - Make an Anaglyph Image.srt

So here's the correct output of generating an anaglyph image. We use the numpy function dstack, which takes you know, different images, in this case, and stacks them together to create one combined image. The secret here is also knowing a little bit about how RGB is represented in OpenCV. So if you remember, in OpenCV images are B, G and R. The first channel is blue. The second channel is green and the third channel is red. So blue, green, red. Why do we have image right, and image right, and image left? Well, remember, image left is the one I want to show as left, because that's where I want the right information to be for the left image. And then this is blue and green, which means now as our mixture of blue and green is cyan. So if I now put the right image in blue and green, that will become cyan, this will remain red. And just stacking this up together, you should be able to see result that will be a stereopair. We run this, this is the left image. That's the right image. And that's the anaglyph image. Now, I will admit, I actually don't see 3D vision very well because that requires both of your eyes to be equally strong, and many people do have issues with both their eyes being perfectly the same strength.

20 - Making an Anaglyph.srt

It's just let's quickly review the how to make an anaglyph image. First take a greyscale stereo pair. We copy the left image to the red channel of the new image and then copy the right image to the green and blue channels of the anaglyph image. That's cyan. With, view through a red-cyan glasses, left eye only sees the left image. Right eye sees only the right image and the brain fuses to form 3D, simple. Now some of you might be curious as to why the glasses that we use in watching 3D movies are no longer. They used to be for those of you who remember from days earlier are red and cyan. Nowadays they're different. I encourage to start looking up that and ask that question. And figure out why that's the case and what's different nowadays in the cameras. Both and also the display mechanisms that actually work with different types of classes.

21 - A Simple Stereo System.srt

So now let's actually start understanding how to build a simple stereo system. Let's take our coordinate axis, here I just put a simple left camera here. I'm going to put them at coordinate axis point 0.00, which is here. Now let's imagine I can translate this viewpoint by amount Tx just in the X-axis, no change in y to another location here. So now, I have another camera, which is located just Tx in this direction farther from where the original one was at zero, zero. So both the cameras are exactly the same. They're just now translated by a certain distance. And I have , a new camera which has its own coordinate axis. So right camera is simply just looking at the same view you just translated by x along the x directions. Everything else is the same. In this case, the focal length is the same orientation is the same, everything is the same. Let's look at this scene from the top. Just looking at the X-Z plane. So here again, I have a left camera. Here now, I have actually listed the focal length. Right camera. Again, we know the focal length here. So, , the two cameras are translated by distance Tx along x-axis. Tx is also know as the baseline. We have a point, X, Y, Z. , that point appears and this viewpoint here we'll point, call this Xl for the left image, a location just an x-axis. Looking down on X, Y is pointing outwards, we're not change, looking any changes in y on this one. So then that point, appears as Xr on this viewpoint here. Question now is if I know Xl and Xr, can I compute Z? Which would be the depth of this point here, right? Now this is where actually we can bring back simple geometry to help us. So let's look at this again back into our 3D simple stereo system. I have a point in 3D, that ray of light goes from this camera, goes through this image and hits that and another one across just like we saw in the previous example. And , that's Xl and there would be an X right here for this one. Same kind of setup as before. Now we can actually compute, you link the same equations we looked at the beginning of this lecture. That Xl would be f X over Z and yl would be Y over Z. Right? That's for this point here. And in this case, , you can also do the Y. In this case , yl is the same as yr. The same point in the right camera, Xr is different and Xl is not. So here is one thing we can start doing. , we know that this camera is moved by a distance Tx. Well, that suggests something to us we can actually use. That , if I was to move this point here by that same amount, nothing else is changing, the same point would appear. So, in essence, the converse of me moving the camera by distance Tx is me moving this point in the other direction minus Tx and actually taking another picture. So now saying is rather than taking two pictures by moving the camera by distance Tx, I'm just moving the point minus Tx in the opposite direction and taking another pictures. So , that means is now I have a point T, a point here that should be visible in this one. So we don't have to look at this camera. And , now this ray is coming in this way, implying that Xr is actually moved there, same translation as minus Tx. So now, I can compute Xr. Again f, X is now just this location x here minus Tx in X direction divided by Z. So that's , becomes my Xr. Yr is the same, we haven't changed anything in the Y. So , this starts giving me a very interesting way of comparing both Xl and Xr which are now different just by the top numerator.

22 - Stereo Disparity.srt

So, let's look at that a little bit more carefully. Left camera is giving me these equations, right camera is giving me this equation. Looking into the from that image we looked at, I have Xl and Xr here. That was just showing you the image itself and, , Yr and Yl are the same. Let's now come up with a new term here. We'll call this d for disparity. And, that's the distance between point Xr and Xl. D, , is difference between Xr, Xl and Xr, the two positions in the left view and the right view, which are now projected on the same view here. So, in essence, d now because becomes f X over Z minus f X minus Tx over Z from these two equations here. And now, , by doing simple you know, math on this kind of stuff, f X over Z would actually subtract each other. This would become a plus sign and actually the disparity d would just be f Tx, that is the translation of the camera, for when we point to the other, divided by Z. , now, we can do something interesting here and now we can use this. If we know the disparity, right, if we know the disparity, we can now actually compute that. That, in essence, is the important cri, equation. This, in essence, is the important equation of computing depth from two viewpoints. , if you recall, this is baseline. This is depth. This is disparity, which we've been looking at. F is focal length. , that mean I need to know the focal length of a camera.

23 - Stereo Example.srt

Let's look at this, and now with a specific set of examples and images. Here I have given you two images, this comes from a very famous data set that has been collected by computer vision researchers for years now to do real hardcore analysis of stereo algorithms. I'm showing you two images, left and right. Left image and the right image are, you know, shown here and now using this, this depth could be computed. I haven't told you how to compute depth yet but we'll get there in a bit, specifically for images like this. But you know how to do it if I could find the same points and do those two images, right? To compute the disparity that is. Here you see a couple of interesting things. You see, , a few dark points here, this is where for example no information was available. Remember when I showed you the images of the child earlier, you just saw a lot of black image and black points in those images. Again, those were, again, but there's no information available partly because two images cannot just actually see those points clearly. Partly, the reason it can't see these kinds of things is that each and every pixel has to be visible from both cameras. This image is also referred to as a disparity image or disparity values. And here we're showing gray values form zero to 64. Zero, , no information or no disparity, 64 most. And, , if you look at it in this context it means that anything that's closer is you know, and farther is lower values, closer values are higher.

24 - Computing Disparity.srt

So, how do we compute disparity? Let's take the simple example here and we know disparity is, in essence, the difference of Xr and Xl. We need correspondence. It, means is, in both those images, I need to know exactly the locations Xr and it's the same point that should be visible in the other image. In essence, what we need is each pixel, or any pixel, has to be corresponding to the other pixel in the left image. So, any point that's in the right one should also be appearing in the left one. Now , we have learned how to do feature matching at stuff like that already. , you can imagine that any point I find in the right image, I can search for it and match for it in the left image. And, many different methods we have looked at, including even just simple correlation methods, and normalized cross-correlation methods that we've looked at, could be used to find a feature in one image and find that same feature by doing again as I said, you know, cross-correlation does is , a inner product or dot product, and it will find the most similar pixels. Computationally, you can imagine this gets to be pretty complicated. Similarly, the other method we can actually do is sum of squared difference measure errors take any pixel region and neighborhoods. Remember, the kinds of stuff we've looked at already when we've talked about just neighborhoods, and pixels and how to do kinds of filtering and comparisons. Well, that all could be applied here. But, you can imagine, this is going to get pretty complicated if I have to do pixel by pixel region by region matching and also perhaps a different scales. To help us, we can actually rely on something called the epipolar constraint. , that says is, you don't need to search the whole 2D image. When I actually found something, I want to match to from the left to the right. , what it does, is it reduces the search space to one dimensional line. Let me show you what that means, but the idea really remains is in this case, there is no change in the y-axis. So, why should I search for everything that is in this point across the whole image? I should search for it just across that line. Remember the interesting things about stereo imaging, right? Both eyes are on the same y-axis, most of the time we want to use that constraint for two cameras on, being at the same axis.

25 - Matching using Epipolar Lines.srt

So looking at the same stereo system again. So all information that we are searching for this likelihood of all the points corresponding would be on the same line. So we actually need to just look for images on the same line. This can be demonstrated by simply looking at the same example we have looked at before. find the patch on the left image. In this case here, find this polar line constraint. And take this and match it to all the regions just on that line. One by one, it reduces the computational complexity quite a bit. And now I can actually come up with matches on this one, use that to create a match score value. Again, could be somewhat square differences, or normal, normalized cross correlation. And if you notice this starts giving you a very high score, in this case high being good of a match. And this tells me this is a match to this, and this gives me correspondence. And that could be used to find the matches that could be used to do disparity computations.

26 - No Matches.srt

Let me now show you some examples of where things don't work out as well. For example, here I have one image that actually you see a lot of different black holes and stuff like that. And if you were to compare this to the ground truth, which is again what I've shown you before. You notice that there are problems and not a lot of things are happening. Partly it is, because in an image like this, sometimes even a good matching algorithm may not find matches at the patches that you're looking at. There's a lot of research going on there, stereo matching and stereo reconstruction has become really mature these days. I encourage you to look at various sites. And the many algorithms exist really do a much better job of this type of disparity computation. Another problem exists is there are occlusions. Therefore there are no matches. So for example, here you notice in this instance there is a whole lot of black spots here. Where partly because if you note this, these two regions here don't actually appear in this one. And similarly there is, you know, lots of information that's included because as I moved the viewpoints, different things are appearing. So this region is missing because, again, I just don't have this information here, in this viewpoint, so there are no matches. Similarly, when you look at this, this is caused by the fact that while this region is appearing here, it's not actually appearing cleanly in both of them.

27 - Effects of Patch Size.srt

Another part of it is the size of the patch we've used. We've looked at patches, and scale and all that kind of stuff here. If I have a smaller patch here would result in much lower, disparity calculations, much smoother disparity when you actually move to a bigger patch. And you can see that more information is available and smooth information available. In some areas but that might not be the case across the board. So here for example you see a lot more finer detail, and here you see a lot of smoothness going on. So, patches, and again refer, I refer you back to all of the lectures we've looked at on trying to deal with frequencies and stuff like that, that could be used, and scaling and using pyramids and stuff could help with this. there are lots of other methods that I'm not covering today actually you can lookup that tell you more about how to do this kind of disparity calculation from surrogate pairs.

28 - RGBD Cameras.srt

Before we end, I wanted to now actually talk a little bit about some additional types of things that are now known. For example, there are many well-known RGBD, that's the kind of a new camera, because not only does it capture RGB, but also depth information. Perhaps the most widely used these days is the Kinect camera. This actually does have a VGA camera, in addition has a infrared projector and sensor. It has lots of other types of things. What it does is actually, it projects from one of them, a pattern of light, infrared light, that this sensor picks up. And, in essence, what it does is, again, a pair, but one part is illuminating a 3D surface. The other one knows a little bit about how to use that, again, and it solves correspondence problems and everything else. How the light shape is kind of coming in. Here you kind of see this and uses that to construct a depth in it. So again, just by using a pair of this, and then, , there's an RGBD camera that also gives you the RGB information, creating an RGBD camera. So that's one of the very well-known methods, and actually a relatively cheap method if you know the cost of these types of cameras, that's been widely used to now capture depth images. There are other types of cameras also coming onto the market. after the first Kinect we now have a newer Kinect. I'm just comparing the examples of the Kinect image from Kinect 1 to Kinect 2. So the big difference between the Kinect 1, which as I've said is an RGBD camera, but it's using structured light that actually is used to reconstruct the scene. The Kinect 2 is known to use the time of flight sensor. A time of flight sensor camera is one that, computes the range or the depth of an image by measuring the time of flight of the light signal between the camera and the subject on each point of the image. More details, you can please look up yourselves, but as you can see, the two big things that actually were improvements on this one, one was resolution, which actually becomes much better in time of flight types of cameras and also more detail. , these days, additional cameras are also showing up. This is a handheld camera from Google called Project Tango. Which actually lets you extract depth sequences in real time on a handheld device like this. And, , if you paid attention these days, the Amazon Fire phone actually has five front-facing cameras. And in fact, one of the applications they have with these four of the cameras around the corners to create a way that when you move your camera, it shows parallax effects on the scene itself. Again, a lot of computer vision and face tracking goes into all of this kind of stuff. We're not going to talk much about the kinds of face tracking or stereo reconstruction anymore in this class, but I want you to everybody to think about how you can leverage depth information in computational photography.

29 - Summary.srt

So to quickly summarize, talked a lot about how we can extract shapes, structure, geometry, range and depth from a variety of methods. Specifically, talked about stereo imaging and also how do you compute depth or disparity from a stereo image pair. A lot of information exists. I just put one paper for you to look at in terms of what can be done with image based rendering types of things. That the Szeliski book has a lot of information stereo. There's a whole chapter dedicated to it. If you really get more interested in 3D computer vision, the most classic textbook in this area is by Hartley and Zisserman. While dated, it still has the most foundational methods for trying to reconstruction or geometry construction from images. Just to finish up, , there are, you know, lots of nice stereo cameras available in the market these days. Here's the one that I have. , if you notice it has two cameras and can be used to generate 3D images. And have, you know can show you this one actually. I just took a picture of the scene that I am at. I don't know if you can see this when I rotate this around. But it should show you that this has also got a display that shows the scene in 3D. Again I'm going to let you folks spend a little bit of time learning about these types of cameras and stuff like that. There are some cameras. And , there's a long history of 3D cameras, even with film are coming out. And there are these days, , game devices and a lot of cell phones that have multiple cameras that could be used to generate 3D images by just doing stereo pairs. Here is actually just using this camera, I took the images left and right pair of Daniel and they can be used in this instance to generate anaglyph image here. There are lots of other different types of ways you can display 3D images like this. For more information, please look at some of the other sites and thank you.

# 05-06 Photosynth.txt

01 - Intro.srt

Now I want to introduce to you the concept of Photosynth. Photosynth is a very unique application of computational photography that builds up on all the different steps that we have looked at so far. It brings together all those concepts to allow you to now take a series of pictures of a site and then bring them together to visualize that site in many different ways. It actually allows you to do equivalent to photo tourism, where anybody takes pictures of a site, and now you can actually visualize them and see different sites from different viewpoints.

02 - Lesson Objectives.srt

What I'm going to talk about how we can go beyond the concept of panoramas. Now, we started this lecture with panoramas being as one of the foundational things of computational photography, and we learned different steps to get the, get towards the end of building a panorama, which by now you know. But now we're going to kind of step both backwards and forwards to kind of start thinking about what can we do with just a bunch of photographs that we take in the space and use that to. We could make panoramas out of it about other types of things. And within that , I'm going to talk about the concept of taking a bunch of pictures of space and being able to kind of move around it equivalent to what is referred to as photo tourism. using that concept, I'm going to introduce a variety of things, how to put photographs on maps, and also get into well-known concepts which , you may have already used, like street views. So let's get started.

03 - Recall Panoramas.srt

Recall that we have actually talked a lot about Panoramas. We said, if we took a bunch of pictures, this is the picture that you've seen many of times, of Lesko's Sports Stadium. Each individual picture does not carry enough detail to kind of showcase the space very well. By combining all of them, we can actually, now generate a newer image. In this instance, a Panorama that actually captures the space very well because primarily, it goes beyond the limitations of a limited field of view of each and every image, and generates a larger field of view. So, in this case, 7 pictures, which are each about 3 by 2,000 pixels about 7.1 megapixel. Was used to generate a 31 megapixel image with a field of view of 151 degrees wide, and , it took a lot more disk space, and everything else, but , it shows a lot more detail. Recall also, that while we were looking at Panoramas, we learned about how to generate Planar, Spherical, and Cylindrical Panoramas. Again, took a lot of pictures, stitched them together to generate a Panorama, and by now, you know, , how we can do this kind of stuff. Now, we're going to start taking one step forward, and try to think about what else can we do with a bunch of pictures of space?

04 - Photo Tourism Photo Synth.srt

This leads me to the concept of Photo Tourism or sometimes also referred to as photosynthesis or Photo Synth. Now this a widely used method in computer vision, and computational photography, started off by Noah Snavely, Seitz, Szeliski and they referred to it as Photo Tourism and their goal was to explore photo collections in 3D in reference to each other. I will be showing you more details of this in a bit. This was a paper that was published in 2006. It's a very interesting idea, and I'm going to tell you more about it in a bit. , this was taken into a complete technology preview that's available online for you to even play around with called Photo Synth developed by Microsoft following this work. So, during this lecture, I'm going to tell you more about both these two different technologies, again something the foundational work in 2006, which has been evolved into a available package online since 2008, and , has just been recently updated. But , there are other similar solutions also available, that I'm going to introduce you to, and my assumption, and hope is that you are actually going to play around with them, because they are going to tell you more about what we can do with a collection of photographs and how to use them to showcase space?

05 - Photo Tourism.srt

To start off, let me actually go back to the source, and actually, show you the video that Noah Snavely, Steven Seitz, and Rick Szeliski did with the paper that actually, resulted in the concept of Photo Tourism. And how we can actually use this, to now showcase photographs of space, to showcase a variety of instances. And specifically in this case, they're going to talk more about how we can actually, see popular sites, where pictures have been taken, by variety of people, not just by one-self. >> Searching for a particular image of a well-photographed object using conventional tools, often results in a large number of images that are not ordered, in an intuitive way. Finding the exact picture you want, can mean browsing through page after page of thumbnails. How can we organize such large photo collections, in a more intuitive way? In this project, we present a novel system for registering large sets of photos, and exploring them in a 3D browser. Our system discovers the relative positions of the cameras used to take each photograph, situates the photos in 3D space, and provides intuitive controls for exploring the scene, and finding interesting photographs. Our system takes a collection of photos, from the same scene as input. We first find key points in each of the input images, then match keypoints between each pair of images. Next, we run an adjustment procedure to estimate the parameters of each camera, and the positions of the observed 3D points. Once the photos have been registered, they can be browsed using our photo exploration interface. Our system provides standard controls, for moving around a 3D scene. In addition, when the user selects a photograph, the virtual camera is smoothly brought into alignment, with that photo. Information about the photograph appears in the information pane, on the left. Our system provides several intuitive ways, to select new photos. One is to select an object. The user can highlight a region of the current photo, and the system automatically finds a good photo of the selection. And smoothly, moves the virtual camera to the new photo. During transitions we use a simple plane-based morph, to provide context as the camera moves. A thumbnail pane along the bottom of the screen, shows other photos of the selected object. When the user moves the mouse over a thumbnail, that photo is displayed in the main view, projected onto a planar approximation, to the selected object. Here, the user selects a thumbnail to see a different view of the statute. We also provide tools for viewing the scene, at different scales. The user can step back from the scene, with the zoom out tool. This finds photos that display a larger area of the scene. The Show Me Similar Images tool finds images of the scene with scale, and orientation similar to that of the current photo. The Zoom In button finds details, showing the user what parts of the scene, can be viewed at a higher resolution. Here, the user selects a photo of the in the upper left, and the browser zooms into the more detailed photo. Our second example uses a set of photos taken by one person, over the course of two days. We registered the photographs, and reconstructed line segments, as well as points. We can align the reconstructed model with the satellite image, to situate in a geo-referenced coordinate system. We rendered the scene using the reconstructed line segments. We also project blurred, partially transparent versions of the photos on to the scene, to convey more information with a non-photo realistic look. An overhead map is displayed in the upper right, the user can select a photo using the map. Here, the user selects a building, to see a photograph of it. For this data set, we can also move left, and right, along a row of building facades. We provide geometric controls, for this type of interaction. For each photograph, we pre-compute a left, and right neighbor based on the projected motion of the points preserved by the image. We also pre-compute a step back image, the user can quickly view more of the scene. In this example, we explore photos of the Notre Dame Cathedral in Paris, downloaded from the web. The user can select regions in the point cloud, to find images of an object. Our system also, allows you to annotate photos. These annotations are automatically transferred to new images. Here, these are labeled several regions of the current photo. As each is labelled we transfer the annotation to the other photos. The transferred annotations are highlighted in the thumbnail pane. So, as not to cover the photos, we'll hide the panes, and use the hot key to step to the next photo, in the sequence. As we move to each photograph, the annotations appear. Our system uses simple heuristics, to determine if a annotated region is occluded. As in this example, where one region is hidden. We can also transfer annotations from other sources such as, annotated images on Flickr. In this scene, we've also added several other annotations by hand. Our annotation transfer algorithm is sensitive to scale. If we look at photographs taken at different scales, we see different annotations. Next, we explore a set of photos of Half Dome in Yosemite National Park, gathered from the web. If the user finds a view point they like, our system makes it easy to find images taken from a similar view point. By selecting the Lock The Camera option, we can generate a slideshow, where an object remains fixed in the view. Now, we unlock the camera. We can also register historical imagery such as, this photograph of Half Dome taken by Ansel Adams in 1960. Here's our estimate of where Ansel was standing, when he took this photograph. [SOUND] Here, we compare the photograph to a synthetic rendering, from the same location. The port has been manually, added for clarity. Our final example is a scene created, from about 80 photos of a walk along the Great Wall of China. We organized about 20 of the photos seen here, into a slide show. We have experimented with an alternative morphing technique, that creates a mesh from the 3D point cloud. Which is used as an imposter, for the true scene geometry. This methods often works well for nearby view points, but creates artifacts in cases where the matching fails. We hope you have enjoyed our 3D photo tours. >> So, for more detail, visit the website that I recommend you folks to look at. And actually, there is additional data on this website. I wanted to show the entire video because there's a lot of additional stuff, in this work, that is actually exciting. it covers a lot of things, we've already kind of discussed. I'm going to give you more details, on some of them. It builds on the concept of feature detection, that we've looked at before, and are using that extract parameters of cameras. And then, , you heard the mention a little bit about port morphing. Where they actually use the point cloud, as the proxy to help you do morphing, between different types of images. So now, let's dive in deeper.

06 - Photo Tourism Overview.srt

So the whole photo tourism pipeline that we actually looked at starts with a bunch of input photographs. And actually, what's exciting in this one is you can actually get those pictures from, you know, public sources where people have taken pictures. And this works very well for well known historic sites and tourism sites. Using those pictures we can actually need to do a little bit of scene reconstruction. That is how we can actually model the environment. And this actually gets into a whole lot of stuff called structure for motion and we'll talk more about that in a bit. So the goal here is doing scene reconstruction to get the 3D cloud points that are best captured from the scene and also locating the camera positions for each and every one viewpoint. So, what we really need to figure out is the relative camera positions and the orientations of each camera, generate a point cloud and also correspondences between all of them. So, so far, we have looked at information about how we can actually understand and model relative camera positions when we do panoramas and also sparse correspondences. But we never went towards the whole world of trying to create a 3D geometry of the scene in any of the work that we so far have looked at. And , a beautiful part of their work was this interactive explorer mechanism, if we doing track with the photographs, annotate and all of that kind of stuff. I'm not going to talk much about that, we're going to showcase some examples of this for sure.

07 - Scene Reconstruction.srt

So let's talk a little bit about scene reconstruction. The whole purpose of scene reconstruction is to automatically estimate the position, orientation and focal length of cameras. We also then want to be able to model and extract 3D positions of feature points. I'm not going to get into a lot of details about how we can get to focal length of cameras in fact, Noah Snavely, on his website, has a very nice tool to bundle our. That actually let's you do this kind of stuff with variety of different types of methods and pictures that you upload to it. I will actually make available to you some other resources to look at how to do this kind of camera calibration. But we, kind of, skipping that for now, because the kind of stuff we need to do would be , we'll be able to move forward without getting into those types of details. So what we really want to do to be able to automatically estimate is first we want to do feature detection, something we have talked about before. Then we want to do a little bit of matching of features, something again we have looked at before, and we will talk about again now. Using this we want to be able to find correspondences. Remember we talked about correspondences a lot when we talked about stereo. We needed to find two points in an image that allowed us to then actually compute a disparity map, which would be then used to create a depth map or a depth image that we can actually use the extract depth arrange of an image. Same concept, now we want to do this not just with two images but a lot more. we want to run to something called structure from motion, I'll talk a little about that next, in a few minutes.

08 - Feature Detection.srt

So let's actually start talking about this whole concept. We start with feature detection, again we've looked at how we can do feature detection using concepts like sift. Something is also that you're playing around with, with the development environment you're experimenting with right now. Here means is I take a bunch of images. And again we have all of these images here. And now we're interested in , is running a feature detector on all of these images. So here you kind of see in a pictorial manner that each and every image now has a bunch of different features. Now recall again what we learned about feature detection, that might have similarities that would actually be dealing with both illumination changes, scale changes and rotation changes. I'm here just showcasing some simple examples. These are not the actual features. you can do this in your code yourself. You can take a bunch of pictures and start dong feature detection on it. Let me show you an example of what this feature detection look like for real images. Here's an image. And we run a feature detector. it identifies veracity of different types of things at different scales. And this would allow me to start kind of now finding interesting features that we can do matching with. Recall we did this kind of stuff already when we looked at panoramas.

09 - Pairwise Feature Matching.srt

So, after we have done feature detection, let's talk about matching images that have similar features. Again, something we have done before when we talked about panoramas. Remember, most of the pipeline of this is similar to what we did for panoramas, except in this time around, we're not interested in putting those images together in creating a seamless larger image. We just want to be able to model, which images have similar content, so we can say that they're related to each other. So for example, I look at these features. I say, okay, this image and this two images have something in feature. We create a connection between them and we do this for each image with relationship to each other. And , create a graph structure, which says, this image is common with this, this image is common with this. So, in essence, what we do is take one image, find the features and match it with another image and do this across for the whole database. Yes, it takes computational time, but this is something valuable for us to do. one of the contributions of the original Photosynth paper was a piece of software, the Bundler, which is available from their site that does it quite efficiently. Now you may remember one of the ways of doing this kind of matching that we have talked about before and that was using RANSAC random sampling and consensus. Again, allows us to match different types of images together based on features that are within it. And uses that to find images that are well, similar and closer to each other. This is what we use for doing again, matching and then using that to generate a pattern same technique. But in this instance, what we're really interested in is finding out, which images are related to each other in a parallelized fashion. something we know we can do quite well if we have all of these tools. This is a similar process to what, , we talked about when we talked about using recognizing panoramas, which said is, if I took a bunch of pictures, that were not in order. If I threw them into a process, can they figure out which pictures will actually use, be used to create a panorama and which won't. Same process applied here.

10 - Correspondence Estimation.srt

Now let's talk about correspondence. Again, similar idea to what we talked about when we talked about depth imaging. here I'm showing four different images of the Trevi Fountain taken completely under different lighting conditions and different scales. If I run feature detection on it, it would find four features perhaps that are actually on the head of the statue here. Course what I'm interested in making sure is that these features are the same. And , that is what my correspondence is going to be. After matching, within, you know, thresholds and stuff, it says this feature and this feature are the same. Similarly, it will do the same for the next feature, and , from there to the last one. So in essence, now this says is all features are the same. And this will allow us to link up pairwise matches to form a connected graph of matches across several images. This image is connected to this. This image is connected to this. This image is connected to this. in the UI that was shown in the video, you saw how that was used to be able to kind of scale in and look at different images at different locations. I'll show you examples of that also.

11 - Structure From Motion.srt

The final step is when we now want to run something referred to a structure from motion. Some of you have taken a computer vision class are going to learn more about this. I'm going to just brush over it and, if needed, can provide more details to anybody who is interested. Let's start off by saying here's an image. I'm using a simple box here, but actually let's think about this image here as the image that we're trying to match. , we have one camera. In this case, we're going to talk about more than one camera. Remember, in stereo, we talked about two cameras, but let's talk about three in this instance, shown by these three simple cones. Again, in this larger image, this is the one image that we're actually trying to pay attention to. , there might be more images, because this is a real scene, right? This is a real scene with lots of 3D points. One image is captured. We want to figure out where this image is in all of these three different cameras. Pretty much like what we did for panoramas. Again, when we looked at a bundle of rays, and asked the question if the bundle of rays could be used. We can now actually synthesize a new image. In this case, we're not trying to synthesize a new image. We're trying to use the fact, that can we figure out the registration and alignment of this one image in all these three different viewpoints? , there are lots of caudal points visible to each and every one of the cameras which relate back to all of the points that are in the scene itself. Just take point p1 here and we're going to project this into all three of the different view points. Again, each and every one camera has a transformation matrix. For simplicity's sake, I am referring them to having a rotation, because that's what's actually true here, and a translation. So, for camera one R1,t1, camera two R2,t2, and camera 3 R3,t3. So using this, , these are the transformations we need to optimize over. Each camera is rotated differently and, , translated. So, , this will allow us to hopefully figure out what the transformation matrix for each and every one of the cameras are. But , it's coming as each point is, , being seen in each and every one of the three different cameras, as shown by this ray of light. So again, like what we've done before when we talked about panoramas, here we want to be able to take this function and minimize it to have, this could be done with a variety of different points that we have. To compute a minimum function which actually optimizes our rotation transformation for each and every point. Recall again how we did this when we talked about the whole concept of panoramas.

12 - Incremental Structure From Motion.srt

So, one of the other ideas that was very interesting and novel in this paper on photo tourism was let's try to do this incrementally. There's a lot of work on trying to do this model adjustment and structure for motion, doing it for the entire graph. That is figuring out all the matches and then trying to connect all of them and rather completely for each and every image. One of the interesting things that they proposed was lets try to do this incrementally. Let's not try to solve the entire problem. Let's do it as a new image comes in. So which means is let's just take two images and use that to build a kind of correspondence and matches and as another one comes in, let's keep on adding them in. Say, for example here you see two cameras looking at a scene, the claro point is a scene here. And , how can we use this incrementally to not compute the structure for the scene, but at the same time if more cameras come in, how can we model more? So here , just by two, you get a little bit of sense of the scene. But next, what we'll do is start adding cameras. More cameras, more points are visible. Incrementally, you add to it. And now, actually you can see a much detailed plot of points generating for the scene and that's what we want. Now we know the orientation of each and every camera. We also know a cloud of points. I'll show you examples of this in a bit, that'll clarify this. So , now you see a cloud of points and using the mouse, we can browse and click on one picture that shows the registered image through the whole 3D cloud of points. And then , we can move around and find other images as we have seen before in the other example. You can also then, , click into different parts of am image. Zoom in and see other image that may be best registered things. See more details and browse around. Quiet an interesting and intuitive way, looking around the scene like this. So these are researchers who developed this whole photo tourism system were able to then work with the whole concept was actually then taken over and generated into a technology preview referred to as photosynth.net.

13 - Make a Photosynth.srt

So here's what I want all of you to do. First, I want you to try to make a Photosynth for yourself. The best way to do is go to the Photosynth site and actually create a account for yourself and then upload some pictures and then use that to kind of showcase a variety of things. And , if, just, this is one of those things, completion is just entering your a Photosynth ID in this box here to showcase that you've done this. Again, make sure that there's no private stuff on this thing. You don't have to make all of this public. Just, this is just for us to know that you've actually accomplished this goal. You will actually be doing a complete assignment on Photosynth as one of the assignments in the class. So here, we are at the site. You can sign in by creating an account for yourself. I'm going to sign in as myself very quickly. So when you are on this site, you can kind of walk around and see a variety of things. There's a whole lot of detail about. It will tell you how to also do different types of synths and explore different types of things. , these are a variety of things other people have been doing. In addition to that, , you can also go and create your own synths and there are a variety of ways of doing this kind of stuff. I recommend that you play around with that technology. They just announced a newer version of this, which also lets you do this kind of synth in 3D and the best way to create it is going in here and uploading a bunch of pictures. Again, there is a lot of help on how you can actually learn how to do this. This new version allows you to do four types of different synths spinning camera, walking Circulating around to create a panorama also walking this way. Again, there's a lot of details. And , there are also apps available on an Android, iOS and windows platform for you allowing to do this kind of stuff. So please create an account, try to upload an image or two, but you'll need, , more than two to create any kind of synth. Load a variety of images. For those of you who have what, Windows Workstations, you can actually do much more with Photosynth. And , you can just use the browser for what I do these kinds of things to. So please explore and return to this site when you're done and enter your Photosynth ID.

14 - Photosynth Example.srt

Let me now show you some examples of what we can do with this. So, here is an example of something I did myself in the earlier version of Photosynth. I was in Barcelona, and , I took a bunch of pictures at the Sagrada Familia. So, these are all the pictures I took, and I'm going to show you how we can actually use this to generate a Panorama, or a Photosynth, not a Panorama. So, these are some of the images that I'm just showcasing. I just did a captured of screen from my laptop. When I was doing this, and I'm just showcasing the video to you. Variety of interfaces are available here you can now see the same kinds of things you saw on the research version that became a complete technology preview, that Microsoft invested quite a bit in. Several of my students have worked on that team that generated some of these technologies, and now you can see, just by clicking on different images. You can browse, you can see all of different types of details of this amazing structure. If you haven't seen it yourself, I mean, I'd recommend, if you're in Barcelona to want to go see this church, but again, just by photos, you can start browsing this, i.e. the term photo tourism. You can also see things like overhead views. Very hard to see it, but now as you browse through, you can actually see pictures that are actually used to generate the cloud of points in 3D. And this is the cloud of points, hard to see on this screen most probably for you all, but if you notice, when I click on different types of things, it shows the bounding box of the original image. Then you can then zoom into, or actually get much more details for. Again, purely based on pictures taken by me walking around the Sagrada Familia, and now we have this whole wonderful visualization of a site. , the newer version allows you to do different types of things, and continuing in my theme of taking pictures of Sports Stadiums, this was me at a game of the Georgia Tech Yellow Jackets. Just took a bunch of different pictures there, and here, now you see a visualization of me just now taking, and generating a panorama. Except that if you notice, there are exact different pictures here, and you can see people moving around, and I can interactively control it. So, if you were able to do this, you should have been actually been able to see simpler things like this on your own screen from your own pictures. We can also, , see these different types of additional scenes, here's a beautiful example of another just walking scene. Bunch of different pictures, not perfect registration across the board, but a very nice visual, , in this instance, the person who generated this also showed details. Here I'm just scrolling through to kind of move through the videos faster.

15 - Google Maps Example.srt

, I don't want to just show a bias towards Microsoft here. They've done some amazing stuff in this area. But , Google has also played into this area recently with all of their Google Map stuff. So here, for example, I'm showing an example from Google Maps, showing the Coliseum in Rome. They have actually captured, but more importantly, citizens have captured and put up and registered on the scene. So, again, just looking at the coliseum, there are lots of pictures available using sites like the outside Panoramic and we can look at different pictures. And when I can click on it, I can actually zoom in. And , now I can browse the scene in a lot of detail. And , if you notice with this interface as I move around, it either finds a picture and a best viewpoint for it or lets me browse around in the space itself. just browsing shows you different images and you can kind of again, do a little bit of touring around and see interesting sights. Now, I want to make sure that people understand while in the old days, this was important for well known sites with a lot of pictures. But with the growth of cameras, something we've talked about before in this class and the availability of, you know systems like Google Maps, where they're actually doing allot of capture themselves. We can actually do this for sites like For example, the Georgia Tech campus. So here now, let me show you a video. Say, you want to come to Georgia Tech campus and some of you, , have not been able to. You can now get a tour of the Georgia Tech campus. So here , I'm going to first show the Klaus Advanced Computing building. If you notice there is a blue mark where I was, that's my office. And , I clicked on one of the panoramas. And now we can see the Clough Undergraduate Learning Center, the Van Leer building. Zooming back out on the map. Now, I'm going to show a little bit more detail and actually get into something. Again, you may have used called street views. The Clough's building, I'm at this dot here in my office. Loads of different types of available images. Here we're going to get on to the first drive. I'm just going to put myself on that street. There. And we're going to go for a little walk. Here you see these are , images captured by street views. We'll talk about that in a bit. But now, I can actually start moving around. Again, if you notice there are lots of pictures here. Now these pictures have been taken by a street view camera system. I'll show you that in a bit. I'm going out on Ferst Drive, the biomedical buildings are on this side. Once I get to Ferst, what I want to do is I want to have in down Ferst a little bit more, but I want to actually come back on Atlantic Drive. And on Atlantic Drive, we can move up. This is again Bandolier from the other side. Bandolier is , where the electrical engineering and, electrical and computer engineering department is. And this, is the College of Computing building. And this, is where my office is.

16 - Google Streetview.srt

Now Street Views is captured with a camera that's mounted on a car that goes all over the country or the world these days. , there have been other types of examples of this type of capturing system on a tricycle here. And there have been sightings also of much more mobile capture devices, like this. Again, this is an example of more authoritative, you know, a company like Google is walking around to create these amazing maps. So, I just caught a, captured a segment from the about Street View site, which shows at least a simple pipeline of about how Street Views work. Starts, starts with collecting imagery. In this case, it's similar to the kind of imagery we have looked at in a panorama. , use that align all of the imagery together. And then , using that to generate a full panorama, like the one in attaching it to the maps on the street. So this is in essence a simplistic pipeline of how Street Views works. , there's a whole lot of machinery in there that I cannot tell you anything about, because most of it might also be proprietary. But in essence, what you see is they actually can use images from the special panoramic cameras, registered, aligned. And then , create images like this. , we also want to do this kind of stuff with our own images. So here is something, this morning, I, before I was walking into my office and I just used my iPhone to take a bunch of different images. And , they are interesting in their own right, but I just uploaded them to Photosynth and here is the output. remember, these are pictures I took. Now just the walk through here into my office. Now, if you, if I had office hours, this is where you would come to see me. My office here again. If you're really interested, you can zoom in here and see the friends that we've seen in a variety of different lectures already. Sitting there in my office, waiting for me. Now, again, this is giving you a sense of its 3Dness of the environment is far from perfect. Alignment is not there. But remember, this is just like 20 pictures that you use to generate this walk-through. Quite impressive, you can manipulate it and visualize again, a photo tour.

17 - Summary.srt

So today I showed you a variety of fun and interesting things. You will also have, exposure to them in an assignment that we are going to have on exactly the topic. So, taking a bunch of pictures and generating panoramas and other types of artifacts like Photosynth. The whole purpose of this is, , ho we can use photos to show space or environments. We can imagine this has value in real estate and in architecture, and all that kind of stuff, beyond tourism. We talked a little bit about the whole concept of photo tourism. That is, somebody took a bunch of pictures. How can we actually tour that site without actually going there? , it's always fun to go there. Photosynth and also something we looked at in detail. And I just kind of touched on the whole concept of street views. Again, it's building on a whole lot of concepts that we have looked at already. Lot of detail, again, refer back to the original paper by Noah Snavely, Steve Seitz, Richard Szeliski. There's a newer paper by the Google folks on the photo tours, which is running on Google right now. There are many other similar examples that are out there. I encourage you to look for them and post them in Piazza. And , the Szeliski book also has more details on this. One thing I also recommend when you are actually on the, photo tour site is you will actually see examples of additional work. For example, building Rome in a day, where a bunch of pictures that are taken of the city of Rome, how they can be used to actually model the whole city. That's the kind of stuff that both is being done for more targeted, you know move your own car with a bunch of different cameras in the city, or have more citizen-generated imagery to generate, models of cities. There is another paper also referred to as Building Rome On a Cloudless Day, which does, attempt to do the same kinds of reconstruction of cities like Rome, but with less computation. That is, lack of a cloud, computer cloud, that is. Again, loads of stuff on this one. We'll discuss this on Piazza as much as we want. But hopefully, I hope you have enjoyed all the concepts of your coverage so far.

# 05-07 Extrinsic Camera Parameters (opt).txt

01 - Intro to Camera Calibration.srt

Hello and welcome back. In module five, we have covered a lot of interesting topics, starting off with image transformations, where we learned about how to warp and transform images, applied them for image morphing, then we learned about image panoramas and HDR, and a few other things, like stereo. Now I want to introduce to you a series of lectures recorded by my very good friend Aaron Bobick, for the Computer Vision class on camera calibration. When we talked about HDR and panoramas, we talked about the fact that we wanted to be able to take images and actually take the pixels from those images and model them to the environment. And , that meant the geometry of the scene, and also the radiometric, the color information in the scene. Now, what we want to do here is talk about calibration in a general framework. In those lectures, I talked about relative information. That is, to take a series of images, and the pixels from that could be used to model relative information in a scene. Now we want to be able to do this in a much more general framework. What I'm referring to here is a series of three lectures by Aaron Bobick. Aaron and I have known each other for a very long time, he's been teaching computer division here. He is much more funnier than I am and his interesting lectures will actually keep you engaged. This is optional material and I just want you to watch them, and it'll give you guidance of what happens when you want to actually start getting into things like the photosynth application that we've looked at that actually gets 3D points from scenes.

02 - Preface Extrinsic Camera Parameters.srt

So in the first of the series of three lectures by Aaron, he talks about extrinsic camera calibration. In this one, , the goal is if you take a series of pictures of an environment, with a camera like this, we want to be able to figure out, by those pictures, what is actually the word model with respect to where the pictures are. That is more of understanding where this camera is in the real world, with respect to anything else in the environment. Again, remember when we talked about photosynth we talked about the whole concept of figuring out pictures as they relate to each other. What this process of camera calibration, especially the extremes of calibration, is aimed at locating the camera in the 3-D world and giving the position and all the degrees of freedom of it. You'll be building on all the concepts of homogenous coordinates and the transformations that we have looked at, and actually will use some of the similar notation. Be warned, some of the notation that Aaron uses is slightly different. Pay attention to the concepts in these lectures and that's where the foundation of this approach comes in.

03 - Intro.srt

Welcome back to Computer Vision. I hope it's welcome back because if you're jumping in now you missed some really good jokes and some of the better lectures. the, this one's okay, actually. Anyway, today we're going to talk about extrinsic camera calibration. We'll define what that means in a minute. We took a little hiatus to talk about stereo so you guys could get working on your stereo, and stereo was our first look at multiview geometry, multiview cameras. And we talked about how, in order to do real depth reconstruction, we have to understand the geometry of what's gong on between the cameras, and that's what we're going to start talking about today. So, before the stereo thing, the stereo sections, we introduced a projection, perspective projection. And here is the model that we used. In particular, we had a system where we went, where we had a center of projection that was located at the origin of a three-dimensional camera system. And then we derived from similar triangles the location on the image of the point projected down onto the image plane. And then in order to figure out where the point was going to land on the image, we just eliminated that last coordinate. Now we said that this was a bit of an issue because this division by Z was non-linear. And because we had to pull out the particular Z, it wasn't a constant Z, it was the particular Z. So we introduced this notion of homogeneous coordinates. And the homogeneous coordinates essentially added an, another component to the vector. And if it was 2D, it became a three long vector, 3D became four. And the idea was, that we were going to be able to convert from homogenous to non-homogenous when we needed it. But before we did that, all of our operations could be done through matrix multiplication. Which, by the way made homogenous coordinates, the whole thing, invariant under scale. I could scale a coordinate, homogenous coordinate by anything and when I did the, the normalization, divide by w here. It would go away. And, one of the reasons we did this is we said that perspective projection could now be done as a matrix multiplication. So, here I've written, one, one, and here, we've got 1 over f. And by the way, just to make life easier, I'm using the absolute value of z, so we don't have to worry about z being positive or negative. So, when I do the multiplication, I get this homogeneous coordinate. And when I want to normalize and go to unhomogeneous, I get the u v by dividing it out. But in all of this discussion about projection, we have the notion of a camera's coordinate system. By the way, I'm going to go like this. And it's not some like, weird curse in Georgia. It's, it's a coordinate system, one, two, three. Okay, so we have an origin and a coordinate system. And we said that we put the center of projection at the camera's coordinate system. And we have the z axis, the optic axis going down the z axis. So to do geometric reasoning about the world, we need to know, we need to be able to relate the coordinate system of the, I guess I'm going to have to do this. We have to relate the coordinate system of the world to the coordinate system of the camera. And, in fact, today, what we'll do is the coordinate system from the world to the camera, and then next, I don't know, today, I don't know when you're going to watch it, next month. The next lesson will be the coordinate system from the camera 3D coordinate system, to the image.

04 - Geometric Camera Calibration.srt

This whole thing falls under the labeling of geometric camera Calibration. In order to be able, for the camera to tell us about things in the world, we need to know the geometric relationship between the camera and the world. For reference, you can take a look at the Forsyth and Ponce book. The sections are, are listed here, and there's a nice description. So as we said, geometric camera calibration's composed of two parts. There's the first part that goes from some arbitrary world coordinate system. You know, put your origin wherever you want it to be to wherever the camera is, and that tells you where the camera is in the world and its pose. And then the second one is from the 3D camera to the image plane. The first one is called the Extrinsic parameters. That's the Extrinsic it goes from the world coordinate system to the, to the camera coordinate system. The Intrinsic, which we'll do next time, is from the 3D camera system to the image. So let's talk about camera pose or the orientation and location of the camera frame with respect to the world. In this diagram, this transform T is a transform that goes between the world and the camera system. Okay? And that's what this T with lower, lower w, upper C is going to mean. All right. We're going to talk more about this notation in a minute. The transformation that we're going to talk about is this going from world coordinates to camera coordinates.

05 - Degrees of Freedom Quiz.srt

So that's a good time for a quiz. How many degrees of freedom are there in specifying the extrinsic parameters? In specifying the relationship between the world coordinate system and the camera coordinate system? A) 5, b) 6, c) 3, and it's three dimensional space, or d) 9.

06 - Degrees of Freedom Solution.srt

Okay, well, let's take a look. This slide says for rigid body transformations we need a way to specify the six degrees of freedom of a rigid body. So the answer was six. But why six?

07 - Rigid Body Transformations.srt

Well, you can e, easily think about it this way. Let's define a rigid body as just a collection of points whose relative positions to each other can't change. And, for the mathematicians in the audience, we're going to pretend that's a well-defined statement, okay? Or, you can think of it as a box. So, the first thing I can do is I can located that box in 3D. I can take one point of that box, say the corner here, and figure out the x, y and z location of that box. So, that's three degrees of freedom. Then, I can take some other point on that box, let's say the corner, and I can move it around. Now, I can't change it's location space arbitrarily because I'm holding this point fixed. So, essentially, this corner can move around on the sphere. So, this point here can be moved around anywhere on the sphere. So rot, this is supposed to be like rotated backwards this way, all right? And just the way you can think of on a globe. >From the middle, there's a latitude and longitude to get any location. There's two degrees of freedom of a vector's direction. So, that's another two degrees of freedom. So, we're up to five. And finally, once I have this vector specified, I can rotate, I can spin about that vector. So, the cube here, as indicated by this fancy Microsoft PowerPoint arrow that's in here, we can rotate it about that diagonal. So, that's one more degree of freedom. So, that's why there are six degrees of freedom for a rigid body. I'm going to assume most of you knew that already, but there you go. That's why there's a fast forward button.

08 - Notation.srt

So now life gets ugly. I'm going to be using the notation from Forsyth and Ponce. It, it might not be the best notation, but it is a notation for dealing with quarter transformations, which is what we're going to be doing. So the idea here is that superscripts are going to represent what coordinate frame you're in. So here I have some point P and I've got the A coordinate frame. And the expression of the location of point P in the A coordinate frame can be thought of as a variety of ways. You can think of it as the location x, y, z in the A frame. But if you remember a little but from your, I don't know, algebra, calculus. The right way of thinking about the vector that goes from the origin to P, that's this vector here is, it's got the i component of the amount x. The j component of the amount y, A and this k component of the amount z, A. So a vector is actually the sum of these three components, the i component, j component, k component. Each scaled by the coefficients, x and A, y and A, z and A. Suppose I want to express the location of point P, whose value I might know in coordinate frame A, but I'd like to know where it is in terms of coordinate frame B. Well that's just a translation and it's handled very simply by saying, the location of P in B is just the location of P in A plus the location of the origin A expressed in the B frame, all right? And so that equation just gives us that new offset and this OA in B, that's just a three vector. That's the offset of the origin of A in the B frame. I told you this was ugly, but, we, you know, we have to slog through it. The good news is once again, homogeneous transformations or I should say, homogeneous coordinates are going to come to our rescue. Where translation can be expressed as a multiplication. So we've rewritten this top equation P in B is equal to p in a plus OA in B, as this matrix transformation. A couple of things, first of all, that i, that's a three by three identity matrix. So this is, and since OA and B is a, is a three by one. This is a four by four matrix, which means that this vector down here, that's actually a zero vector of length three transpose, so it's three zeroes in a row. It's actually 0, 0, 0 and this is 1, 1, 1, 0, 0, 0, 0. That's clear, isn't it? That's why we draw it this way. Okay? And just to remind you, translation is commutative. Okay? And you can actually show that in the matrix multiplication, if you wanted.

09 - Rotation.srt

Now life gets uglier, rotation. What I have here is a figure, or I think this is also from four-side composite slides, and what I'm showing you is two coordinate frames, A and B. And you'll notice that A has an I vector, a J vector and a K vector. And B has an I vector, a J vector, and a K vector. And one of the important things to realize is that this P value, the vector from the origin, is, can be expressed in two ways. It can be expressed as some components in the a frame times the x, y, z components, or some components in the B frame with the components in the x, y, and z frame, right? They're the same vector, right? And what's key is understanding that there are these basis vectors and we need to know the amount of component that multiples each of them. What we want to be able to do is say we're going to rotate the frame from A to B, and that's what this says. What this says is given my point described in A, I'm going to have a rotation operator that would give me the P now expressed in the B frame. And R A to B means describing frame A in the coordinate system of B and it says, so if you gave me the location of a point in terms of the components of A, this is and it's only a rotation after applying R, I get the components in the frame B.

10 - What Does R Look Like.srt

So what does R look like? Well there are two ways to think about this, first we'll think about it the hard way. R A to B expresses how each basis vector in A would be expressed in terms of B. So the first column of R A B is the component of the I vector of A expressed in terms of how much it has in the i direction in B, in the j direction in B, and in the k direction in B. So you can think of it as like the dot product between i A, and each of the components of B, i B, j B, k B. And likewise each of the following columns is done, is, is that way. So, one way of thinking about this is that the columns of R A B are the i vector of A expressed in the B coordinate frame. Then the j vector of A in the B coordinate frame, and the k vector of A in the B coordinate frame. All right, so why is this true? Well let's think about it this way. Suppose I had a point, vector, in the A frame. That was just at value of 1, 0, 0, okay? So what that means it's actually a distance of 1 in the i direction of the A frame and none in the j and k of the A frame. What should the value of that be? All right? And I'll just write that down here. Well, this multiplies this. This to that so. So it would just give me the iA dotted with iB. So the first component of the transformed frame is just the i vectors amount in, in the i direction of B. Well, now lets go through this. We go one, two, three. It's again, is going to get iA.jB. Okay, and again iA.kB. In other words it's just what it says here is what we have to get out if we had just a 1, 0, 0, needs to be how that vector is dotted which each of the components. And that's why this matrix can be thought of as having it's column vectors as just being the each of the basis vectors of the a frame expressed in terms of the b frame. Do you get that? So on the sides you'll have that. So just press the pause button and that way you can. See what's going on. Just to remind you I labeled this that the columns of the rotation matrix are the axes of frame A expressed in frame B. Why? because we just went through all that nonsense showing it to you. By the way, it can also be thought of as the rows are just the column, are just the bases of the B vector expressed in the A frame, right? So here's iB in, in the, in terms of the i component of A. Here's iB in terms of the j. Here's iB in k. So you think about this is that if I were to transpose that vector. So I made the columns the rows, the rows the columns, I would now have instead of RAB, I'd have RBA, all right? This is an orthogonal matrix, right? The orthogonal matrix, all of the rows, all the columns are unit vectors that are perpendicular to each other. So the determinate so the, the magnets of determinate is 1. And it's your traditional rotation matrix and by the way, really important is that the inverse is equal to the transpose. So, if you had a rotation matrix and you want to go back and forth between the two, the inverse and transpose, which realize it has to be. Because the inverse of RAB has to be RBA and we just showed how the, the rows are the sort of the transposed of the spec to the columns.

11 - Rotation About Z Axis Example.srt

So let's take a very simple example. So here I have two frames and I'm telling you that the rotation of A to B is just about the z-axis. So the image on the right, I'm looking down on the z-axis. Now this should look very familiar to you when I ask you, what is the rotation matrix? Why? Because you did this in algebra, right? You talked about just rotation of an angle theta, about the origin when you were doing x, y. And hopefully, you remember something that looked like this. See it says, cosine theta minus sine theta, sine theta, cosine theta. Right? George Thomas who wrote the Calculus textbook that many of us use, taught it to me as Charlies little sister, sock Charlie. And that way you remember where the minus sign goes. anyway, so the point is that this matrix here is just for rotating the x and y, keeping the z constant. So if I wanted to get an arbitrary orientation, what I could do is a series of rotations to get things where I want them. Turns out there are many standards about how to do that. One that most of, many of us know in math and computer vision are Euler angles. Euler angles say, you rotate about Z. If, if this was Z, you would rotate, let's pretend Z is up, rotate about Z. Rotate about the new X, and then you rotate about the new again, Z. All right? For those of you who fly airplanes, I think it's heading, pitch and roll? Maybe it's boats, I don't know. Heading, you orient yourself. So, you know northwest. You pitch, that's up and down this way, and then you roll. That's rotation about this way. All right. So you're about the world Z, the new X, the new Y. There's roll, pitch and yaw. There's azimuth, elevation and I guess roll for those of you who are used to launching mortars, azimuth, el, anyway. , there are these three basic matrices, rotation about the X, Y and Z. The order matters, okay? We're not going to worry too much, in fact not at all about getting that order. But what it is, is here are the three rotation matrices written as a function of the, their angles. There's the rotation about x, rotation about z, rotation about y. I put them in that order. Why? I have no idea they used to be a different order, but, but it doesn't matter. The idea is that you can rotate about each of these different axes. Now, whether you pre-multiply or post-multiply, that's an issue. So do we do the x1, then the y1 and the z1? Or the z1, the y1 and the x1? And that depends upon whether you're rotating in the new frame or the old frame. Then, is theta positive or negative? So you have to worry about these things really well when you do this. And this is why we build spacecraft in simulation before we build them for real? Because when it doesn't work in simulation, the engineer goes, says, I don't know, try negative 20. Because knowing which way your angles go is, is, is a very difficult thing.

12 - Rotation in Homogeneous Coordinates.srt

How about, just an easier way. Once again, to the rescue, is going to come homogeneous coordinates. And, we're just going to assume that we have a rotation matrix, okay. So here, I took that top equation, that the p b is the rotated version of p a, and now, instead of the identity matrix in the top left, and the offset in the right, we have the rotation matrix here, and that's a three by three. So this is zero vector is a three by one of zeroes, so transposed is just zero, zero, zero, this is zero, zero, zero this way. Okay. And that makes, rotation a matrix multiplication. we, we're using homogeneous coordinates. And to remind you, unlike translation, rotation is not commutative.

13 - Rigid Transformation.srt

So now, we could do the total rigid transformations. So a total rigid transformation, if I have some point in the A system, I first have to rotate to get aligned in the B system, and then I have to offset it by whatever the offset of the A system in the B system is, that's what this equation says. Using homogeneous transformation, or homogeneous coordinates, we can do this all in one step. So here, we have a rigid transformation and it's really nice, right? We have our point here. We've rotated it, and then we've translated it. And what that says is we have this single matrix. Right? This part here is a three by three. This is a three by one. This is a one by three of 0s. This is just a 1. So our total transformation matrix is a four by four. And it does the, both the rotation and transfer, and, and translation. Cool, right? It gets better. Thank God. So here I've written what we had before. We have P in the A frame, expressed in homogeneous coordinates. Here is our four by four transformation matrix. Here is B expressed in the B frame. And I'm just going to write this as transformation from A to B. But suppose I wanted to go from B to A? Well, that would be written as transformation from B to A, and I'd have the point P in the, in the A frame. But the way to get that transformation is to just invert the A to B to get me the B to A. And then, this transformation takes the, the value from the B frame back to the A frame. And the idea is that our transformation matrices are, homogeneous transformation four by fours, are typically invertible. And so, once we have one that goes from, say, a camera to world frame, we can go from a world to camera fame or, or, or the other way around. And this invertibility of homogeneous transformations is very powerful and used all the time.

14 - Translation and Rotation.srt

So to review, Translation and rotation. >From frame A to B. To express this in the non-homogeneous or regular coordinates. We take the location of some point p in the A frame, we rotate it and then we translate it. In, in Homogeneous coordinates, we write it as this single matrix. Where the matrix has the rotation matrix in it in the top left and the translation vector located here in the right-hand column. And the key is that homogenous coordinates allow us to write this coordinate transforms as a single matrix, but I said that four times already, so you're saying like get on with it already. So, now finally, we can talk about going from World to Camera frame. Here's our equation, using sort of non-homogeneous regular coordinates. Where the idea is, if we have some point p in the world, so it's a point location in the world frame, we have to rotate it, orient it with, to know which way it would be oriented in the camera frame, and then we have the translation from the World to Camera frame, okay? So, we have this sort of ugly equation that would get us from a point in the world to a point in the camera so that p in the C frame, that's now the point in the camera frame. In homogeneous coordinates, it's just expressed like this. The top left three by three is the coordinate, the right-hand column is the translation. And that whole four by four is referred to as the extrinsic parameter matrix, okay? This is the thing that transforms a point in the world to a point in the camera frame. By the way, that bottom row is not so important unless we're doing inverses, that bottom row is what makes this equation invertible. So when sometimes we're doing projections we're going to use the three by four instead of the four by four, but don't worry about that till the next lecture

15 - Extrinsic Parameter Matrix Quiz.srt

That bring us to an interesting quiz. How many degrees of freedom are there in the 3x4 extrinsic parameter matrix? So 12, there are 12 numbers. B) 6, c) 9, or d) 3?

16 - Extrinsic Parameter Matrix Solution.srt

All right? Well, what's the answer? Well, the answer is still six, remember there were six degrees of freedom? There are, only three angles heading, pitch, and roll. Euler omega, phi, kappa, or whatever. There are, three angles that define that rotation matrix, so that's not nine independent numbers, all right, there's only three angles, and then there are three translational values, and that's why there are still six extrinsic parameters, even though we can use a three by four or sometimes even a four by four. So, we've just taken away of turning those six numbers into a matrix form that allows us to apply it to the location of the points in one frame, to get the location of the points of another frame.

17 - End.srt

So that ends the lesson on extrinsics or extrinsic. Not really calibration, because we're going to do the calibration part. So I should change the title of the, that was about extrinsic geometry. Later we're going to do extrinsic calibration where we figure out how a camera is oriented in the world. We're going to have to revisit this whole thing when we talk about mapping from world points to a location on an image plane, all right. But before we can do that, we're going to have to talk about, once I have the location of a point in a camera frame, where does that point end up in the image? And that's the intrinsic and we're going to do that at the next lesson.

18 - Outro.srt

So I don't know if you agree with Aaron or not, that he's funnier than I am. He thinks he is, I think he is too. We'll give him that. In that lecture he talked about extrinsic camera calibration. That is figuring out from information from a variety of images where the camera is in the 3D world that could actually help us extract information about 3D environments. Next, the lecture is going to be about intrinsic camera calibration.

# 05-08 Intrinsic Camera Parameters (opt).txt

01 - Preface Intrinsic Camera Parameters.srt

So this is the second in the series of three lectures on camera calibration. Again, I'm going to rely on my dear friend Aaron Bobick, and use his lectures on camera calibration. In the last lecture on camera calibration, he introduced the whole mathematics of how we do extrinsic camera calibration, that is, being able to capture the world information of where the camera is in an environment from a series of images. In this one we're going to talk about, more about the intrinsics, the focal length and other information related to the camera, that is, once the light enters the camera, how is it generated into an image. So that's what the focus of this one is. Again, it's a fun lecture, it's going to cover a lot of variety of topics. Pay attention to the concepts. Again, I wanted to introduce these ideas to you in an optional framework.

02 - Intro.srt

Welcome back introduction computer vision. Today, we're going to be talking about intrinsic camera calibration. Last time, we said, that we're going to do geometric calibration in general, and that there were two parts to calibration. The first transformation is from some arbitrary world coordinate system, to the camera system or the camera pose, and this was the extrinsic parameters, and it mapped from, world coordinates to camera coordinates, or camera coordinates to world, depending upon how you think about it. When we write it as T, W, C, it takes you from the world, to the camera. We expressed it in terms of homogeneous coordinates, where we had a world coordinate p here expressed in that world coordinate frame, and it was homogeneous so there's a one down there. And we pump it through both the rotation component and the translation component to get the three dimensional point in camera coordinates. And, that world to camera matrix and codes what were referred to as the extrinsic parameters or the extrinsic parameter matrix. We also said that, that encodes six degrees of freedom, three translation, three rotation. Today, we're going to talk about the second transformation which goes from the 3D camera coordinates to the 2D image coordinates or the 2D image plane. And these are referred to as the intrinsic parameters, and we'll again come up with the intrinsic parameter matrix.

03 - Ideal vs Real Intrinsic Parameters.srt

So, you might say, woah, didn't we already do this? We did the ideal perspective projection, where we said that some value u was just going to be the focal length times x divided by z, and v was, was y divided by z multiplied by f, as well. So you might ask, aren't we done? Well, no, because that would be in some idealized world. The first problem going back to here is, f might be in, you know, millimeters, so we might have a ten millimeter lens, or a 50 millimeter lens, but the pixels, the screen pixels, they're in some arbitrary coordinate, right? That depends upon exactly how many pixels we get per millimeter in the sensor. So the first thing that happens is that we introduce an alpha that's just going to scale that value, because we don't really know what f is. Now sometimes people will give you an f, a focal length, in pixels, which is kind of a weird thing. But what they're actually doing is, they're giving you this combined value that is sort of the conversion from millimeters to pixels times millimeters, just given to you, and pixels. But , because they may be in some arbitrary units, we have a scale factor alpha. So that's one degree of freedom. But, who said the pixels are square? Megan, did you say pixels are square? Now, it turns out pixels are more square now than they used to be. They used to be cool and now they're, nah, never mind. Anyway, pixels are more square now than they used to be. The used to be, back in television, more of television days, pixels had the same aspect ratio as actually a television. So a pixel was wider than it was tall, and some other things, they were taller than they were wide. They weren't necessarily fixed. In fact, even some CCD arrays that I calibrated once, it turned out that well, it was almost square. It was like 95% of the height was equal to the width, all right? So because they're not exactly square, you might have a separate scaling factor between the u direction and the v direction. And so now we've introduced beta. So now we have two degrees of freedom. But we're not done. Next, well you remember we put the center of projection, when we were doing the ideal projection. We put the center projection right at the camera coordinates system. As if the image was taken so that zero, zero was right in the middle. But , we don't have any guarantee of this, right? The image may have been cropped out of a section of the window. Or the, the location of the image actual sensor might need, might not be lined up with the optical axis of the camera. So we have two offsets, a u and a v offset, u0, v0. So now we're up to one, two, three, four degrees of freedom. Two scale factors and two offsets. Are we done yet? But wait, there's more. Here comes the really ugly one. We're assuming that the u direction and the v direction are actually perpendicular. What if there's actually a little bit of a skew? So u and v went out drinking one night, and they came back just off a little bit, all right? So that's what's shown in this figure here. The idea is that the ideal u and v are this way, okay? But maybe the sensors actually sampled that way and that way. That is, that the, the actual sampling of u, v is not perpen, are not perpendicular, and they're off by some angle theta. So that's what these equations are showing you here. They're showing you the relationship between the v-prime which is measured and the actual v, the u-prime and the actual u. And so, when you substitute those into those equations we just had, you get this sort of ugliness, okay? So this is the really ugly intrinsic parameter representation. And now we have, how many? Well we've got an alpha and a beta, the two scale factors. That's two degrees of freedom. A u0 and v0 for the two offsets. Plus theta, which is the skew

04 - Improving Intrinsic Parameters.srt

This is pretty ugly, and we'd like to make it nicer, and we're going to do that through two ways. First, so here we have those Euler equations and the first thing you'll notice is kind of like before, we're dividing the x's and the y's by z. All right. And so that should tell you that see I've wrote up here intrinsic parameters in non-homogeneous coordinates. Well, guess what? We're going to move to homogeneous coordinates by putting this whole thing in a matrix formulation. So now we can express the whole thing in homogeneous coordinates. Notice that here we have z times u, z times v, z, so later when we convert back from homogeneous to non-homogeneous, we divide by z, and we get what we want. We have the x y z one over here, and we have this matrix in the middle. So we can rewrite this as, sort of, this very simple equation where we have a three-dimensional point in the camera frames. So remember, we've gone from some world, arbitrary world frame to the three-dimensional frame of the camera. And we go from that to the homogeneous pixel representation, like that, in the image. And the matrix that takes them from the camera to the image, that's the intrinsic matrix. Okay, so that matrix represents the intrinsic parameters, all right. Now fortunately, we can make it look even nicer than this. The first thing to notice is that the last column of K, when I write K as a three by four, the last column of K is zeros. And that doesn't really do very much, so we can get rid of it. And then we can do even more. Here we have our kinder, gentler intrinsics. We can use a simpler notation. Like I said, we're going to remove that last column. And we've gotten rid of the explicit thetas and things. And you'll notice that we have the five degrees of freedom. We have f, which is focal length, a, which is aspect ratio, s, which is for skew, and cx and cy, those are the offsets. By the way, remember I said we can have two different scales, right? A scale for one, and, for u and a scale for v? Or what we can have is a focal length and a relative scale between the two. Normally, we tend to think of it that way, as a focal length. That's the overall focal length of the image. And then, if there is a non-uniform relationship between the width and the height, we include that as an aspect ratio. And that's why there are five degrees of freedom, okay. Now, it turns out, this can get even easier, all right. And the way it gets really easy is we assume a certain niceness of the universe, okay. The niceness of the universe that we might assume is, if we have square pixels, if there's no skew, and if the optical center is actually in the middle, okay. Then we have no a, we have no s, we have no cx, we have no cy. All we have left is f. F is the only degree of freedom left. So when you're doing a calibration, sort of a lightweight calibration, what you'll do is you'll just search for f, assuming that your optic axis is in the middle, assuming there's no skew, and assuming that your pixels are square.

05 - Intrinsics Quiz.srt

Quiz. The intrinsics have the following: a focal length, a pixel x size, a pixel y size, two degrees, offsets and a skew. That's six. But we've said there are only five degrees of freedom. What happened? A) Because f always multiplies the pixel sizes, those three numbers are really only two degrees of freedom. B) in modern cameras, the skew is always zero so we don't worry about it anymore. C) in CCDs or CMOS cameras, the aspect ratio is carefully controlled to be exactly 1.0, so we don't model that anymore either.

06 - Intrinsics Solution.srt

Well, the answer, as I said, is a, right? We have, the f can be multiplied by a for the two different scales or we can have the two different scales. But there aren't three numbers there. There's actually only two. And so the answer is a.

07 - Combining Extrinsic and Intrinsic Calibration Parameters.srt

So now we found the intrinsic matrix. And last time we found the extrinsic matrix. So now we can combine them to get the total camera calibration that goes from a world point all the way through the camera coordinate to the image. So we write down our two equations here. We have our intrinsic, p prime is equal to K times the point in the camera frame. And our extrinsic which relates the world point, to the camera point. P in w is the world three dimensional coordinates transformed by the extrinsic matrix, becomes the camera three dimensional coordinates, which is used also in the intrinsic equation. And then that's converted to pixels directly. One thing to note here, is that our world coordinate system, is a four vector, it's a homogeneous. And we get out of four vector, but as we said before, for our K, instead of using the three by four, we can use the three by three, in which case we just use the x, y, z of the point in the 3D camera space. We don't have to use the whole x, y, z, 1. That's saying that this K can be thought of as a three by four or a three by three. And when it's a three by three we don't have to have that 1 on the bottom there. So, putting these two equations together, what we have is we take a world point here, we pump it through our extrinsic. So, this is a three by four matrix, okay, which gets us out of three dimensional vector. And then we pump that through our intrinsic matrix, and that get us our homogeneous image coordinates. So this is a three vector. Remember we said it was z times u, z times v. Look at that. My us and vs look the same. Divided, and z, and then we convert it back. And so this whole thing can be written as a single matrix M. M for calibration in some language. I have no idea why it's called M, but we call it M.

08 - Other Ways to Write the Same Equation.srt

Just want to write that, M in a slightly different way, because it's the way that we're actually going to make use of when we solve for this thing. So, here we have the same equations written out. You see, we have the world coordinates mapped through M, gives us the homogenous pixel coordinates. And what I've done, written here is, I've taken this M matrix, which, remember, is a three by four. And I said that you can think of it as three vectors each of length four transposed as the rows, so each of these rows is a vector. I've also introduced something new here, I don't think I've showed it on this slide before. Remember, this is what we said before, okay? I'm, I'm using s as a scale, instead of z, right, and so we have s times u, s times v, and s. And later to get out u and v, we just divide by s. And I've put this little operator here, but it's just a squiggle with a straight line underneath, sometimes two straight lines underneath. And it's what's known as projectively similar. So you'll notice that this vector and this vector are exactly the same except for a scale factor, all right. The vector on the left multiplied by s, gives you the vector on the right. And remember, in homogeneous coordinates or using the projective, those two vectors are essentially the same, because when I use their values, I divide out by the left, by the last, component. So, that's what's referred to as projectively similar, so that's also introduced here. So, just finishing, the way I recover u and v is, I divide the dot product, the dot of the vector of the point p in the world with the first row, divided by that dot product with the third row, that's what's right here. And then to get the v value, I do the same thing, but now with the second row, all right. So , what we have to find, when we're going to find, when we're going to do camera calibration, is, we have to find those m elements.

09 - Camera Paramerters.srt

So finally, we can talk about full camera parameters or camera and calibration matrix. The camera and its matrix M, and sometimes it's called pi, and that's what I've written here, as we know, is described by several parameters. It's got a translation T of the optical center from the origin of the world. We've got a rotation R of the coordinate system. We've got a focal length and aspect, f and a. Pixels or, or pixel size, sx sy. A principle point, that's the offset x, xc, yc or cx cy. And skew. And in this slide, the blue parameters are called the extrinsics and the red are the intrinsics. And we can put the whole thing together, we want to find this as a sort of a single matrix. M is going to be built up of all of the effects of our parameters, and so that looks like this. Okay? What is says is that M is a combination of translate, this is extrinsic, and rotate. So now we have the point in the 3D camera coordinate system. Project, this is just the extraction of the xyz, and then you pump that through the intrinsics. And how many degrees of freedom are there here? Well, there are 11. Five for the intrinsics, three for the rotation, three for the translation. I'll get rid of all that scribble. That equation there, that M, that is the full camera calibration matrix.

10 - End.srt

All right, that ends this lesson. What we've done so far is we've derived how the extrinsics and the intrinsics are combined to form a single calibration matrix. And this matrix maps from some arbitrary world coordinate all the way down to a pixel value. What, you know, some x, y, z in the world, what UV does it end up in the, in the image. All right? So what are we going to do in the next lesson? Well, the obvious thing in the next lesson is, how would you go about finding that matrix, all right? And , the requirement's going to be, if I gave you the location of some 3-D points in space, and I gave you the location of those points in the image, I should be able to recover that camera calibration matrix. And that's what we're going to do next time. And then, a couple of times after that, we'll say, well if we're less interested in going from a world coordinate system to a camera coordinate system, but instead, we're interested in going from one camera coordinate system to another, remember that stereo thing? How would we do that calibration? But first, we go from the world to the image.

11 - Outro.srt

So, hopefully, you liked that lecture. Now you actually know both about extrinsic camera calibration and intrinsic camera calibration. So, you know about the world information from where the camera is to insides of the camera, the focal length and stuff. Again, computed from data captured by a camera. In the next lecture, we're going to get more practical and think about how we can actually do this camera calibration in certain environments.

# 05-09 Calibrating Cameras (opt).txt

01 - Preface Calibrating Cameras.srt

So this third and final lecture on the topic of camera calibration, again, I'm going to show you a video that was recorded by Aaron Bobick for his computer vision class on camera calibration. In this one, he actually takes much more of a practical approach and teaches you how we're going to really do camera calibration in an, a real environment. He'll showcase variety of things, like, for example, using checkerboard patterns like this, where we know the size of each and every part of the cell here and knows the dimensions which can be used as real in, information that could be then used to calibrate cameras. Again, you hopefully you'll enjoy it. This is much more of a practical approach on how we can do camera calibration. And, again, this is optional material that I just want you all to know about.

02 - Intro.srt

All right. Welcome back to com, Introduction to Computer Vision. Today finally, we're going to get to talk about calibrating cameras. Where we're going to be calibrating with spectra 3D world. So finally, we can get to the parameters of cells. If you remember last time, we solved for the full projection equation. That was made up of really a composition of the translation, rotation, projection and intrinsics. For the rest of today, we're not going to worry about the fact that M has this internal structure, till maybe the very end. We're going to think of M just this way. That is M is going to be this matrix that's going to take your world points and homogeneous coordinates. And eventually, get you out your image points also in homogeneous coordinates. So what we're going to try to do is find that M.

03 - Calibration Using Known Points.srt

Fundamentally at the heart of calibration is this idea of having some points whose three dimensional location in the world we know, and that we identify them in an image. That is, we can find the correspondence between this point in the image is that point in the world. And then we have to compute some sort of a mapping from a, a scene to the image. And one thing you can do is you can actually put an object like this in the scene that has a bunch of points on it. And you've measured or you know something about the shape of that object and you know where all the points are. Another way of doing it is to make use of, I mean, it's sort of the same math, but it's typically referred to as Resectioning. Which I'm pretty sure is a term that came from photogrammetry, which is a science that predates computer vision by about a century, that talked about going between images and three dimensional world, typically used for mapping, cartography, and stuff like that. So in Resectioning which is what we're going to do here. The basic idea is we're going to have some known points. So on the right here we have a picture of one of our labs. And one of the times one of my colleagues was using a theodolite. That's the thing that surveyors use in order to measure things out, you know, for property or houses. And we set up the theodolite in the lab and we established a world coordinate system. And you'll notice we put down these markers. These are actually sort of printouts of what surveyor marks look like. And what we did was we measured the three dimensional location in the world, with respect to this coordinate system, of those points, all right? So what that does is that's going to give us a set of points, X, Y, Z. And then , given an image like this one, we can find the location of uv in that image. Notice again that I'm using the homogenous version of the points. Here I'm calling it w. So this is w, u scaled by w v scaled by w, and that's w. And this was that original equation. So clearly, given enough points in the world and the image, I should be able to calibrate, recover the calibration matrix. And here's how.

04 - Direct Linear Calibration Homogeneous Part 1.srt

All right, so first, some notation. As before, we've got our world times our m is our projective equation. And notice we've got two versions here. what we're saying here is that uv1 is projectively similar to wu, wv and w. And in fact, in order to get back uv1, I could take wu, divide by w in order to get that. And that's actually what's written on the bottom. All I've shown is that ui, that's for point i, I have to take the product that's the first row divided by what becomes the third row. Same thing for vi. So the point here is that I get a pair of equations for every point. For every point i here, I have an equation that's, that goes from x, y, z to ui, and x, y, z, to vi. So what we're going to do is we're going to have to solve for this m matrix using a, a whole bunch of points. Just continuing, I just slid those equations up, and now I've just carried through, just expanded it through. And one thing you'll notice is that every term has an m in it. And for those of you who know anything about linear algebra, that should start to worry you just a little bit because typically when I want to solve a linear set of equations, I hope that there's some term that doesn't have a variable in it. And we'll see that in just a sec. So, I have this pair of equations for each point, all right, writing it again like that. So now the question is how best to solve this, all right. I'm going to write this a little bit different way. So here I have all my variables, and here for just one point, I have my coefficients. So notice my coefficients involve capital Xs and Ys and Zs. Those are the values out in the world, along with the v's and the u's, which are the points in the image. Now, stretch way back into your memory from linear algebra, and you right, might remember that usually you were solving equations of ax equals b. And you could have more equations than unknowns, and you'd solve to a least square solution. But there was a particular kind of sol, of set of equations where the right-hand side was referred to as a homogeneous set of equations when they were all equal to zero. And that's why this is a homogeneous set of equations, not just homogeneous coordinates. These things are all sort of related. And when you get really, really old, you'll figure that out. But for now, you can just remember that these are homogeneous equations. So the question is, how do we solve that? Because obviously, there's a trivial solution. What's the trivial solution? Well, if a times x equals 0, and I'm looking for an x, well no matter what a is, x equals 0 would be a solution. So I could just say all the m's are equal to 0. And that would not be a very satisfying solution. So the question is, what's the best way to solve this?

05 - Direct Linear Calibration Homogeneous Part 2.srt

So as I said, this is a homogeneous set of equations. Now, when you have a homogeneous set, like Am equals 0, and let's suppose you have a lot of equations, well, clearly what you'd like to do is you want to place some constraint on m. because I don't want to let it be 0. But then given that constraint, I want to find something that sort of computes the smallest square value of A times m. And that's what's written here. So I'm going to try to minimize the magnitude of Am. Now, m is only valid up to a scale. And we know that for a couple of reasons. Right? One is you can just take a look at these equations here. Right? Since, the everything's equal to 0, I could just scale m. Or I can go all the way back to my perspective equations. because clearly if I scale all those ms by a value, when I go from the homogenous to the nonhomogeneous, when I divide uw by w, that scale factor will go away. So m is only defined up to a scale, that matrix, all the little m's. So since it's defined only up to a scale, let's just assume it's going to be a unit vector. Okay, so it, it's magnitude is going to be length 1, and the question is, what's the best unit vector m to minimize the magnitude of Am? I'm going to tell you the solution here, and then I'm going to show you that it's the solution. And some of you will have seen this before, and some of you won't. The solution to minimizing Am, the magnitude when Am as a unit vector, is, as you all know I'm sure, because you've studied this all the time, you want to take the eigenvector of A transpose A as the smallest eigenvalue. , you say. I know that all the time. I'll show it to you in a minute. And by the way, this only works when you have six or more points, and that should make sense right? How many degrees of freedom are there in m? There's 11, because it, there's 12, but only up to a scale. Well, I get two equations per point, so I'm going to need at least six points. But this thing works a lot better the more points you have.

06 - The SVD Trick Part 1.srt

All right. So let's go through the solution. So here I've written as if I only had two points, point 1 and point n with a whole bunch of points in the middle. Okay. So I've got the matrix A times m equals 0. How big is the matrix A? Well, it's 12 across. Right? Because there are 12 ms and how many rows does it have? Well, it has 2 times n, where n is the number of points. So it's a 2n by 12 matrix. M is just and m by 1 vector and here's the 0 vector of length 2n. Okay? So now let's talk about solving that. One step at a time here. All right. So here's our goal, right? We said, we want a unit vector that minimizes Am. A unit vector for m that minimizes Am. So let's do what's called the singular value decomposition and I'm going to assume that you've seen that before, at least some place. But the idea is that any matrix can be decomposed in to this UDV transpose format where V is a real orthogonal matrix. So in our case, it would probably be a 12 by 12. U is going to be a much bigger matrix. It says, orthogonal, but I should, can be more precise. Each column's orthogonal to each other, so that would be, in this case a 2n by 12. And most importantly, D is a diagonal matrix. So it only has values in the diagonal and it is customary to write it in decreasing order of absolute value. So from the biggest D to the smallest, all right? So we're going to write A is equal to UDV transposed. So minimizing Am is the same as minimizing UDV transpose m. Got it? So far nothing significant other than you have to use your singular value decomposition code from Matlab. Here comes the first little bit tricky part, that is that the magnitude of UDV transpose m is equal to the magnitude of just DV transpose m. Why? Well, these are just unit vectors. Right? The U's and V also for that matter are made up of orthogonal unit vectors. So multiplying them doesn't change the magnitude of the matrix and that's why these magnitudes are the same. And it's also why the magnitude of m is equal to the magnitude of the transposed M. For those of you who think about such things, V in this particular case is a orthogonal. In this case, 12 by 12 matrix. You can think of it as like a rotation matrix. So when you rotate a vector, you don't change its magnitude. So the magnitude of m is the same as the magnitude of V transpose m. So that means instead of minimizing UDV transpose m, we can just minimize DV transpose m, subject to, instead of being subject to m equals 1, subject to V transpose m equals 1. Got it?

07 - The SVD Trick Part 2.srt

Now you might ask, why are we doing that transformation? >> Why are we doing that transformation? >> Here's why. It's going to let us do some very cool substitution. So we're minimizing DV-transpose m. Let's do a substitution of y equals V-transpose m. So now we're minimizing just Dy. Subject to the magnitude of y equals 1. Right? because V transpose M is y. Its magnitude has to be 1. Okay. So y is a unit vector. Why? Why not? No, never mind. Because it is. All right? But think about D times y. All right? Dy right there. Remember, D is a diagonal matrix whose elements are in decreasing order. So, what is the smallest that Dy can be? When is it the minimum? Well, it's going to be a minimum when y puts all of its weight, remember y is the unit vector, just in that last element, all right? So the best y for minimizing Dy given that D is a decreasing order diagonal matrix is just 0,0,0,0,0,1 okay. That's the y that minimizes Dy. And, since we defined y to be V transpose m, then m is equal to Vy. Why? Well, remember I said that V is orthogonal. Right. So when V is orthogonal its transpose is its inverse. So if y is equal to V transpose m, the inverse of that is just V, so m is equal to Vy. So now we've solved for m in terms of y, but, V and y, but remember. Y is just 0,0,0, all the way 0,1. So that means that this equation just pulls out the last column in V. All right, because if I take a matrix, all right, and I multiply it by some vector that's all zeroes down to 1. That 1 just multiplies the last column. So I just pull out the last column.

08 - The SVD Trick Part 3.srt

So, m equals Vy is the last column. And, here's something, just because of what we said before, these singular values of A, that's d, well the singular values of A are the square roots of the eigenvalues of A transpose A, and the columns of V are the eigenvector of A transpose A. So, I could show you this, it's actually pretty, well, I was going to say but you know, if I just write out this, since A is we know is written to UDV, transpose. Okay. A transpose A, is just that transpose, so that's VD transpose U transpose. Well, okay, U is an orthogonal vector, so U transpose is also U inverse, so that's an identity, okay, so that goes away. D transpose D, well D is a diagonal matrix, so D transpose D is just, I'll, I'm just going to call it D squared, it's just the squared values, okay? And V transpose V. So, I'm just, so I'm just going to write that A transpose A is equal to VD squared V transpose, and that is the equation of the Eigen-decomposition, where V is the, eigenvectors of A transpose A. And by pulling out the very last one, I'm pulling out the eigenvector with the smallest value of A transpose A. Just to recap. Given that, what we have is Am equal 0 for some long A, what we do is we find the eigenvector of A transpose A with the smallest eigenvalue, and that's our m. That is the set of values that create the calibration matrix for us. Cool.

09 - Direct Linear Calibration Inhomogeneous.srt

How about another way, okay? I'm going to show you another approach. Which in some sense is easier to understand, but it's actually not as good. That's why I showed you the other one first, all right? So here I'm using the same equation again. Now using uv1 as being projectively similar to m times XYZ1. But remember, if m can be changed by a scale value without affecting anything, then I could divide all the values by whatever that was on the bottom right-hand corner of m. And what I would get out would be a 1 here. So I can just put a 1 in the bottom right-hand corner of that matrix. And the reason I do that is now my equations for Ui and Vi become this. All right? And what you'll notice is I now have terms when I multiply out. This, the 1s and the denominator multiply Ui and Vi, which remember I, those are knowns, because those are the locations of the points in the image. So I now have terms that don't have the variables in them. That's why this is referred to as the inhomogeneous solution. And one of the reasons this is not as good is suppose the original m23 was supposed to be much, much smaller than the other values, that is close to zero compared to the others. Well you've just set a number that was supposed to be really close to zero to one, that's dangerous with respect to numerical stability. So in general and, and also in terms of overall minimization, doing the singular value decomposition or doing the eigenvector finding on a transpose a, is, is the preferred method. And, by the way, it's one line in MATLAB, so you might as well just do it.

10 - Direct Linear Calibration Transformation.srt

So what I just told you is referred to as the direct linear calibration or transform or transformation method. It has a couple of advantages. One is, it's pretty simple to formulate and to solve. Like I said, it's literally, you just take your points, and you make that a matrix. You then do some SVD. You pull out the column. Bang, there's your m. Now these methods are referred to as minimizing an algebraic error because essentially what we did was we set a m. We fixed m to be a unit vector and then algebraically forced found the m that gave us the smallest algebraic magnitude of a m. There are some disadvantages to this method. First is that it doesn't directly tell you about the camera parameters. But we'll get, we'll fix some of that in just a minute. It's also approximate because that m was based upon a particular matrix based upon pure extrinsic and intrinsic projection. What if we had something like radial distortion that we could actually model, but we couldn't model within our projective transform equation. How would we do that? Suppose I actually knew the focal length, right? I went out and bought a really expensive lens with focal length 3.746586, really precise millimeters. I don't want my m screwing with that. It needs to stay that value. I want to come up with the minimum solution using that precise value. And then finally, another problem, I put down mostly, I think it's just another one of those problems. It's not actually minimizing the thing that I really want minimized. It's minimized this kind of cute algebraic trick that I had. So, what is it that I actually want to minimize? Clearly, the amount of weight I put on every holiday. But besides that, what? Okay, that was not even a funny joke, but it, it was going too long without any jokes.

11 - Geometric Error.srt

So what is the right error function? Well, we refer to this error function as the geometric error, and it works as follows. Let's suppose I have some points in the world capital Xi, and here they are shown in yellow. And furthermore, let's suppose I have an image and I know where those points are in the image. And those are the, the red Xi's here and these are these little red points here. Now, when I specify a camera projection, some sort of an m. Right? What I'm saying is I know where those capital Xi's, the real points, are supposed to project to the image plane. And what I want to do is I want to find the M that gives me the smallest distance between the predicted X size and the ones that I actually found. And that's what this equation here says. What it says is we have an error function that is the sum of the distances between the observed points, those are the red Xi's. And the predicted image locations, the white Xi's here of, of where the, the particular M predicts that the points in the world would land in the image. And what I want to do is I want to find the M that minimizes that value. And now what's kind of cool is, I might have a more complicated mapping from points in the world to my 2D points that include things like, modeling radial distortion, but fix for me the focal length. The idea is I'd have some great big nonlinear function that does the full projection. And what I want to do is I want to minimize this error function by min, by manipulating the parameters of that big nonlinear function. And you might do that using your favorite Levenberg–Marquardt or some other way of doing nonlinear minimization. So if you read about this in a book and the, it says, the Gold Standard. Well, the Gold Standard of this type of work is the book on Multi-View Geometry by Hartley and Zisserman. What they refer to as the Gold Standard of calibration is, Given some number of points, more than, greater than or equal to, but should be greater than 6, where you have 3D to 2D point correspondences. The way to find, they refer to it as the Maximum Likelihood Estimate. We could talk about why that's the case later, I'll tell you right now. You're assuming that your the place that you located the points in the image was perturbed by a little bit of Gaussian noise. So you maximize the likelihood by making that as small as possible, and you want to minimize the square error, if it's a Gaussian error. So the way you do that is as follows. You first compute a linear solution. Now Hartley and Zisserman talk a little bit and, and this is for those of you who are going to be more involved in doing some of this. What you might want to do is, you know, if you're measuring your points out here in meters, you might have some big numbers or small numbers depending upon your coordinate system. Your size of your image, you, you may have different pixel values. They do a normalization, so that everything sort of goes between zero and one to try to get numerical stability. So when they do that, they, they create new vert, new external world points and new image points. You can do that or not. Then what you do is you solve the thing I showed you before. The direct linear transformation using, typically, the eigenvector approach. That gives you a starting point for the M matrix, which gives you a starting point for your nonlinear function. And then what you can do is you can try, u, u, using the M, you project the points. And then you can minimize that total distance that's what's written down below using, like I said, your favorite nonlinear method. After you've done all that, if you've done the normalization, you have to denormalize or, or, or translate back from the, the normalized methods. So that's how you get out your M. And that's what they refer to as the Gold Standard of calibration. We only use bronze, which is why in your problem set, you're going to use just the direct linear method. And you're going to see that even without, I, I will tell you this, that the first time we did this we had people do the normalization and it turns out the normalization by just some, but not a lot. When we do it this time, you can use the normalization or not.

12 - The Pure Way.srt

So now we didn't have M. Now we know, that M encodes all the parameters, the extrinsic parameters and the intrinsics. So we should be able to find things about the camera from M. For example, the focal length which should be an intrinsic. Or, how about the camera center? In order to be able to do the projection, the camera center, remember the translation vector that translates from the world coordinate to the camera center. That, if we just knew that translational value, that would tell us the location of the camera center in the world coordinate frame. So we should be able to get that directly from the end matrix. So I'm going to tell you about two ways of doing that. Sort of the pure way, which is beautiful, and then the easy way. So we'll start with the pure way. Here's a slight change in notation just to pull out our parts. Okay. So we have our matrix M, and I'm just going to assume that M, which is a 3 by 4. Is going to be made up of this 3 by 3, which I'll call Q and b is just this 3 by 1, it's just a, a vector. Okay, so we've got Q and b. I'm going to claim that the center, C, which is a point, which I'm going to represent as a vector, is in the null space. Of the projection matrix. And what that means is that if you multiply M times C, it equals zero. And if you found such a C, that would be the camera center. And you should just trust me. Okay, maybe you don't trust me. Let me show you why that's true. All right? It's really pretty simple. Except drawing it is painful, all right? Let's suppose I have some camera center C. And I've got some plane. And I have some point P that's out here. So I have a ray that goes from P to C. And I'm going to have a point X that's somewhere on that ray. Okay, so that's where X is. All right? So that's all that this equation says. That X is equal to something that's a blend between p and c. Alright, lambda times P, one minus lambda, times C. So given some M, that's the camera, the projection. All right? Of X is just MX and just by linear algebra, that projection is just lambda MP plus 1 minus lambda MC. Now for any point, P, all the points on that ray have to land at the image of P. So, any point P, all the points on this ray, have to land at the same point on the image, okay? So, any point X along here will land at that same point, no matter what lambda is. Well, if this equation has to be true, no matter what lambda is. And it has to land at MP because lambda might be one. That means that this value, MC, has to be equal to zero. So if C is the center, MC equals zero. And that's why, I could, I said before that the center C can just be found by finding the null space. Of the projection matrix M. That wasn't too painful.

13 - The Easy Way.srt

But suppose you actually wanted to like do something useful. What would be the easy way of doing that? Well the easy way is I just give you the formula. So, if M is equal to Q b as I said before, then, the center is just found by taking this quantity above that quantity in a vector. So this is a, remember Q is a three by three, so minus Q inverse is a three by three, b is a three by one, so this is a three by one, and then I add the one there, that is the vector that is the camera center in the world coordinate frame. So if I made, you know, the corner of that table that you can't see, there's really a corner over there, if I said that's the origin, right, and this way is x, and, this way is y, and that way is z, and then I have a camera here. And I, took a look at a bunch of points here, and I gave you the value of those points in this coordinate system, and then I located those points, in my image plane. And then I did all that math to compute that M vector, and that, that M matrix, and then I pulled out the three by three and called that Q and three by one and called that b. And then I took minus Q inverse times b over one, that, vector would be the actual location of this camera in that world coordinate system. Pretty, pretty cool.

14 - Multi Plane Calibration.srt

Finally, and this is the half bit that I was telling you about, there's an alternative way of doing calibration, which is pretty cool. And in fact, it's really what everybody is using today, because typically you don't have all these points. And it's a way of getting the intrinsics, and then if you know where you put these checker boards it's extrinsics. what you do is you take a checker board and you move it around. And you can end up calibrating the camera, with respect to that checkerboard. The reason everybody uses that, these days, is that there's. First of all, you can print out a checkerboard and it the checkerboard has to be on a plane, well that's easy you print out a checkerboard. You got to make sure your printer doesn't screw things up, you got to make sure that it actually comes out that squares are squares. You mount it on something that's a nice piece of rigid cardboard or foam core. And you don't have to know the positions and orientations in order to get the intrinsics of the camera. And later you can get the intrinsics as well. And why does everybody use this? Because the code is easily available online. OpenCV supports this directly. We list here a MATLAB version of it. If you go to Zhengyou Zhang's web site, he's like doctor calibration. He did spectacular work for us, thesis etc., in calibration. I'll also tell you that there's something cool you can do, and in fact, the a student of mine, Kelsey Hawkins did this. And I, I think the code is made publicly available. If you take a checkerboard and you mount it on the end of a robot arm. And remember, a robot knows where its arm is in terms of its coordinate system. If you also know let's say the offset of that checkerboard from the end effector, the hand. Then what you can do is with a camera you can just move that checkerboard around using the robot, and we know the 3D location of the checkerboard at every point in time that we take an image. And so, by taking those images of the checkerboard, you end up calibrating the camera to the robot's coordinate system. So, it's a cool use of doing this directly within robotics.

15 - End.srt

So this concludes the lesson on calibration. In the homework, you're going to use the theory we developed for direct linear calibration, and we'll give you those images that have the three-dimensional locations of the points and the two, and you get the images. And you have to solve for the calibration. And it's, it's interesting that you can actually recover that relatively precisely. In the next lessons, what we're going to do is we're going to move away from the notion of rigidly calibrating the camera to the world. And instead, we're going to talk about the relationship between multiple cameras, specifically starting with two cameras. And that's going to eventually get to the point, where we were talking about in stereo, that if you know the relationship between cameras, that you can know where the epipolar lines are. So , if I gave you some corresponding points, you should be able to solve for the relationship between the cameras that will allow you to find those epipolar lines. And that's what we'll be doing over the next three or four lessons

16 - Outro.srt

So this was the third and final lecture on camera calibration. The series of three lectures that Aaron Bovick talked about. Camera calibration starting from extrinsic parameter calibration, then intrinsic parameter calibration, all the way to a practical guide on how to use checkerboard patterns to calibrate in a real world. This is an important part of what we need to know about cameras in general. While it may not be directly relevant to the kinds of stuff we have talked about so far, it's an important foundational principle of what we can do with computational photography. Especially when we want to get into knowing more about the structure, the range, and the depth of a scene. And hopefully that's what you found interesting about it. More on this topic, we'll follow up when you get to topics like video. We won't be talking much more about calibration anymore in this context. But again, as I said, its a foundational thing, I want you all to know about.

# 06-01 Video Processing.txt

01 - Introduction.srt

So far, we have learned a lot of different fun things to do with images. Now let's get to an even more interesting topic, and that is how to do similar kinds of things of both analysis and synthesis, but apply to video. The simplest way of thinking about it is, images over time, a sequence of images forms a video. In this lesson, I'm going to introduce a variety of methods or representations that will let us do the same kinds of things that we've done with images, but the constraint of time or sequence of time will be added to it. But look at the representations and actually start thinking about how we can apply some of the things we have learned in respect to images applied to video.

02 - Lesson Objectives.srt

The specific objectives of this lesson are for you to learn about the relationship that exists between images and videos. We will talk about the whole concept of persistence of vision. because, that is the key process that helps us understand how we can actually play videos, and see a flicker free motion of things moving around in video. We'll talk about the whole concept of how filtering and all the things that we have learned about images. How can they be applied to video? And then we'll talk a little bit about how we can actually do feature detection to tracking and all of that stuff in video.

03 - Recall A Digital Image.srt

Recall that we started the whole conversation around computational photography by looking at how we're going to capture 3D world with all of the illumination associated with it into an image through a sensor and a set of optics. The sensor gave us, among other things, an image representation in terms of pixels. That's what we looked at, and here is example of the image we'd looked at before. We're just going to look at a black and white image of this. And we said that, in essence, this image has a certain width and height, and that helps us look at concepts of what a megapixel image is. So a digital image in that time, and we discussed it, was a numeric representation in both dimensions, in x and y. And this is the coordinate frame we looked at. And we looked at the whole concept of how we can access any part of this image by just traversing through the rows and columns in x and y for continuous form, and is and js, , again, looking at each and every pixel in discrete form. This allowed us to look at concepts of image resolution. We looked at image resolution as width times height of an image. And we made claims that each element, or each pixel, picture element, contains light intensities for each value of x and y for the whole image. And that's how we accessed it. , we learn how to do filtering and everything associated with that later.

04 - Video Images over Time.srt

So our video, as shown here by this candle that's flickering here, it's nothing else but a stack of images on top of each other that are displayed to us over time. So we still have the x and y coordinates we had before. And, , the stack of images means that we put them on top of each other. And, , as the next one comes, and in this case I've actually also layered them to kind of show you that time is progressing in this direction. So this is the time axis. , they're off-setted here to showcase the details of this. So digital video essentially is, same as an image in numeric representation in two dimensions, x and y, stacked in time t. And, , just like we did with images, now we actually will look at a three dimensional data structure where we can access any information on this stacked image here by traversing this in x and y here and in time. So this becomes accessing things in a continuous function, I(x,y,t) and a discreet function (i,j,t) in a discrete formulation. So, similar to what we've done for images, except now we also want to traverse for information in this dimension, the time dimension. In case of video, just like images, we have to discuss the concept of resolution. It's expressed as a representation of width and height of an image. But, usually, and this is again more standardization exists on video and actually was much more standard until recently, and most of the videos you see are in the aspect ratios of four by three. So, most television footage that you got in the old days when you had bigger TVs was four by three, much more standard definition. As we move to higher definition, we move to resolutions of sixteen by nine. An important thing to note, even when I'm showing here a video which is square, most of the time, you never actually saw videos that were square. You always saw them in aspect ratios of four by three or sixteen by nine. , only recently have we started seeing videos that are actually in portrait form, not landscape form. And that is because again, the artifact of using handheld cell phone cameras which usually have this form factor as opposed to this. But mostly, and one of the things that's standardized in video, , both the resolution and the aspect ratio. it's pretty much driven again by the display mechanisms, because most of the videos are watched on televisions and they're, , of specific resolutions. , these days, we're going away from even just standard resolutions of sixteen by nine, into real 4D or 4k videos and stuff like that which have more resolution, the aspect ratios are still being kind of made consistent. But, a lot of changes are coming in this discipline. The other thing that is different about images on a videos, is again that it has different types of file formats. These file formats include the images, and , what they also include is information about what frame rates to play these images at. And additionally there is much more information about, what kind of compression code acts are being used and what kind of wrappers are being used. And , there are lots of different file formats available for videos. I'm sure you've played around with things like AVI, MPEG, MP4 and what not. Again, we were not going to get into any conversations about compression in this class, but there's a lot of literature out there you can look at. But, overall we're interested in being able to access the pixels over time to be able to do the kinds of processing we're interested in, and that's what this whole lecture is about, to get you thinking about that concept.

05 - Persistence of Vision.srt

Let me talk to you a little bit about what's the underlying concept of what makes video what it is. And the whole concept comes through some literature that actually comes from psychology or understanding of the human vision system. Some of it is controversial and not completely bought into. But the idea really is what is referred to as Persistence of Vision. Our human vision system, when you actually show it frames of something, and if you could take those frames and, , look through them very fast, that is being able to change from one frame to the other at certain frequency. Our vision system merges the information, such that we actually don't see any flicker. And that's a large part of what the whole concept of Persistence of Vision is. That our vision system, when shown something at a certain refresh rate, merges the information such that we actually see motion that's flicker-free. So, for example, here, this is very classic example of a horse, and this were bunch of pictures taken, one by one, and then put on top of each other and that gives you the fact that this horse is running with the jockey on top. Again these were individual images, taken one at a time, placed in front of the horse as the horse was moving. Again, they were lined up at different locations. And, now if you notice, by just aligning them and being able to display them, you see motion, and a very continuous smooth motion. , in this case, you might see a little bit of jerkiness. So, in essence, the idea is, if you give me a bunch of image frames, and if they're captured in such that when we can play them back at a rate somewhere around 124th of a second, we will see flicker-free appearance of motion. And this is again a very classic example from 1887, by the way, of how this kind of stuff was used to generate motion. This another example of these birds here is again showing motion . They've been actually multi-exposed into creating one frame of all of that motion. So this concept of Persistence of Vision is, , the foundational observation of why we perceive video. Again, video is a medium, stacked frames of continuous motion refreshed at more that 124th of a second, gives us an appearance of motion and we don't see any flicker. So this actually is also the rationale behind building and inventing video cameras. We've talked a lot about cameras coming up in the early 1800s, and some of the earliest technologies of showcasing frames of videos, including things like zoetropes and stuff were in essence about taking images and showing them to you in a rate faster than our human vision system can actually deal with. And again that's one-24th of a second or more, And when that happens, we see complete flicker-free motion. Again, examples of this are there. And this is actually pretty much the foundation of why people wanted to invent cameras, video cameras, that can capture these frames. Some of the earliest work in this space of pretty much building these ways of kind of taking sequences, images and then using that to understand motion, and that's what he was interested in, was done by Muybridge. And in this case, in fact, this video that you see of the horse was done by him. , what he did was he used stop-motion photographs to study animal motion. So, yes, this is, in essence, stop-motion photographs, photographs taken by a bunch of cameras that fired as the horse was moving in front of them. And then, , using them to generate this whole motion, and a stabilized version of this. Another person, who actually at the same time frame as Muybridge, did work in this, too, as well, is Marey, and he developed a, a technique for chronophotographie to capture motion, and this example image is from him. Both of them were pioneers and actually had allot to do with the basic concept of what video motion capture that allowed us, in fact both of them were interested in actually understanding behaviors of animals and people as they moved. They started using cameras and video cameras to support this.

06 - Filtering Video.srt

Now we have spent a lot of time looking at various methods of processing images. So in essence, the same kinds of stuff can be done for videos. So rather than apply to individual frames, we want to apply it to whole video volume. Again, remember, in essence, what we're talking about is now, , this a stack of volume. And when the t deck sees this putting up this way, x and y, sorry, y and x are still there. So, in essence, what we're trying to talk about is how we're going to now take a video volume, and this is the number of frames, how would we actually run the filtering mechanisms we've talked about. So all of the stuff that we learned about convolutions, cross correlation, and using those methods can be applied in this case to a three-dimensional dataset. So , we can do all of this stuff in 3D. What we do is, our space that we interact with is x, y, and t. An interesting point to also remember is that this kind of motion information and filtering is widely used in compression for videos too, not only do in images. Remember we talked about compression briefly. We just did this in space, that is in x and y, but now we can actually do this x, y, and also use the motion information in t, to compress the amount of data. Similarly, all the stuff we looked at, change detection for doing gradient computation, we looked at how to do this in x and y where we can do this thing also in video. Except that this time around we can apply an x and t and also y and t, besides x and y. So now we have three different dimensions. So simple observation that I like to make right now is that if I have two images, and if each and every pixel from that one image to the other image is different, then it's just likely that there is a drastic change, or a motion change from those two frames. So, for example here in this image, you're seeing a lotta changes, but they're changes right in certain parts of it. What we could be actually doing is, if this whole image, each and every pixel changes, then we can actually look for a much drastic change. And those kinds of drastic motion change could actually be indicative of something special about the video also.

07 - Feature Detection and Matching.srt

The concept of Feature Detection that we'd looked at, in case of images, still applies here. Here I'm showing a video, and you can see all of the green dots. And these are features that are being tracked. And, , there is a red box in the middle that actually is trying to remain stable. We'll cover this a little bit more in detail when we actually talk about video stabilization. But you can notice that, in essence, feature detection is going on here as it was for images, except that we're doing this over time. So, the whole concept remains the same apply to images. But we're interest in here is in leveraging the fact that features found in one frame may also be visible in the next. So as long as there's not a drastic change, that is each and every pixel has not changed from one picture to the other, If there is some sort of commonality in frames between one to the other, we can use that fact to help us track features and observe similar types of things from one frame to the other. So I could learn for example that in this frame there was a building here with these types of colors. As it moves around there might be a bunch of pixels coming up in the next few frames that may have the same kind of color and appearance.

08 - Feature Tracking.srt

We can also use this kind of stuff to do tracking of features. There are many types of approaches to do this, I'm going to just summarize two of them. One is direct approaches to tracking, where again, what I can do is find a specific feature, something that I'm interested in, or interest point could be a corner, and match it to a feature in the next frame. So that would be much more point tracking, finding a point, or interest point, and seeing where it is in the next frame. Motion based approaches are also widely used, where we compute motion at each and every pixel. We do not find an interest point like feature track detection had been doing, which we applied for variety of other projects but here , it says I'm going to look at each and every pixel, and find that pixel again In the next frame. This is a method called OPTICAL flow and is also widely used in computer vision types of methods. Those of you who are taking the computer version class or will take it sometime, you'll learn a lot more about OPTICAL flow in that class.

09 - Tracking, Registration in Video.srt

Here I'm showing an interesting video. This is actually a video from the Georgia Tech football game. Me and my group did a little bit a work on analyzing football plays. And here you notice that we can register an image. And actually, again, on top you see a play field, and you can track the players, these are x's and o's. And we can track the players and also keep the field registered to know exactly where the yard markers are. This is all done by using computer vision methods of tracking information in these videos. Tracking the players, tracking the ground and using the yard marker lines to align the field as much as possible. That allows us to then, , extract more information about where the players are and how they're moving. Yes, in my group we do a lot of work on video and this is one of the examples.

10 - Registration and Blending in Video.srt

Here's another example, which talks about the registration and blending effects that we have talked about with panoramas, except, now applied to video. Again, something done in my group, just showing you an example. Here, you see us warping around, and moving the different view points. There are five players still moving around, and , it comes from four different views. All of them have some overlap, we can register them together, and as we move around different view points, they get warped and morphed. , sometimes the view changes, you can see some things kind of vanish, but they appear from a different view point. So, in essence, in this instance what we've done is , interpolated and generated novel views by being able to look for information on the playground, and apply that to be able to then model a virtual views. Again, something we looked at when we looked at things like morphing and panoramas, except, this time around applied to videos

11 - Summary.srt

So to quickly summarize in this lecture, I just wanted to get you started thinking about what is video and what are the representations, videos, building off the concept that we have studied extensively, that is images. The concept of persistence of vision plays an important role in this one for both capturing and then, , visualizing and playing videos back. We talked about how we can take the same concepts of filtering and processing images, everything we learned, except now we will do it in three dimensions, not just X and Y, but X, Y, and T. And then we talked about, we can actually apply some of the point the detection feature, detection methods, and use that to track interesting information across videos. same kinds of techniques are used to do people tracking and stuff like that. Again, we won't get into this kind of stuff in this class, but for those of you who are taking the computer vision class some time, will learn a lot more about it. Please find additional information both in the Rick Szeliski book, but also just look at my website. There's a lot of fun stuff about on videos. My group, as I said, works extensively in the area of videos, and we have loads of examples and I'm going to be showing you a whole lot of them in this class.

# 06-02 Video Textures.txt

01 - Introduction.srt

In this lesson, I want to introduce you to the concept of video textures. Video textures takes the concept of what we have learned in images, that is, looking for textures in the spatial domain, something repeating over an image. For example, like a, a pattern of some sort could be grass or could be water. Now, in this instance, we're going to take this concept and apply it to the domain of video. How does one process and analyze a sequence of images that have flowing water, moving grass, and other types of repeating phenomenon, natural phenomenon like this? And you learn actually how to synthesize novel versions of this by just doing a simple bit of analysis on a small bit of video. You're also going to have fun experimenting with this in one of the assignments.

02 - Lesson Objectives.srt

The objectives of this lesson are for us to learn about, what is a video texture, the concept of video texture itself. Talk about different methods to compute similarity between frames. We will talk about how we can use similar frames to find transitions to generate video textures. And then talk about additional things like how can we fade, cut and morph for video textures again something we have done for images in previous lectures. And finally, I'll talk about a variety of applications of video textures.

03 - Recall Video is Images over Time.srt

Recall, in previous lecture we talked about that in essence video is nothing else, but images that are stacked on top of each other in time. And we looked at , how we can access this by looking over x and y, and looking at , how we can stack these images over time like this. And that allows us to look at digital video in a similar manner, as we did for images. , looking for x and y stacked in time, and we can access any part of this image by looking at how we can access it in x and y, or i and j for discreet ones in time. We also discussed the concept of video resolution extending from concept of image resolution, and looked at aspect ratios of videos, and talked about where I did a video formats. Again, this is just to kind of remind you, what we've talked about before in terms of image and video representation, and similarities and differences in each.

04 - Video Textures.srt

So what is a video texture? Let's look at a few instances to convince us of what the concept of video texture and videos itself provide. So, if we look at static images like this. Here you can see that there are static images of four relatively dynamic scene. A flag here, most probably is fluttering and this is just one snapshot of it. Right? Now you can see the flag, I mean, you can pretty much perceive that it must be moving and here is a candle. , we know candles are always flickering. So this is a static version of this. Looking at these two children on a swing set, you pretty much again know that this must be a dynamic scene, where the children are swinging and we have captured them on snapshot and another example of these balloons. And , if you, you don't have to imagine much, but you know that when it might be fluttering these types of things. Each and everyone of them, there is some sort of motion going on. And still pictures, are capturing it, but that's the extent of what they do. That is they just capture one moment of it. Again, photographic value was they're valuable, but they don't actually show the dynamics of the scene. Video on the other hand can actually show the dynamics, right? Now, I can see everything in terms of what the flickering is going on for the flame the flame for the flag, the balloons and the children's swings. But , we notice one thing. It has a very well defined beginning and end. As soon as the motion has ended, the video came to a stop. I want to be able to kind of see more of it. Well, and I want to be able to film more time to it. So here is what we can do it by creating a video loop. So looping video takes this video, which ends after a certain time here and loops it again. So , what it does, is it actually stops at that time and goes to the beginning and starts another video. Now, , there's a lot of luck involved here, because what may happen is you see a sudden jump when the looping begins from begin to end. So for example here, boom, everything shifted by a lot, because it went back to the beginning. So, , that means we can actually loop it, but it does kind of have an artifact of a sudden jump. Video textures avoid all of this. See here, all of these videos continuously play. And in fact, they can go on infinitely long. There is no sudden jump, there is no end to this video. what we've done is created an infinite loop of videos that continue to play forever and ever. And that's what video textures is all about.

05 - Video Clip to Video Textures - I.srt

How do we actually do this? Well, let's talk about this. Here is a simple example, I have a video clip here and I'm going to use this to generate a video texture. Playing this video you kind of notice that it is got a lot of dynamic nature, and , I'm intentionally, taking simpler examples of these flames. But you can see that this is actually a very interesting, you know, dynamic scene. A video texture builds from this and it identifies transitions in time which it loops over. So in essence the red bar here is showing you something interesting. This was the time axis from zero, to let's say whatever, however long the video frame was. It jumps from one to the other, and it can keep doing this in this space to generate a Infinitely long video. So this red dark, in the time axis, is flickering from one to the other, and it's showing that it can actually search and seek out other examples and keep looping. How do we generate that is what we're going to try to get next

06 - Video Clip to Video Textures - II.srt

Let's look at the same video clip. This video clip has ninety frames, captured 130th of seconds, so this is a, you know, a three second video. And, you know, this will play on for three seconds and stop as it did. So let's try to visualize what these 90 frames look like. Here's my x and y, and by now you know I have a time axis. Here are each and every one of these frames. I've stacked them. I've just offset them a little bit, but you can see the animation as it unfolds, right? That's my 90 frames of video. And I label them just f1 to f90. This is my first frame. So, how about I do something interesting? What I do is I compute how similar frame 1, f1, is to all of the other frames, including f1 itself. , it's the most similar graph font. It's the same thing, right. So if I just compare f1 to f1, the most similar is f1. So in this video volume, there should be some other frames that might be similar also. Again, I know I've kept the example simple, all black pixels, that's where the, for our flame is itself. And I can now actually figure out from them which are the most similar ones and the most dissimilar ones. So in essence what I now need is some sort of a distance metric. We want to be able to do this for all of them. I want do this from f1 for all of the other 90 frames. And then I want to do this for f2 and all of the other 90 frames. And I want to do this for f3 and I want to do this for all 90 of them. So I want to find out, , a kind of met, you know, matrix which compares f1 to all of the other frames. F1 to 90, f2 to all of the other frames, including f2 itself, f3 to all of the other 90. And do that across the board for each and every image itself.

07 - Similarity Metric - I.srt

So now, , that begs the question. How do I define similarity? So here I'm just showing you a few frames, f1, f2, f3, 4 and so forth. And, as we know we have 90 of them. First, I can think about computing the Euclidean distance between two frames. So here , let's consider just two frames. And we can try to do this mathematically very simply. That first frame here is p, and it has all of these pixels up to n pixels, and q all of these pixels up to n pixels. Again, they have to be the same size, as we know, right? So , now we want to know how to compute the Euclidean distance here, and I'm referring, that is d2 from p to q, which , kind of, now takes the square of difference from each and every one of the two similar pixels, p1 minus q1, p2 minus q2, for all of them. This, , can be made into a simple equation, which is shown here. Summation across from 1 to N, all of the pixels, and subtracting one element by the other, and , we're taking the square difference, which is the Euclidean difference here. This distance metric is referred to as an L2 norm. And, , this is a way of kind of computing the distance between each and every one of the frames. I can do this again, as I said, for each and every f1 to the other 90, f2, the other 90, and all that stuff.

08 - Similarity Metric - II.srt

Another similarity metric that we can play around with would be compute the Manhattan Distance between two frames. So, here again, we look at the two different frames, p and q, how each one of them has an elements. And here the difference is taking the difference from each, and every pixel value intensity one by one, just adding them together will give you a very large number. Again, if they're completely different, very small number, , that they're not very different for each and every one of them. And so, a simpler equation that does it, if it a summation would be this, and , this metric is referred to as L1 norm. And just to note the bars here to kind of give it an absolute value of this because we don't want to get negatives, and stuff like that, we just want to get the positive value out of it, that's another metric. Actually, in practice, both of them work quite well.

09 - Finding Similar Frames - I.srt

So how do we find similar frames? Again, what we are trying to compare each and every frame, frame i versus frame j, we can do now is create a simple matrix. And here is my output. This is an interesting way of looking at things, right? In this case, assume that if it's a black value, it's the most similar. That is, they're pretty much the same. If I did a Manhattan distance between two pixels, two frames that are the same, so if I just did say that's our difference between f1 and f1 itself, what would I get? Well, I'd get distance zero. In this case, zero would mean a black pixel in this image. So , this kind of starts showing an interesting structure. This is for the candle example we've looked at. And now we start looking at it, you kind of notice that yes, the most similar one is the image itself. That's why the diagonal is black. But once in a while, you see it sometimes off diagonal and off course. You also see other frames that are similar that are actually somewhere far off. So, for example, here might be a frame. This might be f50, but here it's f100. That is also somewhat similar so now I can actually look for these types of things. So, we can compute the Euclidean distance between all of them for all the N frames. And now, , this is what our major axes looks like. These were the most similar. So , one way of transitioning in the traditional way in video is just going this way, right? We go from the beginning to the end, and we play all the frames. In this matrix here, black are similar frames, because again, the distance is almost zero, while and white are dissimilar frames. Similar frames are the ones that would be the best to jump to. So now I can go here, and rather than go all the way there, I can jump to another one. Black are the most similar frames, so I can move to this one. And then I can jump back this way. And that's the looping we come up with. So I've already just shown you one simple loop. You can imagine we can do loads of loops like this. I can go up here and jump back and go there. Go up here, jump here, and come around and keep on looping. And that was the yellow dot you saw at the bottom of the candles that I was showing you.

10 - Infinitely Long Video Texture.srt

Let me show this to you again. This is exactly the same problem. And this time, as I will traverse the diagonal, I will find something, jump off diagonal, come back to the diagonal, jump off diagonal, come back to the diagonal. So, imagine this to be my diagonal, and I'm jumping off, and creating loops. There's one, , I can do jumps forwards and backwards. And now, you see very simply that we can take a video volume like this, and traverse it in many different ways. In essence, what we've done is extended the time axis. You're not bound to just the axis of this we moved around a lot. So, use exactly what I'm going to show you here. This is my diagonal, and in this way we're going to traverse this axis, and whenever off, you know, we find something off diagonal we'll jump out of this way, and keep on looping this way. So, anytime I get these arcs show possible pathways. For example, here you notice these jumps, and whenever the arc lights up , it's going off diagonal, and coming back to the diagonal. So, now we've done what we've set out to do, we created infinitely long video texture.

11 - Finding Similar Frames - II.srt

Again, because this is my traditional way of going through video. Now we can play around and by having it do other things in the path, going there, jumping off diagonal and coming back on the diagonal. Right? Now, let's actually mark these as four of the interesting points in this space. So, these are my four different you know, one, two, three, four, right? And, , what I'm doing is I'm transitioning from one to two. In this case, I'm showing different paths through it, and then I can actually just move from two to three, three to four, and I also have the option of looping back. I can loop back from four to three, four to one, four to two. So this starts giving us a way of representing this, in what is known as a Markov chain. , at one stage, I'm looking as to how far I'm going to go to the next stage, and keeping that memory. So this allows us now represent how we're going to traverse this. This representation gives us a lot of power. I recommend you to look at the paper that we discussed. A variety of extensions of this. I'm going to talk about a few of them in a few minutes. But I wanted to kind of also look at the paper and see what kind of different things we can do with this representation. So here's an example of some of the ways we can actually do this. So here, for example, you notice that just the same representation, and depending on how we do the transitions, we can actually make the same candle frame have different looks and feels. So one, you notice, is jumping around a lot more. The other one, this one here on the right, is jumping around a lot less. Same data. We can actually, just depending on how we do the transitions, create a different impact.

12 - Preserving Dynamics with Transitions.srt

Couple of things to point out here now, this is something that you may have started thinking about. So, imagine there is a pendulum, which is swinging. , in the case of a pendulum, there are two place it might be the most similar, right? When it's coming down this way, if I'm just doing an image, it's similar here, and , it's coming down, but it might be also similar when it's on this side. So, what will happen in this instance is something like this. So, if you notice it started finding similarities before it's come all the way down, and , it's kind of jittering because of it. Again, simply put what happens in a pendulum is, that it's similar at this point, both when it's coming down and up, so it might just do this, rather than go all the way down. Well, that allows, kind of creates a problem, and the way we fix it is by modeling a little bit more of the dynamics. So, the pendulum here is coming this way, there are two different options. I want to be able to kind of make sure that I kind of keep the motion trajectories in check. And that allows us to then create, even an infinitely long pendulum motion like this. So, just keeping in mind, and doing a little bit of look ahead, modeling the dynamics allows us to do this, and the Markov chains representation supports this.

13 - Fading and Blending in Video.srt

Now we've covered a lot of work on fading and blending in images. Here's an example of how we can apply this to video. Here I'm going to show a cut and then I'm going to show where it fades. And then , we're going to do morphing and feathering to kind of clean it up. So you notice this again, I'm just looping over this. But if you do a cut, you kind of see a much sudden jerk. In case of fade, you should be seeing a little bit of ghosting artifacts between the frames. And then , when you do morphing and these types of things with cuts, you get a much better. So, in essence here, we kind of start modeling all of the different types of things with morphing and warping kinds of stuff with a little bit of feathering. So this is the morph here and you can see much cleaner transitions between all of them. Again something we can actually to much better as we know more about the images and how we can actually do all the processing in-between all of them. Again, please see the paper for to here for more details on how we can do all of this. Couple of other things to notice here is the original video of, , it stopped. Here I can actually make the original video be infinitely long. Here we just do a single fade. , there's a little bit of blurriness here. , we do multiple fades, which kind of merges it. And here the water gets a little bit more blurrier. This one is less blurry, because we're just doing very single fades. There's another bit of work that I actually refer to in this paper, in this presentation that actually improves and makes this completely crisp, which is called the Graph, Textures Method.

14 - Not Just Fade or Blend, but Cut.srt

Let me now showcase a simple example of how we can actually applied what we've learned about with images, that is sometimes it's not better to fade or blend, but to cut images except, now this will do this cutting in time. Let me show you this video here, if you notice that we have two videos of you know, waterfalls, but rather than actually fade, and you know, cut or fade it. We finding a surface between both of them, and using that to figure out which pixels to show. Let me actually, now explain this here, so what's happening here is we have an input video and we have an output video, we put both of them on top of each other, and find the best possible seam in time. And use that to find the best pixels between them, and use that to now generate a video, that also, is crossing over the two video volumes, but allowing us to create a much more crisper version, as we did with cuts. Here you see this example in this case of this simple video of, of blowing, when, you know, blown grass. This is my original video sequence, can notice, , when it loops, and you see a jerk, this was actually the earlier method. , when we do it with that, there's a little bit of blending going on. And the new method, which is a graph cut method which is also for me and my students, , can generate this much, much better, and much more cleanly. Same thing again, for the waterfall example that we looked at, original video sequence. Much crisper, with graph cut if you notice, we can actually generate this. Really crisper, and , this can go on infinitely long. So, that's what actually you come in, in the case of using cuts because what they again do is they provide you with the right pixels, and not blending between pixels. Again, something we looked at when we talked about images.

15 - Video Portraits.srt

Here's another interesting example I'm going to show you. This reminds me, and I hope it reminds you folks, of the kinds of stuff we see in Harry Potter movies. By the way, this work was done in 2000, so before any of the Harry Potter movies or books. And the idea was we had was let's actually put a camera on a person and create a video portrait, where they sit and make a few expressions and then we actually now create a portrait that actually has them alive, that is moving around in infinitely long doing this kind of stuff. That's what we did in this case is we video recorded a person for a few seconds and used that to generate an infinitely long live video portrait. Here is, , you can see the red bar will jump and it's kind of showing it's going from one frame to the other in showcasing a little bit more of the alive scene. Again, to me this reminds me of the kinds of stuff of video portraits we see in Harry Potter movies. In this instance we actually go one step further and actually we use two cameras. Remember all of the stereo stuff we looked at, and allow it also to kind of give a little bit of depth to the scene. So now the video texture is actually going on when we do have two different cameras and we're moving them around and kind of going from one frame to the other. This is referred to as view morphing and allows us to kind of give it a much more alive video portrait. By the way, we will be doing an assignment on using video textures and you can actually try to do this for yourselves also.

16 - Video Sprites.srt

Now some more advanced examples. This is a bit of work we also did. Here, , we videotaped a hamster and then, , we made the hamster walk a line. So we gave it a trajectory to follow. In essence, what we did was we actually did this analysis, not over the whole frame but just the region where the hamster is. So extracted what we refer to as a video sprite. We took a background picture and made the video sprite find the best possible path. So no longer are we actually just trying to figure out the whole video volume and finding the best possible frames and transitions. We can do this for a small region, the pixels that just where the hamster is and, , using that we can make it now, figure out a best possible transition set from the data that it has to be able to walk this path. Here's how we do it. We actually analyze the hamster in a green screen like this, so we can remove the hamster out of the background whenever they're visible. So here, , when it's the green screen is what we want and we can separate that layer out. The kind of stuff you know you can do in Photoshop. We did this for video and, , we get all the rich motions including the shadows and stuff like that. There's a little bit of machine learning involved in this work to kind of differentiate from the front and the back of the hamster and that allows us to model all of the phenomenon of different types of things and then, that's used to generate this segment that you just saw. So here's a final result by just compositing all of them but also doing the resyncing to locate the hamster back, and I think it looks kind of interesting. If you look at my website, you'll find many examples of both this and some additional work we've done in this space.

17 - Cliplets, Cinemagraphs.srt

, these days this whole concept of video textures is coming back and it's now coming back as referred to as cinemagrams. So here I just show two examples. This is using the Microsoft Research Cliplets software. And here you notice, it's a static scene except there is only one thing moving, and that's the train. All the people are still. This gives a very nice rendition of kind of showing motion and emphasizing only one bit of motion. You might want to try out the Microsoft Research Cliplets software, interested in developing these types of things have provided a nice toolkit for doing this. There is a whole lot of additional stuff also coming up, for example. There's another example, where if you notice the whole scene is static. The only thing that's moving is the water from the, from the tap here. Again, showing you the emphasis of just one specific dynamic motion and keeping the rest stable. We'll talk a little about these kind of things also in one of my other lectures. We're going to talk about video stabilization, but again, there are lots of techniques out there and you can play around with this.

18 - Summary.srt

So, finally to summarize, we introduced the concept of video textures in this lecture, talked about various methods of how we can compute similarity. We talked about how similar frames can be used to find similar points. And, and we can actually use that to jump, arc back and forward and that would allow us to kind of generate a much more infinitely looping video. We talked about how blending, fading and cuts and all that kind of stuff can also be used in the domain or video, then we talked about variety of extensions of video textures. Listing here a lotta papers have come in, including the three papers from us that we worked on, but there are additional papers that you might actually find interesting. Panoramic video textures is another interesting one. And actually these days as we talked about it, there's much more work going on in trying to create cinemagraphs. I'll show examples of that in one of my future lectures too. Again, look at my website and as usual, search around, you'll find lots of beautiful examples of this kind of stuff on the web.

# 06-03 Video Stabilization.txt

01 - Introduction.srt

As we are talking about video, I'm sure you have experienced taking videos with handheld cameras like this. Casual videos taken with smaller cameras like this are prone to a lot of shake and jitter. We're going to talk about what's happening when you actually have that kind of shake and jitter. And more importantly, I'm going to introduce to a method that we actually here have developed at Georgia Tech ourselves that removes the shake and jitter as a post process.

02 - Lesson Objectives.srt

Things I will cover , I'll introduce to you the concept of video stabilization. We'll learn about how we can estimate the camera emotion just by analyzing the video given to us. We'll talk about how we can smooth the camera paths to be able to then synthesize a new camera view that actually is much more smooth, stable. And how we can render those stabilized videos. In addition to that, I will actually introduce to you the problems associated with something that is called the rolling shutter artifact. That actually is predominantly visible in lower-end CMOS sensors. Again, remember, we've talked about CMOS sensors. We'll talk about them again a little bit here, and how we can actually get rid of those in the case of video, because again, this will actually increase the amount of shaking during a video because of the rolling shutter.

03 - Stabilized Video Example.srt

So let me demonstrate both the need for video stabilization, how we're going to go about it, by showing you this example video, which I got from the Internet. You can see this video is actually from a GoPro-style camera, of a person wearing it on their head. And going for, you know, a marathon in this instance. Very hard to see this and in fact if you look at it carefully you might start feeling sick a little bit. Almost feels like an earthquake kind of motion. Almost impossible to see any details. And this is the quality of video you get from the internet. Lot of you know, noise in this video. this is the kind of stuff we get and, , we would like to improve the quality of this video. Here is what we want out of a video like that. We would like to take that video and generate a video like this, still a little shaky, but much smoother and actually much more viewable. So what I'm going to actually talk about in this lecture is actually a system that we have actually built ourselves. And actually is running on YouTube. And I'm going to actually step through variety of steps that we went through to build this whole system. That will take the video like the one we saw and generate a video like this one. Just to help us see this, let's put them next to each other. Original video, right next to it is the stabilized video. You can tell much more shaky, much less shaky. it's important to know that these kinds of videos are getting very popular. And YouTube has a whole lot of these types of videos, partially because it's just much more of the domain of where we are with videography these days.

04 - Video Stabilization.srt

So in this lecture, I'm going to go in to details of a video stabilization system that's actually in wide use currently. This is a system that has been built at Google in, in cooperation with myself and a bunch of my own students who are now working at Google. Here's just example again of me taking a video of my son giving a speech. This was using a camera, but , maybe a variety of reasons. Again, I did not have a tripod, holding it with my hands, maybe I was nervous. I was able to not keep the camera stable. This is the output from the system, and in fact I'll show you how, I'll show you how we get to something like this. This is actually a research system that we have been working on for a while. This was a Was a PhD thesis project that included my PhD student Matthias Grundmann, another PhD student Vivek Kwatra, myself, we published two papers on it. Matthias and Vivek are both working at Google. In another paper, we also had Matthias, Vivek, and Daniel Castro are working with me on a paper to the International Conference on Computational Photography. So here just to show you, this is Matthias, Vivek and Daniel.

05 - YouTube Enhancement Suite.srt

Now let me so you what this system looks like on YouTube. This is one rendition of it. When you go to YouTube, you can upload a video. And after you upload a video, it's now part of the enhancement suite. You can actually see the videos and you can actually apply a variety of different fixes to the videos including change the colors. But the one that I want actually mostly emphasize is the one stabilize. Click that button. And a few seconds later, you actually got a result. This is the original video. Shaky. This is the final one much more stabilized. And , since it's running on YouTube. Thanks for the amazing efforts of all of the YouTube engineers who have done an amazing job of building this whole UI. In real time, you can actually now see the impact of what your algorithm just did. I'll show you what active different examples of this in a bit. But what you can do is after you've uploaded your video, you can actually stabilize it and actually save the MP4 file. And after it, actually does the whole full in the scales, HD resolution stabilization on your video, you can download it when its done. Additional feature in this thing is that when you upload a video, it pops up and says, oh, we think your video is shaky, do you want to improve the quality of it? Completely automatic, as I said, this is for purely casual users. And once you press this button, again it pops up, the improve quality next to your self, next to the original one and then you can visualize it. And then , after if you like it, you can save the changes. Encourage anybody who wants to play around with this to upload your own videos on YouTube and try it out. The interface has evolved a little bit from this version. But , it has the same software that I'm talking about.

06 - Steadicam.srt

Now before we go on, let's actually talk a little bit about video stabilization. Video stabilization is an extremely important field and anybody who's been doing work in video knows that what we want to do is try to keep as stable of number of shots as possible. This has actually resulted in variety of innovations. The most well known and most established is the steady cam invented about 1975 by you know, cameraman Garrett Brown. He actually did some of this work when the early Star Wars movies were being filmed and in fact some of the shots that were taken in the, in the forest with the Ewoks and stuff. If you notice lot of motions that were going on were all relatively stabled, captured by a camera like this. I've had the privilege of meeting Garrett Brown, he's an amazing person. >> This is a steady cam. Mine is a Master Series Elite. The steady cam was invented back in the 70s, and it's revolutionized film and video, by freeing the camera from the jib, and the tripod and the dolly without the bouncing and shaking of hand held. Many people think it works by gyroscopes but in reality it's just balance and the physics of the steady cam's arms to isolate the body movement from the camera. While it makes the camera seem weightless the rig is anything but, and the operator needs a strong back as well as a creative mind. >> So I just showed you a small video from YouTube about what this whole steady cam is all about. And if you notice one the interesting things is it's got a counterweight and it's the balance of this counterweight on this you know, counter lever from the camera that keeps it stable and removes it from the motion of the user. Widely used camera system. , for those of you that are interested you can look around on the web and you'll find even DIY versions. Where you can actually make your own setups to do something. Not as well as this, because this is a professional one, but similar types of things yourselves. Again, just like any concept in photography and videography there is a huge DIY culture on these kinds of things. I encourage you to look at that and there are other solutions too.

07 - Recall Evolution of the Camera.srt

Now remember, one of the things that we've talked about in this class, is how the camera has evolved. Our relationship with the camera has gone from this kind of a tripod setup, to much more distant, to even this kind of casual relationship. And that's what's important and especially true now that we have cameras like this or even Go-Pros, which is like the, one I showed you earlier. And you know variable cameras and stuff showing up. So more and more of these devices are coming up. So now we need to start looking at other ways of doing stabilization without actually having to carry a profession rig like the one we just saw.

08 - Video Stabilization Types.srt

So, what are the other types of video stabilizations available? Now, , there has been a whole lot of work of doing optical and in-camera stabilization. , in this kind of stuff, people have attempted to stabilize the lens it's self or also adaptively change the sensor as, for example, motion happens on a camera. And for example here, just showing you two different pictures of how we can actually stabilize the optics itself or the sensor to the motion. And these kind of are being embedded into the cameras itself. you see on high end SLRs and stuff, they have these types of features. Sometimes, it can also use the accelerometer and a gyro, which actually is becoming even more and more popular, the newer hand held cameras. Where actually you can use the fact that there is a sensor built into the camera which tells you how fast the camera is moving and you can use that as a priori information, a priori information to remove some of the shutter. But remember, one of the biggest things with all of this is that they're actually there to get rid of the high frequency percolations. The shake, because I'm moving the camera too fast would be the ones that I'd do it right there. But , also means that they're looking at a smaller temporal buffer. In the case that we're talking about, we'd like to also consider post-process stabilization, that is that removes the low frequency perturbation, large buffer. So for example, in the case of the jogger that we saw in the marathon, the head motion was relatively slow and that's very kind of low frequency motions. And we'd like to get rid of that, that means we need to keep a bigger buffer. That can only be done as a post process. That is after all the video is done, uploaded or saved, you run the process on it afterwards. This can be done actively in the camera itself. Another part of it is that if you do the post-process stabilization, we can actually use a distributed computing back-end. Which means, we can actually spend a lot of compute energy. This is what I showed you with the example on YouTube. It goes to a cloud, a whole lot of processing is done. And actually, we get much more accurate results afterwards. The biggest advantage of the post-process stabilization, is can be applied to any camera. It doesn't care what, where the footage came from. And in fact, you can actually apply to legacy footage. Now, , stabilization is something we want to accomplish for the kinds of videos where we want to get rid of some of the errors. And whenever I talk about stabilization and many people will come and say to me, oh, and some artistic ways we actually want to add shake and jitter. I mean, movies like Blair Witch Project and Cloverfield, were ideally designed to actually show you all of the kinds of motions and jitters that the director wanted. We don't want to get rid of that kind of stabilization. We're particularly interested in stabilization of videos from casual cameras. So, our goal is to now focus on this part of the process here. We're going to talk about post process stabilization. But the kinds of techniques I'm going to talk about with knowledge of more information from the camera itself can also be applied to in-camera stabilization.

09 - Post-process Video Stabilization.srt

The three main steps in the post-process video stabilization that I'm going to talk about are one, I want to estimate the camera motion, second, I want to stabilize the camera path, and finally, I want to be able to re-synthesize a new path. Now, couple of other options we have to note. Most of the casual videos we will look at would be from relatively wide-angle cameras like this. And the other thing is, one more given is most of the time, if there's a videographer involved, they will do the best they can to keep one subject in the center. Those two assumptions help us doing the kind of stabilization we're going to look at next.

10 - Stable, Virtual Camera.srt

So what we're interested in is a stable, virtual camera, which means is we're interested in finding a viewpoint in the whole image itself, or the video itself, that is actually much more stable than the original one. So here, , now we're going to start seeing this red line. This red line is going to start showing you a virtual camera within the original one, and our goal is to identify from the data itself, what's the best possible red box here within the big one. Do remember I did say things like oh, most of the viewpoints that we're going to get from these cameras from wide angle, so this does reduce the scope of the viewpoint by a little bit. We'll talk about variety of aspects in a bit too. So here you see now What we're doing is tracking the subject and the camera and refining where this red box could be. in this instance you notice the red box is going out of the range and the domain, the aspect, of the image Itself. The challenges with this is if that happens, we really will actually start creating problems for ourselves. We really can't deviate too much from the original camera, because if we do like for example, when I see the red box move to this side or that side. We'll have to start filling in holes, otherwise we'll get black borders in the image. We can't actually go to content that we don't have. We have to be always finding the content that we have. So we want to be able to get this red box in a region so I don't have to synthesize other regions. What that means is if I actually just look at the red box as it is. And I started tracking it, this is the kind of output I would get. So as the red box moves, , you start seeing black borders. Yes, the subject is now more stable. But, in keeping that stabilization, you notice that I'm actually now going out of the scope of the image. And I have to then figure out how to fill up that image.

11 - Stabilization by Cropping.srt

So our purpose here is to now start seeing is, can we actually find this red box? Can we actually find a crop of the original image and just keep that red box inside the scope of this entire image? So here you see the red box move. And , it's a cropped version of the original. This one is the result, this is the original video. And what we want to do is find the best possible red box. But we want to keep this red box cropped so that I don't have to look for pixels outside. I don't want to look for those black borders, and I don't want to be able to fill in the holes. So our solution is constrain the crop window to stay within the frame bounds. And also, this allows us to guarantee that we never go into an undefined content. You don't go outside the borders, and therefore, we don't have to do things like in-painting, that is figuring out how to do the pixel and color them pixels around to be able to make sure that there is no black regions. Remember, this is the kind of stuff we looked at when we were doing filtering, where we had to flip images and start doing simple stuff. But, , in a video like this, we would have to do a lot more, and technique usually used for that is in-painting and actually does always create artifacts.

12 - Estimate Camera Motion.srt

So the main steps again are in video stabilization are estimating camera motion, stabilizing camera paths and then being able to resynthize them. Let's look at each one of them. So let's start off with first step estimating camera motion. Here we will actually talk a process that we've looked at before. To estimate camera motion, we need to find features and then we need to take those features and track them over time. So, again, what we can do is we can find image corners. Remember, what we did and we were talking about feature detection. Look at points that have a high gradient in both x and y, that's a corner. And using that corner with the fact that we want to do is we want to track hem over time. And here, you see a lot of these green lines. , these green lines are saying is okay, this point was here in this frame and moved to this one and that starts, if you're noticing, give me very good motion, even when I'm zooming in and out. You can see motions very clearly going on as to where the different parts of the region are changing with respect to how I'm moving. Or in this case, even zooming the camera in and out. , the camera's going through translation, some rotation and scale here. Couple of things we have to note when we do this, we have to start differentiating between background motion and foreground motion. So for example, in this case, if you'll notice there's a car coming in. We want to be able to differentiate between the motion of the background itself and some of the foreground elements. We don't want our camera to be dominated by the car moving. We want to be able to lock in do the background pixels itself, so we have to do a little bit of separation of foreground and background in this analysis. So we only, I want to estimate the motion of the background, because that's the one we want to stabilize on. And we want to be able to kind of create some sort of way to separate out the foreground by, with the background by giving more weight to the background.

13 - Motion Models.srt

To help us understand kind of how to do this analysis of camera motion, let's look at some of the motion models. Now this will remind you of the stuff that we looked at in one of the previous lectures about image transformations where we were trying to model how an image would be transformed from one to the other. Now, all of a sudden, this should start making sense in the context of how we can do this with video. So here, for example is a video that we are interested in. What we are interested now in is finding, in this image by these point moving around like this. What's the best possible camera motion in terms of a variety of degrees of freedom. And , we want to be able to model the least number of these degrees of freedom. Remember, the degrees of freedom when we talked about image transformation? Well, the image transformation is here is from one frame to the other in the video volume and we're now interested in finding out what those degrees of freedom are. Let's start off with , the simplest one, translation. So here, , we are looking for a translation where the frame is just moving from one print to the other in x and y. If we just did this, this is a two degree of freedom. And here, you see the example that I've just modeled the two degree of freedom. And if I was to show you the result of how the stabilized crop window would look like, that would be on the top here. And you see it, it has a little bit of wobble still. Which suggests that just by doing a two degree of freedom transformation between frames here is not going to result in a very stable video. You can see the jerkiness in the middle of the video here, right? , we have learned that when we can't actually model it with just the translation x and y, we can also start thinking of adding rotation and perhaps scale into the equation. So let's do that. , we can look at translation in x and y. But also let's add uniform scale, scale in both x and y and rotation. , we know very well that, that is a four degree of freedom, where now we have a theta x and y and the scale. It's a uniform scale, so it's the same in both. That suggest that now my crop window can rotate a little bit, but also can get bigger and smaller a little bit too. And that actually results in this solution here. Good. It's not very shaky, but still has a lot of little, you know, wobble in the middle of the image. You can see this part shaking around. , let's move to homography. We did all of this kind of stuff when we talked about image transformations again. In this case, we want to be able to translate the image from one frame to the other in an x and y rotating scale, but let's now add perspective and skew to the whole thing. That now means is now we have an eight degree of freedom model. We remember that from our early lectures. And now, I have a crop window that can also have a shear and perspective whole thing going on, in addition to the translation and rotation and this is the output. So, if you notice this looks much stable. In essence, we're what we have done is now gone through the steps of identifying the camera path by looking at these degrees of freedom for this rectangular region, the crop window. And if I can figure out those, those degrees of freedom are, I can actually start generating a new video. Just showing you the small region here, which is moving around in the larger viewpoint here. And remember, we are cropping the image, but , in this case, we're getting something nice in the output there.

14 - Similarity Model Over Time.srt

Let's look at this whole concept a bit more in detail with real camera paths and stuff like that. Let's first look at four degrees of freedom in the image itself. Here we're going to look at the same video here and let's look at translation just the changes in x. So this is the path that changes in x and I'm traversing it over here, and you saw happened. Also, we can do this for y. We can look for translations in y. Again, y is changing as I move from number of frames down this way. We can do the same for scale. This says the scale is getting you know, smaller and getting bigger. It was much bigger here, and then, finally, we can also look at things like rotation. So, this starts saying is that these are the four parameters that now we can model purely by looking at the whole region and going into a degree of freedom calculations, and being able to model the transformations of the image. This gives us the path that we are interested in.

15 - Smoothing Camera Paths.srt

Now that we have actually looked at how we can estimate the camera paths, let's talk about how we can go to figuring out how to stabilize this camera path. So, again, I'm just showing you these paths as before. What we're interested in now is approximately find a path that as stable is possible. Here we refer back to some of the simple cinematography principles. That is what are the best properties of a stable path of a camera? Well, if I had a tripod and this was my, you know, degree of freedom etc., of the path itself, what I assume is that green line here will keep the camera stable. Right? There would be no changes in the degrees of freedom in x and y here, so that would be my path. So, I need to find a smooth path like this. Then the second one would be I'm putting it on some sort of a railing and moving it around, so that would be a dolly or a pan. This, in this case of differential, it would be a linear segment. So, again, we have the red path is the original one. We now added constraints to it and found a path that has the yellows and the greens. And then , we need to start dealing with the ease in and outs, which would be this and this. And here, it would be this here. So these ease in and out with respect to all of this kind of start giving me a set of different constraints and different paths moving functions on the red line itself. That's what we want to do in figuring out how we want to smooth the path. So what we do is look at these different types of models and use that as a constraint on the path itself. we do this with various types of smoothing algorithms. Again, I refer you to the paper that allows us to do this much more efficiently. So now, let's look at what this means. , what we are interested in is finding a crop window within a frame, which is for example, this. And what that means is that there's a path. This whole window itself is the envelope in this path right here, right? So for example, I just made this small and this is the whole region that I can actually cover within it and as long as I keep myself within this region, I would be actually able to find the best possible crop window that traverses through this range. Let's look at that again. So here , this size kind of defines the window, the envelope in this thing.

16 - Path Smoothing Demo.srt

So now I'm going to show you a variety of conditions where actually would allow us to both increase the envelope and figure out the best possible smooth paths to the original red path here of a camera. So, for example, we can define various ways of defining how big we want the envelope to be. And, , in that condition's based on different types of constraints we can add, we can actually look at different types of paths. We can actually have constant length paths, linear paths, that actually now, if you notice, there's linear connections on the blue line itself. Parabolic ones, which start giving it more curves, much more smoother. And if you notice, these are now allowing us to control how we can actually figure out both the envelope. And this envelope defines, how big my crop window is going to be and how smooth the path would be. We actually did a lot of analysis in this work to figure out some best paths, and that's what we refer to as the YouTube paths. That's the one that's been widely used in the system right now.

17 - Crop and Re-Synthesize.srt

So now we know how to stabilize the camera paths. Let's talk about how we can re-synthesize some of them. So here again, we're trying to do stabilization by cropping. Here is my window. It's rotating, translating, and, , scaling within this. This is my output. , as we talked about, this crop is constrained to remain within the frame bounds. So , what we are doing is we're rendering the image cell that actually is red box. So, it's much simpler and we don't have to worry about doing any kind of interpolation. We don't have to worry about any kinds of filling in the gaps and stuff like that. , what we're trying to do is, we're applying a virtual crop and that results in a stable video. And you can see on top the result.

18 - YouTube Example.srt

Let's see a bunch of different examples. This was one of the first classic examples we looked at was ice skating. we wanted to find pure YouTube examples do this and these kinds of examples are hard because finding the features that actually have corners and stuff like that and ice are relatively difficult. But you'll see actually it works quite well in this example. So again, this is the output, the red box is showing you what the region we found that actually synthesized this out. Hopefully, you'll agree that this works quite well and you know, it works quite well on any of these types of videos and it can be either extended video sequences and it works quite well on most of them. Here you'll actually see even a little bit of zooming going in and out. Now , it did crop her leg off a little bit at the top. It doesn't know anything about the content. In fact, it's not doing any kind of tracking of a person in that video. It's just trying to figure out the best possible camera path. The advantage, , of doing this kind of stuff is now we have lots of videos that we've seen. Here's a video that also was from YouTube. Here you can actually see a couple of different things. Here you see a lot more shake. We'll talk about this shake in a bit. This is actually because of the fact that this actually is a camera that has rolling shutter. We'll talk about that in a bit. But this is the output. You'll hopefully agree, much better and much more viewable.

19 - Recall CCD vs CMOS Sensors.srt

So, I did mention something in the last slide about rolling shutter. Before we go there, remember that in a CCD versus CMOS, there's two different things we talked about when we looked at the camera inside of the camera in one of our earlier lectures. In a CCD, all of the photosites, which are these color chan, color sensors here. Give basic information and they're kind of then read into an amplifier after at the end. In CMOS, the complement the complementary metal oxide semiconductor, what they do is they actually take the photosites and they put the amplifier right at the photosites. For example, in this pipeline, what would happen is , all of the information would be read off and the amplification would happen after things are read off from the photosites. But in CMOS, what happens really is there's an amplifier in each and every photosite. So, it actually does a little bit more work to read out the information and it force all distributed at the location of the sensory cell. Now, , CMOS sensors are cheaper. They can also be better than CCD sensors in many different ways. But when ha, what happens is when you start putting these types of CMOS sensors on in smaller bodies and cellphones and stuff like that, they don't actually have a very good refresh rate as things are scanned. For example, this is a video from a simple, handheld cellphone camera and you'll notice an interesting thing. It seems like the whole scene is nonrigid. For example, if you look at it, this pillar here seems to be nonrigid as it's warping. And if you look at any of the lines, they don't remain straight anymore. You saw that a little bit in the other video I just showed you with the trucks and stuff like that. But this is much more easier to see now the fact that there is nonrigid deformation going on. , this is because of rolling shutter, I will talk about that in a bit. Let me actually show you how we can take the original video like this, with rolling shutter and now using techniques like the one I've talked about with more modifications, allow us to generate a video that looks much more stable. Again, original video, this was deliberately shaken, handheld camera. And if you don't believe it, that this is video, you can actually see the person going up on the bridge. This is completely stabilized, what we've done is model this motion and then removed it and we synthesized it.

20 - Types of Electronic Shutters.srt

So what's the issue here with rolling shutter? Let's look at the concept of electronic shutters. Here we're going to talk about a global shutter which is what's in a CCD sensor. Again, there are no photo sites with an amplifier reading things are there. Everything is first gotten onto the photo sites and then read off and CCD. That image is red at one instant in time. So, if I was to take this camera and move it this way, as I've shown by this arrow here. What would happen is you would get a full image every time I move it. So, I have three different images, because what's really happening is, it's really efficient in how it scans from the top down here as I move. And it captures everything in a global manner completely, so nothing is actually moved as I move the camera. In contrast to this, let's take what happens when I have a cellphone. Again, very thin and a very small sensor and a lens associated with it. What happens is an image is read one scan line at a time. So now, if I was to take this camera and move it this way. What's really happening is the first part got captured. By the time the next scan rate comes in, the camera is moved a little bit. Similarly, when I move to the third one, move, the camera is moved a little bit more. So, I keep going. If you notice the camera is moving, which means this straight line, which would've been down this way, is now curved or at least slanted. So this whole scene has been slanted a little bit, because I moved the camera this way while I was taking the video. So the global shutter in this case would have kept line straight, the rolling shutter has created the small angle that you can see from this line here from this one. And that is , I'm moving this in this direction. This gets much more complicated if I was to move this in that way, right? And that's why you saw the nonrigidity. because if I had gone this way and come back, this line would have curved this way. I'm just showing simple motion, that's why I'm seeing a slant. So that's a problem and partly it is again, as I said, as we go down in the scan in time, we get a refresh rate like this. The problem with this is we don't know exactly what it is because it's proprietary known for each and every camera by each and every camera manufacturer, we need to figure this out pretty much from the data itself.

21 - Global Shutter Model.srt

So here is an example of video that actually has been tested for these kinds of things. I'm just showing example, so this is the first at one of the earlier attempts at trying to do rolling shutter removal, so this was the original video they had. This is known as the helicopter data and, , you notice lots of non-rigid warps. So this is our original footage that we want to actually start playing around with, see a lot of non-rigid deformations everywhere. So when we applied the first system, the one found a crop window, and tried to do this as a global shutter we also could not actually stabilize it very well. We see a lot of non-rigid warping going on. So what we have to do is go back to drawing board and start thinking about how we can actually do this knowing a little bit more about how a rolling shutter works. Not going to go into a lot of detail right now. I'm just going to kind of say is what we did was we actually took the whole image and kind of dissected it into small regions. Almost the scan rates that are coming down as I move the camera, and applied the same kinds of processing of computing the homography and all of that for different regions and different slices itself and used that to figure out, what is the likelihood of this region moving here. Use that as an estimate, and then, , unwarp it, knowing more about how each and every region would move. So, the problem we had was we didn't know exactly what the readout dor each and every camera was. We kind of said is, well, let's look at multiple motion models. So I had a different motion model for each one of them and then blend them back using a mixture of Gaussians. So, in essence, we took two of them this way, another two and did a blend between those and then I did a blend between the other two. Remember the kinds of blend concepts we've looked at similar in this context and what we did was use a mixture of Gaussians to kind of take all of the homography information for each integral region and use that and unblend it or unmerge or unwarp the image to correct for all of the motion. So here is the original video again. Lot of non-rigid warping going on in the middle. This is the solution using our newer approach now, we actually look at the rolling shutter also. Now you may be noticing that the input thing is moving around. That's partly the artifact of our cropping algorithm.

22 - Rolling Shutter Wobble.srt

Let me show you a few examples again from YouTube. This is an original video again somebody on a bicycle. There was an explosion, and they're taking videos, and you can see a lot of non-rigid warp going on, and notice things like you know smoke here are actually not known for being able to have very good corners, and features that we can actually track. This is the output, still has a little bit of non-rigid warp, but not as much. You would agree that it is actually much better. Let's put them next to each other. Original. Stabilized, removed the rolling shutter. So now as I said, when you upload videos to YouTube thanks to working with the YouTube engineers, you know, Mattias and Vivek were full time members of research staff at Google, were able to now get this running on the YouTube site. When you upload a video it pops up saying hey, we think your video is shaky. What would you like to do? If you hit this button preview one, two, three seconds later you get a real time preview. You can interact with it, you can see the original, and the stabilized version next to each other. And now I guess, you get the message that using the approach we just did, we can actually stabilize video, and , the person can , at the end save their video, and use it for other purposes.

23 - Adaptive Shake Auto Crop.srt

Now a couple of other things that we can also talk about is, that window that I showed you, the crop window also gets bigger and smaller on its own. So here, we are showing you a very long video. Again, this is a very difficult case if you think hard about it. It's snow, not a lot of features. You see the red crop getting bigger, smaller and also, you know, for example sometimes. I mean, it does crop out some things. It doesn't know anything about where the skier is. We've actually added algorithm that can actually improve the quality of this by constraining it to where the person would be. There's motion in this one except camera motion, and after a while another skier with come down. And one of the other things we've added, there is ability to actually deal with very long video sequences. On of the more important advantages of her approach is that, if actually doesn't find anything stable it reverts back to the original video, so the all times we actually do get a result. So it does have a graceful failure mode. Now, I'm giving you very specific perception of the approach that is specific to this method here. I do encourage you to look at other methods out there, and in fact even professional software like Adobe Premier and After Effects now have stabilization tools, much more detailed, and with a lot more controls than ours does, because in this one you just click.

24 - Alternative Stabilizer.srt

Not just to, actually, for those of you looking at other ways of doing things, let me actually show you something well, relatively fun and interesting. Here's a video of a person who actually then figured out that if you, if you look at a chicken, chickens have an ability to keep their head stable. Here, if you notice, he's moving the chicken around, but the head is as stable as possible. Right? This is not made up video, this is original, if you have access to chickens, you can try to do this yourselves. But you'll notice the chickens have an ability to keep their heads really stable. this whole idea was made into a little bit more craziness by this person here, who then kind of claims to have made this mount for a chicken and used that to generate a stable camera or steadicam. Ok well, to be honest, this is a fake ad for a new LG camera, but it's kind of interesting to note that this, chickens do have a way of keeping their necks stable and in fact, more and more fun things have been done. This, I'm just showing ti to you, just to be funny.

25 - Summary.srt

So, in summary, I introduced the whole concept of video stabilization, with the caveat that , I introduced our own algorithm, a very specific, but very simple algorithm, to do video stabilization. I demoed a working system. This is a widely used system, used by millions, you should have access to it. I encourage you to upload videos to it and try it out, and tell me what you think. Discussed the whole concept of degrees of freedom that are used to do the motion modeling of a camera that can be used to create smooth paths. Discussed how can we actually take the rolling shutter and remove the rolling shutter artifacts from the video to actually improve our video stabilization for casual hand-held cameras. I showed a system that you can play around with yourself. I encourage you to look at the papers, the two papers of ours up there. I've also listed two other papers that actually talk more about rolling shutter. I'm going to put pointers to other additional papers on video stabilization with this lecture also. Again more information is available from there and from my website.

# 06-04 Panoramic Video Textures.txt

01 - Introduction.srt

So far we have actually looked at a variety of different topics, like for example, how to use a panorama to give you a better sense of space, actually have a view which is much larger field of view than usual images that you get. We've also looked at how to analyze videos, and one of the specific things we've looked at is video textures. In this lecture, we're going to combine both the concepts of video textures and panorama building to create what is referred to as panoramic video textures. We're going to look at how we can take a video of a scene, use that to generate a larger field of view of that scene, but also incorporate in it the dynamics of the scene, specifically the kind of dynamics that's captured through video textures.

02 - Lesson Objectives.srt

So the specific objectives of this lesson are, first we will review what video textures are and panoramas, very briefly. Then we'll talk about how we can combine these to create a panoramic video texture. We'll learn about how we can actually construct a panorama from video, again remembering that the in the video there might be dynamic motion. Some things might be changing. how are we going to use that part of the scene that is dynamically changing and construct a video texture from it.

03 - Panoramic Video Textures.srt

So what, really, is a panoramic video texture? As I stated, it's a combination of two things we have looked at before. Let's look at a specific example first. This is actually a work that was done by Agrawala et al and it was published in SIGGRAPH in 2005. So here you see this video, and in this video you see in the camera is being panned and, , you can see the panning shows that it is now a panorama but if you notice, in this panorama, it's not a static panorama. The flags were moving, and now, actually, when you see here, you also see the water is flowing. When you get to the boats, and you look at it carefully because of the wind, the boats are also kind of moving a little bit and if you look at it very carefully, the background, the traffic is moving. So yes, it is a wide angle view but each and every element that is a dynamic element in this wide angle view, is actually dynamic, it's no longer a static scene, those things are moving. So a panoramic video texture is a video that has been stitched into a single wide field of view. As you noticed in this case, I was moving the camera around, it's a wide field of view image, or a video but , the dynamic parts of it are now video textures and that's why they continuously play and can play on infinitely long. So now you actually have a view that's dynamically evolving and all the dynamic parts of it are moving but still, it's a much more wider field of view than a single video would have been.

04 - Recall Panoramas.srt

Just to help us start it, let's go back to some an image that we've seen many times in this class and the whole concept of panoramas. That is , given many images like this, we can generate a panorama by doing all the stuff that now you know how to do. Finding the features in images, stitching them together, blending images, cutting images, and getting a smooth transition between all of them spatially, and these seven pictures with limited field of view let me generate a much larger field of view, image of panorama. , if you notice in this case, everything is static.

05 - Recall Video Texture.srt

We learned again when we talked about video textures that we can actually find loops in a video and use that loops of similarity from frames to the one frame to the other to generate infinitely long videos. Also, when we looked at video textures, you may recall that we talked about the concept of looking at videos, specifically videos that have some repetition, like this candle flame, find similar features of similarity between frames from one to the other, use that now to generate an infinitely long video. In this case, , you see the transition arch, which say that I found a similar frame here and I'm going to jump from one to the other. That allows me to create an infinitely long video of something that actually has repetition. Again, if you remember the previous example, we talked about doing this for flags, water, and similar types of repetitious motion. So now we have two different ingredients. So that, in essence, is what panoramic video textures is all about. Let's hear from the authors of this paper as to how they actually kind of facilitated this work. >> Panoramic images are compelling because they are immersive. They communicate a sense of being there. Panoramas show a wide field of view, which lets us look around within a scene. However, panoramic images are not dynamic. So a panorama of this waterfall, for example, appears unnaturally frozen in time. A video, on the other hand, can show the motion of this waterfall but still isn't immersive. It isn't panoramic, and it jars the viewer each time it moves. Video textures, introduced by Shodalosolve solve the looping problem by creating a video that appears to play continuously forever. However, the result still isn't panoramic. In this paper, we introduce panoramic video textures. Panoramic video textures show dynamic imagery for the entire field of view, and can be seamlessly viewed for any length of time. Ideally, we would like to be able to create panoramic video textures without any special equipment beyond an ordinary video camera and a tripod. It would be nice if we could somehow just pan the camera slowly across a scene, and then create a panoramic video texture just from that. This introduces the key challenge. The input video, shown here at high speed, only captures the dynamics of a portion of the scene at any given time, while the output must show dynamic content at all times everywhere. We begin by registering the input video frames into one global coordinates system. Then, the user draws a single rough mask that separates static and dynamic regions. We then create a static panorama for the static regions. Finally, we create a panoramic video texture for each dynamic region, and composite them into the static panorama to create the final output. Quite nice, huh? So now, let's talk about different stages that they go through to build this.

06 - Video Pan to a Panorama.srt

But first, let's actually just have a small quiz to kind of make sure that you folks are following what's going on here. Again, notice this video is the one that was shown in the previous video too. It's a camera that's being panned from right to left. In this case, you can imagine it wasn't a tripod and mostly it's a simple pan. And it's slowly done. What we are interested in is using this to generate a panorama, but at the same time, not getting carried away by actually trying to do things like matching all of the moving parts. So here's a question for you. I have various ways of building panoramas, right? I can actually have a small part of a frame and I could then ask the question, how can I build a panorama? In this case, I have a video, which would mean I would take all of the video frames, and register and align them to generate a panorama. So that would mean is, I would take this video and as it's moving from right to left, I will take all of the frames from this one and generate a panorama like this. Is that a legitimate way of doing panorama building? Or is this the other method? And in fact, this is the method you may recall again, that we may have been playing around. That is, I take from all of the video frame. Just select six different frames. , I'm talking about six. It could be many more. remember all of the things we may recall again from panorama building. So, the question for you is, which of these two methods is legitimate method to generate panorama, also could the answer be, both?

07 - Video Pan to a Panorama.srt

Well, this is relatively simple for all of you. I'm sure you know the answer very well. The answer really is both. Because, you can generate panoramas this way. And you can generate panorama this way also. And in fact, these panoramas are referred to as push broom panoramas. And actually we're going with initial of generating panoramas. Before people came up with methods like using images. And you know, trying to do this kind of stuff with different images.

08 - Video Registration.srt

So now let's dive deeper and try to understand what we can do with videos and use them to generate a panorama. , this is the video, you've seen it before. And what we are interested in, if you remember, is when you talked about video representation, what really is happening is that the local x y coordinates of each image is stacked together in the time axis to generate, , a video volume. All right so this is what we refer to as a video volume. In this case I'm just showing you a few frames of the sequence here and stacked on top kind of saying that in time, this is a stack of images, this is my video volume. But what we are interested in really is kind of creating a global representation of this. So lets come out with a global coordinate frame capital XY, and rather than worry about time, lets try to kind of start seeing what we can do with these images. So, that means is if I can take these images and move them around a little bit. So you know, for example, this would be an image that would be here, other images from the back would be realigned in different regions here. Some of them might be the same images because, as you notice sometimes, the motion might be slow. And , getting all of these images together in this format and then doing what we actually know how to do, that is registering images. Remember the stabilization algorithm could also be used for doing this kind of registration or even the kind of simple methods that we've looked at that is finding features and doing ransack and all of that kind of stuff to align those features together to create, , a wide field of view image like this. Using this, we can generate a panorama, which is shown here, but more importantly what we also need to do is identify regions that actually are the ones that are dynamic. That after I build this panorama, those pixels are moving. For example in this case the pixels will be moving where all the water flow is. In the case of this actual approach by they actually suggested that you can manually paint out the regions where the dynamic motion was. I leave it up to you to start thinking about various methods we can come up with to actually do this automatically, in fact if you think about the kind of stuff we did with video stabilization, where we actually look for background and foreground, some of those types of approaches could also be used to kind of generate this automatically. That is finding which pixels are moving after I will build the panorama, and also asking questions about, okay now I have a region where just the motions are. We would now actually use those to generate, , moving parts.

09 - Video Textures of Dynamic Regions.srt

So now how do we generate the video textures of the dynamic regions? Again, those pixels that were just the dynamic parts of the image are the ones now we're going to figure out how to convert them into video textures. So just to kind of start thinking about it, I'm now laying out this whole thing and I'm kind of saying is, this is the y axis, and this is x and this is time. So now we're going to stack each and every of these images in time and just show the X and t. So as time evolves, , we're getting more and more of these frames. So in this case, I'm kind of saying is, this is one frame, second frame, third frame and all of them kind of create this video volume, right? Let's think about it a little differently. And that suggests is let's actually only concentrate on the dynamic regions. So here, this part of the frame, not the whole frame, this is the only part of the frame where there is some dynamic. And again as I go down in time, there other frames come in. Remember again, this is similar to the whole part that we looked at before which said that if I had an image like this, there's only a part of this image where the dynamics is. So, in essence that means if this is my X, I'm going to be only looking for small parts in that image, not the whole part. And in this case I'm kind of suggesting for each and every image in the region. Only a part of it is the one that is where the dynamics is, and as I move further into time, this part may have moved here. And that's where I have other frames. So, in essence, this is the only part of the dynamics that I actually want to model. So how do we use that information? Well, for example, what we can now start doing is build video textures just for those small regions. So in essence what that means, this is where the video texture is, I don't actually have to do it for the whole part of it. I can find a sub part of this region and use that to generate a video texture. So the video volume in this instance would simply be in time, in, in the local coordinate X, simply a part like this. So this would be my small video region, I have well defined slits here. So this allows me to create now a smoother transition from each and every one of the frames and generate a nicer, smoother video volume to represent this region here. So, in this case what it says we're going to map the continuous diagonal slice of the input video volume to the output panorama. That's what we actually will do, in outputting the video textures onto panorama, again in that region where the dynamics are. , in this case we will restrict the boundaries to each and every frame. So each and every frame is kind of looking at just that frame and you're going to find a match the next frame. And the problem with this is, while it works fine, actually creates some sort of shearing motions across time. Now remember we actually had these issues when we talked about video textures too, and we said, match one frame to the other and , either you'll get a little bit of blurriness, because it has not the best match. Or you might actually see unnecessary motions that are impacted because how things are matching from one frame to the other. What does that mean? Well, let's look at that example. This is my original video frame or sequence, just from the part from the water flow is happening and approximately the same region, where we are now doing a continuous diagonal slices. So here's my original video, you kind of see nicer motion, and here you see a little bit of you know, shearing going on. Motion is not the most cleanest, and in fact, its the first frame, and it actually generates it, because it's not actually doing a very good job ungenerating the video volume, it's actually keeping the boundaries stable.

10 - Not Just Fade or Blend, but Cut.srt

To address this Agarwala et al kind of went back to an approach that I'd also talked about before, which is what we're interested in doing is we want to be able to find cuts not just blending from one to the other. So here example, is something I'll show you here. This is something again from an effort by Quatra et al, where we rather than actually just frame by frame do boundary matches. We find a surface between the two of them that actually lets you find a better match. In essence of that means is I have an input video and output video, I find a patch like this. And this again, we had seen before was when we actually had videos like this. If we just did a simple video texture, you found blurring and kind of mixing of things going on. But if you actually did this kind of approach, where you found the cut, you found a much better video sequence, because the blending was much better from one frame to the other. Again, showcasing in this video example of a waterfall. If we just did graph cut textures again found the boundaries, you can see a much cleaner version much defined transitions and actually much better quality video. So, in essence, we can use the same kind of thing here for us to help us do this. And in essence, that's what Agarwal et al did. So this was , just finding the continuous diagonal slice. And between boundaries of all of this to generate this new video volume. Well, with this, just it is rather than do this, find cuts between different one of them. You still get a continuous video volume here, that's shown by this. But now rather than having simple cuts like this, you find different regions. they use a min cut algorithm very similar to the graph, cut algorithm to optimize on the cost to get this. Please look at the paper for more details, but you kind of see that this kind of way of generating a video volume, allows us to have a much better way of looking at the results. And here let's look at the same example again. Here is the video and now with coherent fragments, much better no more sharing going on. And actually get much cleaner transitions between each and every one of the frames to generate a much better video texture.

11 - Examples.srt

Let's look at some examples additional examples of how they've done this. >> Here's the input video for another scene played at high speed. And here's the panoramic video texture created from the input. This entire grove of trees is one dynamic region Notice the rippling lake in the background. The middle portion of the scene is static. One frame of this panoramic video texture contains over 9 million pixels. The two trees form a single dynamic region. Finally, notice the gentle waving of the trees in the left. Here is another input video, played at high speed. And here is the output. The entire region of flags, as well as the highway in the back is one dynamic region. The motion of the vehicles on the highway is visually seamless, though they often disappear behind the occluding boat masts. Here is the input video for our final scene. And the final result. This scene is composed of one large water region, flags, and tarps covering some of the boats that blow in the wind. In conclusion, we present a method for creating high-resolution panoramic video textures of a location. Panoramic video textures combine the wide field of view of panoramic photographs with the infinite length of video textures. The result is an immersive experience of being there.

12 - Summary.srt

So in summary, I've talked about how we can combine, again, the whole ideas of panoramic for imagery with video textures to generate panoramic video textures. Again, you know, you saw lots of examples of generating these kinds of things. It's a step towards trying to build those types of active photographs that again you've seen in movies like Harry Potter and stuff. , displaying them reliably and in large format, , means an interesting challenge, which some of you can think about on your own. it takes the whole idea of video textures and panoramas and extends it to this new medium. Please do look at the paper by the authors of this idea and the two other papers again we had looked at when we looked at video textures earlier are also relevant. this is the website of the authors and there is all of this basic video, including also some of the code and also the data sets that they collected for this work.

# 07-01 Light Field.txt

01 - Introduction.srt

Welcome back. Now let's actually start looking at some very interesting advanced concepts. Recall when we started talking about computational photography. One of the things we talked about is we have an illuminated scene. We want to take this illuminated scene through a set of optics and processing devices to be able to capture and generate an image and that image was rendered in concepts of using things like pixels. But during that whole process I talked about the fact that we want to be able to capture and model the rays of light. Which is the primitive that we want to capture, one of the things now we want to do is look at the concept of light fields. The concept of light fields says at, at any point in a scene you can put a camera and capture all information about that scene. And from the three dimensional light that actually points and captured at the scene is what we're trying to capture. So, in the concept of light feed, what we're going to do, , is look at that whole concept. And see how we can actually capture light at any point and generate newer forms of cameras that actually let us have all of this information. That after we have captured the images, we can generate novel views from, and perhaps get to the pixels a little later.

02 - Lesson Overview.srt

The objectives of this lesson are for you to learn about, what is a light field, and the different seven parameters of the plenoptic function that go into creating a light field. We'll learn about different types of light fields. Then we'll look at variety of camera configurations and pinhole systems that can be used to view a scene. Then I will talk about the eccentric aperture and its impact on a simple lens system. Again, how we can use that to capture light fields. And then again, how we can actually use an array of pinhole cameras to create an array of images that could be used to generate light fields. And using all of this, I'll introduce to you a basic camera, a light field camera, that can capture 40 light fields.

03 - Recall Photography Light Rays.srt

Recall, one basic premise we started this class off. And that is photography is about capturing rays of light. We came up with this whole pipeline that we want to capture a 3D scene that's illuminated. We want to use optic, sensor, processing and display to generate an image. But one thing we actually also looked at was while the rays of light were the most important primitive in the 3D scene, we took that whole pipeline to give us pixels at the end. In essence, that was the goal of the part of the pipeline. So, in essence to us an image of a scene was nothing else, but 2D array of pixels. While we did actually believe that rays of light are the fundamental primitives that we'd like to capture. Most of the illumination in a scene is actually rays of light that follow a straight path most of the time from the scene to a sensor. We have looked at a variety of ways, where we have actually taken rays of light through a lens or even a pinhole camera and put it on a sensor where the image is formed. And we argued that computational photography, controls the various set of parameters. And the optic sensor and illumination and that's how computational photography is now impacting the whole discipline of photography. So the question I have really in this lecture is, are pixels, you know, are rather limiting form of what we can capture. So, in essence, we are doing a whole lot of this and ending up at pixels. Is that a limitation? Can we do something not to be just stuck with pixels at the end? Can we do something that will let us capture more of the environment? So then, for example, we can do something at the end to generate the right image. So this whole pipeline, , we do a lot of processing after we get pixels. Can we do something by capturing something a little earlier that would let us actually generate the pixels at the end, like the way we want them. We saw various examples of this already. Now we're going to actually kind of come up with a formulation that will let us understand the both the parameters of how this can be done and what are the variabilities faulted.

04 - Pinhole Camera and a Light Field.srt

Let's start off with the basics. The pinhole camera. Recall again, a pinhole camera had a opening, a small opening in a scene and , an image was formed upside down at the focal length at the image plane. An upside down image is formed there. In essence, we also kind of discussed this, that this point here captured all of the light in the scene at this point and , that's what was then projected at the back. Now, , you can imagine, that any point on this surface here, the gray patch which is where I have the pinhole, any point in the surface is also actually kind of capturing all the light. , since it's no longer a pinhole, they're not actually coming in. So, in essence, I could close this hole and put another hole anywhere else on this screen, and that will actually act like a pinhole. So you could actually put a number of points, pinholes, in this surface. And each one of them will then, for this point here, become the sensor, the light source, that will actually create an image like this, from a different viewpoint. So you can imagine now anywhere on this one surface here, a two-dimensional patch, I can put a pinhole and it will generate a different image, right? So in essence that proposes to us that any point in this whole region has the same capability, not just on this two-dimensional surface. So, a point here in space is also getting what we refer to again when we looked at earlier stuff as a bundle of rays of light converging to this point. And that bundle of rays of light here, , is captured here. And , if I was to put a pinhole camera I would actually capture an image here. So this is a pencil or bundle of rays of light at this point. , we need not actually just be stuck to all of the monolith's rays of light coming from just one direction. Any point anywhere in the world gets all of the light converged at this point from all directions, right? So any point in this scene is capturing a whole lot of rays of light converging to one point. And it could be any point here, any, you know, any point in this scene could actually be doing this. So just to kind of reiterate that point, a 3D scene, many points exist in that 3D scene and I'm just pointing out, you know, a few of them. So any 3D scene there are lots of these points around it, each one of them a bundle of rays of light are actually being collected at it in the 3D scene. Again this is just pictorially showing it, look around it at any point in the scene, for example, here, all rays of light are hitting this point. Another point here, all rays of light are collecting on this point. I could actually create a camera, pinhole camera, that would capture all the light and similarly, I could capture a ray of points of light here to capture all of the light that is hitting this point too. Now, just to simplify this, we can actually say is that each point here can be represented as a sphere, all right? All of the rays of light are converging onto this sphere from the world. So all rays of light are coming in to each and every one of these spheres from all three directions, right? So we can say that in essence this world is full of these types of, you know, specific spheres, and the point of that center is where all the light is accumulating. So in essence, any scene is full of these types of points. So in essence, again, that says any point in the whole 3D world is actually capturing pinhole types of camera information, except that we are not just capturing it. That point is already capturing that information. We just haven't put a sensor there to capture it.

05 - Parameterizing the Light Field.srt

So let's now actually think about creating a parametric representation of it. Put some variables and see how those variables change. Let's say that we are looking for a function P, which is the intense distribution at any point in space. So P is the intensity distribution at this point and now we'd like to figure out how that changes. Again, just to simplify, we can always imagine this to be a sphere. So just to look at this now, how would we parametrize it. , any point on the sphere, if i was to traverse it, there are two ways I can traverse it. Right? I can traverse it on the axis of rotation this way, which captures the slice in any direction. Then , force the elevation. The lo, longitude, latitude kinds of information. So two angles could actually let me traverse any point of this sphere. So that says, I have two angles, theta and phi. If I had those two angles if I could measure at any point here in this sphere, I could now traverse this whole sphere. Okay? So from this point on, I need to be able to take any vector and I have two different angles that could represent it. So P would be determined by these two parameters, theta and phi. Now, in the extreme case, let's imagine this is a huge sphere very far away and in the infinity. Then , each and every one of those regions would be a plane. Traverse that plane I would actually just be looking at x and y. So, in an extreme situation, we can almost kind of say, as the other parameterization to point out any information and this way would be, this as I said was a far away point on the sphere would be just x and y. So, , another parameterization, we looked at in these two dimensions would be P with the parameters x and y. , remember, light has color, has, which is represented by various wavelengths. So we need to actually also model that, which is the wavelength of light. And , as we learn, when we actually start looking at videos and stuff like that. Any scenes changes over time and we actually interested capturing the dynamics of the scene. Irrespective of captured on video or not scenes are always changing, so we need to represent the time aspect of it too. So, , now we need to add two more parameters. Lamba, which is the wavelength, t which is the time. We can do it for both the angle version or the string x and y version. So that's how we would now start representing a light field. Let's dig deeper into this one. So now , this point, the one we're looking at, also is a viewing point. Right? In the whole 3D world, I have a point, I need to know where the location of this thing is too. When we talked about where are two ways of capturing images, we also said that we need to know the location of the camera itself. So now the viewing point is also important, right? If I view a scene from this point or that point, things are different. I need to capture that and that's another set of parameters. So we can add those into our function. So now, if you notice we have the orientation of the ray of light. Location of the ray of light in 3D, lambda colors and frequencies and stuff like that and time. Again, a similar version for this could also be generated, where now we look at x and y.

06 - The Plenoptic Function.srt

So, let's look at these set of parameters. This is, in essence, referred to as the Plenoptic Function. The Plenoptic Function is, in essence, equivalent to putting a sensor, an eye, a pinhole camera, any point in the world, and measuring at that point, which, , the location of that point is Vx, y, and z, Vz. Recording the intensity of the rays of light with wavelength at any time, and , at all possible angles. Again, the bundle of rays of light around Vz, or in terms of x, y, and z. So, that's what we are interested in, and this is , what a Plenoptic Function is. It's a concept that was introduced by Adelson and Bergan in 1991, look at the paper, if you want more details please. Just to be specific, the term Plenoptic, is a Latin word, which kind of merges the words plenus and optic, and in essence, again, it's , is capturing the light traveling in every direction in a given space. This allows us to kind of start thinking of the Plenoptic, or a Light field camera that says, that at any point in the world we can capture a light field, and render pixels from that as needed. So, now actually rather than converting pixels right away, we want to capture a lot more information about the Light field at that point, and then we can render pixels whenever we need to.

07 - Light Fields 7 D.srt

Let's look at a variety of light fields. , if you notice we started off by looking at this, function here, which has seven parameters. So this is a seven dimensional light field. Has seven different dimensions, again, we know what these are by now. It captures the entire scene. And, in essence, this would be the best way to represent a holographic video. Right? At any point we can see all 3D, as we move our viewpoint we can actually see different types of 3D. And this is the concept really behind holographic video. Something you may recall from an old movie we saw Princess Leia show up in complete 3D. I'm not going to go into a lot of details about holography here, but I encourage you to look at this kinds of stuff. this has been one of those bigger efforts of trying to understand imaging and being able to generate imaging out of nowhere into ten space, like in this case here. We can also, , create a five dimensional light field. Well if you notice it's there ignoring the time and wavelength. So we're kind of not paying attention to the video aspect, temporal aspect, the dynamic aspect. And also we're kind of saying it's okay, let's not worry about too many different types of colors. Let's simplify it as much as possible. It's really more aimed at capturing the viewpoint and the direction of the information. And you've seen this kind of stuff at very simple holograms. Like this one that you most probably have on your credit card. I point this out because one of the people I had privilege working with, Steve Benton, was one of the people who has invented these. And actually was always involved with a whole lot of holographic video research when I was a graduate student.

08 - Light Fields 4 D.srt

Another thing we can actually now look at is a four dimensional light field. Same equation as before there are five parameters in there. But we want to actually use that to reduce the dimensionality by one, so we can actually now pay specific attention and create controllable types of light fields. In this case, it's a four dimensional light field. What we really want to do is create a bounding box and suggest that a space of all lines in 2D space is 4D. Let me show you what that means in a bit. But in this one, the basic assumption is, we will not be able to represent occluding objects with different viewpoints and directions. The kind of concept that exists here is something similar to a globe that you may have seen. Usually the plenoptic function, or light field, is five dimensional. But in this case, we can kind of start seeing is that, if this was my world here, anything outside of the scene, outside the sphere of the snow globe here, light does not actually get occluded by objects. And therefore, it could be represented as a 4D light field. The best way to represent this is by creating these two different regions, two different parameterizations, u and v, and s and t. And that will allow me to create a four dimensional light field like this. To achieve this, , what we talk about is having two different planes. Each plane has coordinate axes u and v, and s and t. Those are my four parameters that give me this whole parameterization in four dimensions. In essence, you can imagine this to be a beam or a slab of light and everything is contained within it. So light really flows from the uv plane to the st plane. I'll show you examples of that in a bit. Just to help us with our parameterization, we can always kind of say is that parameters both for uv space and the st space are just simply between 0 and 1. This will allow us to create an array of images that we can actually now use to generate a light field from. To help us understand this, let's kind of create this parameterization uv at the camera plane and st at the focal plane. Two different parameterization, different discretizations would exist for each one of them. Now at any point, I can actually now have a ray of light come out from this point here. So now I can put a pin hole here, and that would capture this whole scene from this point. I can, , have a pin hole at this point, which also captures this whole thing. There could be different types of information here. And, , using this we could generate a variety of different arrays of images at the uv plane. And, , based on which viewpoint I look at, I would have a different field of view of the scene.

09 - Visualization of a Light Field.srt

So let's now visualize this light-field from this perspective. I have my uv plane and my st plane. Any point on the uv plane is now looking at this whole st plane. So any ray of light arriving at one point on the uv plane is arriving from all points on the st plane. What is the st plane? Well, here is a simple way of looking at it. The uv plane has variety of arrays of images. Now you can imagine that at this point, I've taken a bunch of different images at different orientations. So, again, if I could create a pinhole here, this would see the scene differently. So what I would have this and the st would just be one single image, from different viewpoints. Now by just reversing this versus that again, imagine this to be in the st plane and this to be in the uv plane. I actually now have an array of images and I can traverse them one by one by looking at them. , the interesting parts of this is I can now , be looking another one and interpolation can be used to generate in between images. What happens in the other case? Where now, I'm looking at the st plane and extracting information, so on the uv plane. So rays arriving at one point in the st plane, that were bound for all points on the uv plane. Again, the same light structure here, but except this time around, we're not seeing the object, but we're actually seeing more local details of the same objects. And , this would actually give me a uv plane. So this shows me local details in the uv plane, while sd plane was actually showing me in the previous case a lot more detail of the object. So the best way to think about it is imagine, we can create an array of cameras and I would put a bunch of different cameras in a grid pattern and use that to take a bunch of pictures. And now, I would have a huge amounts of pictures there. And what I can do is using this kind of stuff, move from one view to the other, interpolate between one of them and actually, that would actually create a light-field rather than pixels themselves.

10 - Capture a Light Field, Store and Render.srt

Let's look at the simplest example of a light-field. Here is a two-dimensional light-field which , same viewpoint, we never change. And the best example of this would be a panorama, something we have looked at quite a bit in detail in this class so far, right? I put camera at one point. I rotate it around to be able to traverse the whole spherical region around it, and that can be used to generate an image like these two types of panoramas. we have looked at a variety of things like Google Maps, Google Street Views and also other types of panoramas and stuff like that that we have looked at including photo and stuff. They also use geometry, as we have looked at.

11 - Light Field via a Pinhole Camera.srt

So now actually, lets start thinking about how we can build a pinhole camera. Let me do this by a simple exercise. Lets imagine I have a scene with an object like this. So now the reason I chose a simple object, four corners and we can actually now trace rays of light through it. Let me build a small pinhole camera, which is right here. Rays of light coming from the three corners go through the pinhole and , hit the image plane here. And I can now recreate the image of this thing on my camera. So this would be a single pinhole. What happens if I can actually now play around with multiple pin holes. Remember, we talked about stereo? Well, this is equivalent to that, but in essence also trying to now give me more information at the camera level. Remember, this would be two different images forming. And now if I could save both of them, I can actually create more information, perhaps geometry, and then use that to create pixels later. So in this case, , two images would be formed. And I can use that to create geometry of the scene or a depth map. This would be an example of double pinhole. Another example, again, to look at the same scene. I have one pinhole camera. All the rays are coming in this way. But, what I also do is move this camera by a bit. Another pinhole camera, same aspect here except that I have moved the camera to be able to do this. This allows us to capture things like motion parallax. Depth parallax is what we've got here. Motion parallax is what we get here. Again, recall what we had looked at when we looked at, stereo imagery. Final example, it would be at this instance here, right? In place of pinholes, we can replace it with the lens, and , if the lens does the right kinds of things we would be able to create an image, and be able to catch a variety of things with that image. But, , remember, now we can actually play around with control of the lens to get different types of things too.

12 - Single Lens System.srt

Now, lets actually build on this idea. And ask ourselves a series of questions. I'm going to make this into a quiz for you to kind of think about also. Lets concentrate only on a point light source, an idealized point light source. So, lets say imagine I put a idealized point light source in this set up here. I have a lens based system and you know, this is my camera and I have three different set ups. I'm going to show you one of them first. Okay. focus point. This is the image plane. This is where the location of this is. This is an ideal camera and the image is formed here. this pin single point light source. Would then , create a one intensity pixel on my image sensor. Right? Focused and everything else. The question for you now is if I move this point source with the same camera. And now I generate this thing, on the image plane this is happening. And, for the other one, this is happening. So, just to see if you've been paying attention a little bit, simple stuff. Just put in there is what's happening in these types of three different scenarios. Put the number, one, two, and three in the right box. Which one of them is the near object, and the output would be blurred. Far object, the output would be blurred, or in cases of, you know, in focus image and a perfect image would be generated.

13 - Single Lens System.srt

So this is very simple, and I pretty much answered the question when I set the quiz up. This , is the object is near, the output is blurred. The object is far, the output is blurred. And , everything is in focus. Just a simple way of kind of making us remember some of the ideas we've looked at with pinhole cameras.

14 - Single Lens System 2.srt

So now let's look at another simple lens system. Here I'm going to actually just complicate things a little bit, and add an eccentric aperture. What that means is, I'm going to cover a part of the lens, so no light is going through, so only part of the lens is open, and it's eccentric, it's no longer symmetric. I'm going to cover just the, this part of it, so this lens is open. What happens to rays of light in this instance? So this light source, , will go through, and create a point where we expect it to, right? So amount of light is reduced, but we will still get something at the point we want. So , that means an in focus object still forms a point image where we expect it to. When we move the object closer, what happens this time? Remember, again, from what we looked at in the last instance, in this case, forms near object is blurred, but it's only appearing to the right of the optical center, right here. Right? In the old case, , when we didn't have an eccentric, we had, , a blurred region everywhere. But in this, by, by covering this part of it, we're only getting it on this side, and it's blurred. No rocket science needed. If I move the object far away, same experiment, and this time around, the object is still blurred, but is blurred to the left of the center line. So just by creating a simple, eccentric thing, we're now actually capturing a lot more information. Yes, we're losing in the quantity of light, but we're capturing more information, even about the distance of the object itself, too. Again, something we looked at when we looked at things like depth, and all that kind of stuff, earlier.

15 - Encode Direction and Intensity.srt

Now let me actually have you play around and generate another simple way of encoding the direction and intensity of lights. Again, three different examples, select which rst will be highlighted for 2 and 3. And I'll tell you what rst is. What we're going to do is we're going to focus in onto this region. And , any ray of light in this thing is coming in. So if you notice in this form I created a bunch of small pinholes within this whole system here. There's a lens, but down here, after the light goes through the lens, I've created a bunch of pinhole sensors, right? Smaller pinholes, and here I'm just showing you an example of three. Any light goes through this pinhole will hit this one here, and this one will go hit there, and the straight one is going in the middle. We will end each one of these pin holes create three different regions, r, s and t. Right? So in this case, when the light is coming in, it's going through the pin hole, the straight line is straight, going straight to s, this one is coming and hitting r, and this one is coming and hitting t. So let's see what happens in the case of the first one. I have the object at the right distance, here I've just kind of color-coded the lines red, green and blue. Not to imply the red, green and blue channels, just to differentiate it. When it hits, , this instance, you notice what's happening. Now , to help us, we can create a small decoding platform out here, r, s, and t. So here, , this red one is coming in, converging to the blue point, and hitting this point. Looking at this, this suggests that red should be t. The green one is s as it should be. And the blue one is r. So, here in , depending on where we are, we are able to create a decoding in this simple three by three grid, to figure out where different light sources are coming in. Again, these are not red green and blue, I'm just showing you rays of light. So, for these two I would actually have the same thing right? Except this time around the rays of light are not converging to the same point. Remember the previous two experiments we did. And for the far object, same thing. Simply put, I want you to now fill out the r, s, and t for this, and for this, as to which ones would be the ones highlighted for the red light, the blue light and the green light. Just fill out the regions correctly, and I will show you the answer in a bit.

16 - Encode Direction and Intensity.srt

So, here are the simple answers for this one. We knew this one already. Here the red light is going to be hitting the t. And the blue one is hitting r. So, this would be my output for this one. And, , as we've seen for everything else, could be flipped completely. Blue is, , r, but for this bin, not this bin anymore. Similarly blue was in this bin, and red for this bin, and this is how I would actually get the result. So now, just by putting a bunch of pinholes after the lens, the camera plane, and pinhole here, we were able to extract more information about the light itself, the rays of light, beyond just, kind of, what were the intensities. Right? And that's the one thing which we actually want to start encoding. So when encoding the direction and intensity of light using the simple way of combining a camera system with a bunch of smaller camera systems. The lens system and small pinholes, but an array of them. So, in essence what we've done is we've added miniature pinholes at the image plane, and this allows us to analyze the structure of light at each and every micro-pixel.

17 - Lens and Microlens.srt

So this is actually an interesting idea because that can allow us to start creating a light field or a plenoptic camera, something which was introduced by Ng et al in 2005. I have a subject, I have a main lens. And what I can do now is add a linticular or a microlens array before the sensor. And what that could do now is anytime it goes through, it can also do more analysis to figure out more information about the light. Just pretty much light what we saw, by putting it in a series of pinholes. So what does a lenticular or microlens array do? Here's a simple example. This is a lenticular array. By just placing these types of small lenslets like this in front of a screen, We can actually start seeing depth. So, lenticular array is putting in different lenses that can be seen, in this case, by the left eye or the right eye. And , allows you to generate a newer image that will start encoding depth and being able to see depth in images like this. I mean, lenticular arrays are one of the more popular arrays, you may have seen them. Also, we're showing 2D or 3D images in a 2D plane. Or sometimes they could also be just cylindrical lenses that could be used to form lenticular arrays. Again I encourage you to look up this kind of stuff. You may have seen a whole lot of these types of things already. But just in essence of this kind of stuff, we can actually putting something like this in front of it cylindrical lenses like this, before a sensor, we can start actually now capturing more information. So now let me connect it to something we have also looked at already. Remember the UV and the ST planes? UV was where the lens would be put, the camera plane, and the ST is where the image plane is. And here, what I do, is I put in a bunch of smaller micro-lens arrays. And according to what we did previously, that is, put in small pinholes. If I put in this kind of stuff, now I can actually create a light field. Because what we have done now is use this way to create an array of cameras, and each camera is capturing different types of things. Once I have this kind of stuff, what I can now do is traverse from u and v and s and t To start creating not just one image that was captured but a seas of images. And, , interpolating between those, I can actually start getting more information than just one single image. In essence, I would've captured a light field and generated pixels whenever I want to add to the fact

18 - History of Light Field Camera.srt

So this has actually been something that's been, kind of been thought up and hoped for a long time. In fact, as early as 1908, Lippmann proposed that we could actually use this to generate what is referred to by him as integral photography. He actually was a physics person, and he won a Nobel prize for this kind of stuff. you know, lot of fun stuff was done by actually doing the physics on the sensors like this. So, I should, , spend a brief moment on the history of light field cameras. The concept has been around for a long time, as early as 1908. And Lippmann, a Nobel laureate in physics, actually introduced the concept of integral photography and talked about how, actually, you know, the physics of light itself could be used to reproduce colors based on interference patterns and stuff like that. Since then, much work has happened. 1930, they kind of constructed parallax panograms. In 1992, Adelson and Wang proposed a plenoptic camera and used it to generate stereo from a single lens. And in fact, some of the stuff that I just showed you, was based on the work that they did. Similarly then, kind of early stuff happened, in terms of building light fields. Again, the example of using the UV and the SD kinds of stuff, planes, UV plane and SD plane was based on some work these people did. And , the plenoptic camera that you also see now. Actually what has happened now in about 2012, Lytro actually invented this light field camera which is commercially available. There are advanced versions of this now. When we looked at some of our earlier cameras, we saw various cameras which had an array of cameras connected to each other. They can also be used to generate light fields. Much more advanced stuff has happened in this space, and I encourage you to look it up on your own and study more about it. But again, you know these, this is one of commercially available light field camera these days.

19 - Light Field Examples.srt

Let me show you an example of the Lytro Light Field camera. Here, , if you see, you know, I can focus by clicking on different regions. In essence, what this camera does, it captures a stack of images at different focal lengths. And interactively, you can choose which focal plane you want to visualize. So, in essence, what it does is actually to some extent it's learned photography. Here you can even see parallax, right? I'm just moving it, and you can see a little bit of depth going on. Again, is happening between the uv, and the sd planes. Again, by using the microlens length arrays, they were able to capture a lot of images, and now this is interpolating between all of them to give you both depth parallax issues, and also focal planes. Again, when we talked about the photography we look at this whole concept, right? A photography said, find a bunch of images that are just different by a bit, and now, we can actually bring those images to create a representation. In this case, they're using it to create a light field by putting in a smaller micro-cameras with a bigger camera system. More information, we capture it each and every pixel. Then, , we can actually generate more newer forms of pixels after their captured. So, that, in a sense, is the idea of doing all of this.

20 - Summary.srt

To quickly summarize, introduce the concept of light field, to kind of let us get away from the whole concept of just pixels in cameras. We talked about the plenoptic function, and the parameters that could be encoded in the plenoptic function, how we would represent them. We talked about different types of light fields and what dimensions they captured from holograms to simple panoramas. We looked at how a pinhole in a lens system can be used to analyze the scene and capture more than just pixel information. I showed you just by creating an eccentric aperture we can actually encode more information. And also that combining a lens with a, an array of pinhole cameras, we can encode direction and intensity of the rays of light. And, using these concepts, and, again, a lens and micro-lens arrays, lenticular arrays and stuff like that, we can build a 40 light field camera. And I demonstrated at least one of them. The premise of all of this was to, , kind of now ask us to push beyond the simple metrics of cameras as we know them. And start thinking of cameras that capture a lot more light fields, and that could be used to generate images by combining information from the different types of sensors. Further reading on this one available through the papers that I'm listing here. Some of the earliest work on defining the plenoptic function comes in 1991, including the first system to do it in 92. And then , the graphics people developed these approaches to kind of render and capture light fields. And then , a light field camera itself. Much more work continues to happen in this space. I encourage you to look it up, a very exciting discipline on its own. >> More information is available elsewhere including the book that we're looking at and also at the Lytro site.

# 07-02 Projector-Camera Systems.txt

01 - Intro.srt

Welcome back. Today we are going to talk about projector camera systems. Remember, when we started talking about computational photography, I talked about capturing an illuminated scene, and being able to generate a novel image from it. Now in that pipeline, I did talk about the fact that you could actually control the illumination, the optics, and the camera, and the processing all together in the pipeline of computation photography, to generate novel images. Today we are going to talk about a specific set of instances where we're going to merge, couple, for example, a camera and a projector system. A projector is, , where we can actually control light. Remember the example of dual photography we covered initially, in the class, where being able to illuminate a scene, and being able to control it with a camera, allowed us to actually see newer things. Well how, now I'm going to actually show several different examples of techniques people have used, of coupling cameras and projectors together, to generate novel forms of images.

02 - Lesson Objectives.srt

The objectives of this lesson are for us to learn about, how we can actually control illumination in a scene. How we can actually use a projector to control a light source. How we can use a projector as a controlled light source in a scene to be able to then project the right kinds of illumination in a scene. And with the construction of both these, how we can actually build a projector camera system. We'll talk a little bit about how we can calibrate the illumination in a scene from a projector. And then I will actually showcase a variety of different existing examples of projector-camera systems or procams, as it's usually referred to. Now, in this lecture I'm going to be showing you a whole lot of videos. So please watch these videos, I will also be providing you links to these videos and you can watch the full length versions of these videos, offline, on your own.

03 - Recall Computational Photography.srt

So to get us started, let's recall the basic elements of computational photography and the various stages that we actually have looked at many times in this class. we have a 3D scene, which is illuminated. We want to actually take the illumination, use the optics to focus it onto a sensor. Sensor gives us information, allows us to create images and pixels that we can process to improve the quality, which is sent to a display and then used by a user. And this has been the pipeline we have been looking at for computational photography throughout the class. Let's look at it much more pictorially. Again, we're trying to remember what we have already looked at previously in this class. We can actually take a 3D scene like this and illuminate it with a projector. In this case projector is a light source. And to control this light source we can put some sort of a controllable aperture or modulator in front of it that, , opens a different regions here, or different regions of this thing to illuminate the scene. So, in essence, we can now control how the light is actually being entered into the scene and illuminating the scene. On the other hand, we can do the same by actually having a camera in the scene, which is now capturing the light illuminated by this light source. And, , we can also put a modulator, a controllable aperture in front of the camera which only allows a specific light to go through in different parts of this region. Again, both of those are controllable things. And this was the idea we had before of how we can control and computationally control each and every aspect of how light is shining on a scene and how it can be captured to create an image of different types, a novel image sometimes. So this has been something we've looked at before.

04 - Controlled Illumination 1.srt

So the goal of this lecture is to start, us to start thinking about how we can control the illumination in the scene. So this was our projector system, and using this I'll actually showcase two different examples. One, I'll actually talk about the Lightstage, where a light is shown on a subject from these controllable light sources. And, , based on how this light are, is illuminating the scene, we can now create a model of learning the illumination of this person from different light sources. And that will allow us to do re-lighting. Another example, that we've looked at when we actually talked about extracting depth information from images, was this concept of a trimensional device on an iPhone, or software for trimensional on an iPhone, which let us take pictures with different illuminations and allowed us to capture a 3D model. Let's look at them again just briefly to help us understand the importance of controllable light or controllable illumination for variety of applications.

05 - Lightstage.srt

So first, let's look at the Lightstage. Lightstage is a system where a subject is illuminated by a variety of these controlled LED lights. They're shone at different res different times to be able to kind of illuminate the face from different directions. Once, we can use this, and we can capture all of the variety of different lighting conditions. The goal, really, is to then put the subject in a different environment, a CGR, computer graphic environment which has different lighting. And , now we can relight the image based on where the light sources are in that environment. >> In this video, we present a technique from how to find the lighting and reflectance of a live action performance in post production. By lighting the actor with a timed multi flexed series of lighting conditions at a high frame rate, we can simulate their appearance in Nav illumination. Or, we can modify their reflectance functions to produce subtle or stylized modifications to their reflectance. Our illumination device includes 156 white led light sources, which can be run in arbitrary patterns synchronized to a high speed camera. The device also includes a matting background, allowing the subject to photographed in silhouette to obtain a matte. The system rapidly illuminates the subject with a repeating set of patterns, forming a basis for the sphere. Our camera requires up to 4,000 images per second, which can cover the full set of lights up to 24 times per second. In this work, we explored using three different lighting basics, single lights, triples of lights, and hadamard patterns. Our paper discusses the various advantages and disadvantages we experienced with each basis. Here, we see a mosaic over 180 image sequence that includes the triangle basis, captured every 12th of a second. We can produce a full motion film of the subject with Nav illumination by taking a linear combinations of the basis images. This lighting can be captured from real locations, or it can be designed by a cinematographer. Unlike previous techniques, this allow the lighting to be designed and modified in post-production without creating a digital version of the actor. >> So here, you see an example of how a camera system coupled with a controllable light source can be used to capture a subject. And with ideal lighting conditions and how we can take that control to generate novel images and in case here, video sequences. And it's a technique that's widely used in the movie industry at present. I encourage you to look at the website here, and also Paul Dubovick's webpage and the paper that I'm also citing at the end of this talk. To look at variety of examples of this kind of stuff. And how, actually, this whole technology, which is again, a combination of cameras and light sources, can be used to generate this new form of imaging. That, now, as I said, more importantly, takes the subject, and then re-lights it in the new environment. Lightstage has gone through a variety of developments. And now, actually, let me show you another video where they've taken this whole concept of Lightstage. And in the previous case, they applied it only working with just faces, but now they're actually done this to allow it to work for walking subjects. So here, you notice that, , the same set up except that they have a treadmill. With a whole lot of led lights again. But , in this case, they can capture the motion of the person walking. now, using this kind of a scenario, they can control the lights during the action of a subject moving. Generate the same kinds of lighting bases as they did for faces. This is equivalent to the mosaic that we saw for faces. And then, , they can now re-light it, and this time, the subject is walking. They do a little bit of additional work to align the subjects using things like optical flow, and stuff. But now, you see that the subject could be put in different types of environments. And completely re-lit in that environment, with a variety of different lighting conditions, and also different viewpoints. So, , the motion is recreated from the one that was captured in the lighting stage, at the light stage itself. And , using this, they can start replicating the subject, different viewpoints. I encourage you to look at the website again and also the paper that's referenced with this. But , the idea really remains is by now taking cameras and lighting sources, we can actually generate novel types of images and videos.

06 - 3D Scanning on Mobile Phone.srt

So now, actually, let me showcase a much simpler example. How light could be used and variations in light can be used with a camera to extract shape information. This is something we've looked at already before when we talked about extracting shape. And we said okay, now, we can take light and change the variations in light to actually model and extract 3D dimensional shapes. Let's look at this very simple trimensional iPhone app. >> It's a 3D scanner for the iPhone. What it does is it shines light on your face from different lighting directions. It takes those images and interprets them to create, create a 3D model that you can then actually physically realize with a 3D printer. So this is really the first time you can go from just an iPhone to you, do the scanning to then actually get a physical 3D object at the end. So the way that works is, you go to a dark room, you hold the iPhone to your face. And when you hit capture, it shines light on your face from four different directions. It then combines these images into a 3D model which you can then email to yourself. If you go to a machine hooked up to a 3D printer, you then can just physically print that email file, right away. And so within about a half hour you can go from scanning yourself to having a physical copy. So what's innovative here is that we're using the iPhone screen itself as a light source that we use to shine light on your face from four different directions. So that's really the key technology here that makes 3D scanning possible. And with, along with the front facing camera that's observing your face, while the light is being shined by the screen itself, we have just a compact 3D scanner that you can carry around in your pocket for the first time. >> So this was actually Grant Schindler who actually finished his degrees at Georgia Tech and actually came up with this idea. And he started a company from Georgia Tech on this topic and a variety of other things he's also developing and some other computation photography apps. You can look up, look him up and find more apps from him too.

07 - Controlled Illumination 2.srt

So the idea behind the example we've seen so far, has been that we're interested in controlling illumination. We can control illumination with a camera, and that will allow us to extract various types of information from a scene. So, given a control lights, we can scan or re-light a scene. Scan to extract 3D information, re-light to be able to then be able to generate novel images with different lighting conditions. So, one big question now comes up is, how can we actually do much more elaborate computer control of light? In the previous example, we showed how we can use an iPhone display itself to light a scene or how we can use LEDs which are computer controlled. But now we want to actually be able to do much finer control of the light. Well, that's where projectors come in, because projectors can be thought of. And these days, with projectors with micrometer devices and stuff like that, and the high intensity light that they actually can generate, can be used as really detailed types of controllable light sources. Let's see how we can use them.

08 - Projector Calibration Quiz.srt

So to help us kind of situate this, let's actually think of a simple quiz on, how we can calibrate a projector? So here , I have a screen that I'm actually putting a projector in front of. And when I project on a screen the shape of the image would be dependent on what orientation the screen is with respect to the projector. So for example here, the projector might be a little closer on, on the side here, that's why it's showing you a little deformed image. our interest is, how can we automatically get a perfect rectangular image to display on this projection or projection screen, and has the same, and exact aspect ratio of the original image? So simply put, we want to be able to figure out, how we can actually project this image that actually shows perfectly same aspect ratio to fill the entire screen? , one way to do this would be, is to move the screen so it's completely perpendicular to projection of the projector. Another approach would be to actually move the projector, right? I can just move the projector right here, perpendicular to this scene here, and that would actually create a very good image. I can move it up and down a little bit or a third option that is, how about we just use a camera to capture the image? And then use that to analyze the image and then transform the image on the display of the projector to warp it to the exact dimensions of the projection screen. So , choose the right answer here.

09 - Projector Calibration Quiz.srt

So obviously the answer I'm looking for is this one. because what we're interested in is , if I take a picture of this scene here as shown here, it will show me that this original image is deformed, transformed, right? in this case this appears to be a perspective transformation or projected transformation. Now I actually can link this camera to my projector through a computation process. Which will take this and can now do the inverse of this image to kind of say, okay, I know that this, I know the original image because I actually know the original image because they're controlled. This has already been displayed. Knowing the original image and, perhaps, not even the original image, just that kind of the outline of the boundary here. Knowing the image will help us a lot more, I can now figure out the inverse transform. So what it would mean is I would actually change the way the image is displayed on the projector to make it have the right you know, aspect ratio on the screen. So in essence this would mean in the process, sometimes moving this kind of around is keystoning, and we can actually do this much more in detail by actually transforming the image at the display so the projection is actually appropriate. So that starts saying that we can actually do calibration by combining a camera and a projector system.

10 - Projector Calibration.srt

Let's look at another example of this. This time around, what I want to do is I want to project an image, but I have a bigger display or a bigger screen and I actually been allowed four different projectors. What I want to do now is take an image and use that four different projectors to project one large image. A much different aspect ratio onto this large screen. So this is the output I want. So, to achieve this, what we need to do is start thinking about what's going to happen. , what I can do is, I can convert this large image into four smaller images and distribute them to the four different projectors. , the first projector's going to project it this way, , going to have a different aspect ratio here. Another one is going to project it beneath, make sure there's some sort of an overlap. Third one projects another image here, and the fourth one is projecting. Now what we can do is what we'd actually talked about in the previous slide the quiz itself. If I actually had a camera, you know, a camera looking at this scene, I can get the information about all four of the different images and suggest to each one of the four projectors how to inverse transform the image, and then also kind of do the processing at a central server that actually does the stitching. So I can do stitching for four different images after I've done simple transformations and applied to generate a full image. That is actually the idea that we will see in action next. So here we are actually going to see an example of how they've done this with six roughly aligned projectors. What each projector does is it it displays this gray code pattern. We know the gray code pattern, and actually it projects it, and using this projection, and getting the reflections, captured by a camera, we can now use the inverse transform to be able to transform each and every one of the projectors. Here, , we see the process of it doing this for each and everyone of the six projectors. Now , by just doing analysis on this one I can actually learn more about the shape of all of the images. And this now you see all six cameras have been aligned. This is the final projection of the six different cameras. Here, now allows you to generate a much larger display by combining all six of the cameras, here you still see a little bit of an artifact, but you see that if the camera intensities and alignments can be done correctly, you won't see any.

11 - Calibration with 1-Pixel Sensor.srt

An additional example of doing this kind of projector calibration, in this instance with a smaller object in the scene, where you want to actually display the information from a projector, is showcased in this video here. >> Hello, I'm here to talk to you a little bit about our system for automatic projector calibration. The basic problem we're trying to solve is the task where trying to fit a projected image perfectly onto a target surface such as this. Typically that requires a screen to be directly in front of the projector, and at a very specific orientation, to get an undistorted image. But what we would like to do is be able to place the screen in any location that's convenient, and then calibrate the projector onto the target surface. To demonstrate our system I have an unmodified projector, a computer beneath the table, and this target surface. If I turn this surface over, you can see that we've implemented it with some electronics. What we have here are optical fibers that channel light energy from each corner of the screen, to an electronics package containing four optical sensors and a USB connection to the PC. When I turn the screen back over you can see that there is no visual evidence of the fibers. The white surface acts to hide the fibers and also provides a light diffuser which improves calibration reliability. To calibrate onto the target, I can simply place it in the projection area, and then project a series of great coded binary patterns. These patterns uniquely identify every pixel in the projection screen, allowing us to discover the location of each fiber. We can then use this information to project a corrected image. Here in this close-up, you can see that the quality of calibration is very high. The discovered location of each fiber is actually closest to the nearest pixel. The prototype shown here is capable of performing the calibration in just over one second. We're currently working on techniques that will hopefully allow us to achieve interactive rates. Here's what the calibration process looks like from the perspective of the target. You can see the irregular flashing from the projector. This pattern of flashes indicates the location of the camera in the projector's screen space. To illustrate the robustness of this technique, we will gradually decrease the projection angle of the calibration. We have found that the calibration continues to work reliably, even though the projection angle is less than two degrees. In this last calibration, the screen is actually facing slightly away from the projector. The expansion of the projection frustum sufficient for this technique to work. We can also fold the optical path using a mirror, with no effect on the calibration process. The image will be automatically reversed, since the orientation of the image is determined by the screen and not the projector. This wire frame test pattern that I've been using is mainly to make it easy to see the quality of calibration. By using open Gl or Direct X, we can warp real time video on low cost commodity hardware. We also have an implementation that allows us to work the active Windows desktop creating a fully usable, calibrated display. IN this board, we've added a total of six sensors, one in each corner, and two across the middle. This allows us to calibrate two projectors that are placed side by side, creating a method for automatically stitching multiple projectors. We calibrate each projector individually, and then blend the two images together. >> So the previous method showed us how to do the same thing here. The big difference here is in the previous system we had a camera looking at the scene. In this case they've actually improvised and put a small fiber sensors at, in this instance six locations, in the previous one they just put four of them. So in essence they have is a one pixel sensor. Just that single pixel and then the grid pattern that actually is shown on the surface allows us to kind of uniquely identify exactly what each one of those sensors is seeing. And that allows him to calibrate for this kind of information. A really, nice, unique, simple example, where I can, still using a projector and a camera, but in a much simpler manner. the camera is a simple slow lens one pixel camera. And they use multiple ones

12 - Light that is Aware of Obstructions.srt

Now let's think about a light source, an illumination source that can be aware of what things are in front of it so it can actually react appropriately to the obstructions or objects in front of it. Here you see a subject walking, and , you saw a little bit of a shadow, but if you notice, the shadow is not there. Imagine in this instance if there was light being shone on this person, there should have been a shadow. And he's moving around, you kind of see a little bit of the hands pop up but they vanish. Also most projector systems like this, if you notice, there must have been, if the person was there, a much more thorough light source on the person themselves. So this actually is done by actually a scenario like this, where using a camera looking down this way, you can actually track where the person is. So once you know where the tracking of the person is, you can actually turn off the signal that actually is on top of them. , that's the reason there was a shadow. If you turn it off there won't be a shadow. And, , there's another projector. So having a, you know, two project, projectors here, this one fills in the hole, so now we get a full image from this one, and this one is no longer shining lights on it. So by having two projectors and a camera tracking a person, in this instance, you can now design a system that will allow you to get images that are full all the time and also not having a lot of bright light on the subject. This just demonstrates that you can do this in pretty quite, you know, pretty efficient computations, and it works for videos of this type. And , by having these projectors, and remember what we learned about how we can do alignments and stuff like that, we can align both the images to be perfect, like we've done for all forms of image alignment or camera projection alignment camera calibration stuff that we looked at.

13 - Programmable Headlights.srt

Now let me show you, perhaps, the most exciting example of controllable light sources or a projector camera system, where both a camera and a projector are used to generate light that's controlled enough to actually become a headlight on a car. This is what going on at CMU, and I'm going to let them describe it to you. [MUSIC] >> Automotive headlights have been on the road for nearly 135 years. But surprisingly, driving at night is still very dangerous. Today's headlights are bright, energy efficient, and even adaptively illuminate the road. But even the best ones are not generally for performing multiple tasks, and they typically require mechanical parts. We have developed a headlight with a single hardware configuration that can be programmed to adapt to any road environment. The key component of our design is a spatial light modulator, such as a DMD chip, commonly found in DLP projectors. The benefit of using a spatial light modulator, is that a single beam from a light source can be divided into a million smaller beams, each of which can be controlled to react to the environment. The spatial light modulator is optically co-located with a camera that senses and captures images in front of the view. The images are analyzed with a processor that also controls the spatial light modulator to appropriately illuminate the road environment. In order to be useful at highway speeds, we determined that the headlight needs to react within two milliseconds of acquiring an image. A high speed spatial light modulator was built by combining a custom DMD board and the chip with the optics and the light source of an off the shelf DLP projector. The result is a spatial light modulator capable of display rates of over 1,000 hertz. The prototype is a little bulky, so suction cups are used to mount the prototype on the hood of a pickup truck for road testing. We demonstrate the versatility of the headlight by first addressing one of the biggest problems while driving on the road, the glare problem. The solution to this problem is straightforward due to the high spatio-temporal resolution of our prototype. After detecting oncoming vehicles by their headlights, only the light rays directed toward other drivers are disabled. This works for any number of vehicles in any number of lanes on the road. This should look familiar to anyone that has been blinded by headlights. The anti-glare feature of this prototype is disabled to emulate the glare typically seen from standard headlights. When our system's anti-glare function is enabled, the difference is very dramatic. The oncoming driver is no longer blinded, and the vehicle and the road environment become more visible. Now how does it look to the driver equipped with the headlight prototype? Because the prototype has unprecedented resolution over space and time, there is little perceptible difference to the driver even with three oncoming vehicles in the other lane. So with the programmable headlight, drivers can use the brightest headlights available, or always keep their high beams on without losing too much light. This is a stark contrast to LED-based anti-glare headlights. Another problem that our headlight can address is poor visibility in the snow. Driving at night during a snowstorm is a nightmare. Snowflakes appear as bright flickering streaks and are very distracting. The problem is mainly caused by light from our own headlights reflecting off the snowflakes and back to our eye. The solution to this problem is very simple with our headlight by avoiding illumination of the snowflakes. In other words streaming light in between the snowflakes the visibility of the snowflakes will be reduced. This might seem like a crazy idea, but preliminary experiments with artificial snow demonstrate that it is technically feasible, while significantly improving visibility, with little loss in light. The current prototype is ten times faster than our previous prototype with much better performance, bringing this technology closer to reality. >> So, hopefully you will agree. This is pretty impressive, what they are trying to propose here. Again, if we really think through it, this is a system where there is a camera. There is a controllable light source. And both of them are in tandem, working to be able to illuminate the scenes, or turn off the illumination where there are objects that we want to prevent being illuminated. So pretty much like the last instance we looked at, where we turned on the projection, where there was person, occluding it. So we actually turned off the projection where the person was occluding it, not get to the screen. Now in this case, we have a light source that flickers out, for example, where there is another car coming. So we can turn off the lights there, or where there small flakes would be, and we can turn it off, or rain and stuff like that. Very impressive, you saw a little bit of the description of what we have talked about before. That is how we can actually create modulated light sources and how we can actually compute all of the information within it. Pretty exciting. Again, I encourage you to look at the details of this on your own from the website down here, and also the video that I have also linked.

14 - Room Alive.srt

>> Now the final example and I want to show you this pretty interesting example that was produced by a bunch of people at Microsoft with collaboration with a lot of universities. >> RoomAlive is a proof of concept prototype that transforms any room into an immersive, augmented, magical gaming experience. RoomAlive uses projectors and depth cameras to cover an entire room, including the people and the furniture inside with pixels that can be used for both input and output. With RoomAlive, users can touch, shoot and dodge augmented content. That seamlessly coexist with their existing physical environment. Our system consists of multiple projector camera units or ProCams for short. Each unit contains a depth camera, a commodity wide field of view projector and it's own computational unit. These ProCam units can be used individually or combined through a scalable distributed framework to cover an entire room. The ProCam units are auto-calibrating and can self localize within the room as long as their views have some overlap. The auto-calibration requires no expertise or calibration fiducials, so the pro cams can easily be installed by end users. One just positions the pro cams in the room and the system does the rest. The system automatically creates a unified model of the room by combining the depth maps from each ProCam unit. In addition to the 3D model, our system automatically extracts the surfaces in the room, identifying vertical and horizontal surfaces and the floor plane. We expose this information together with the 3D model and the ProCam controls in the plugin to a unity game engine. This enables game designers to offer rich immersive gaming experiences. To show how RoomAlive can transform your living room, we have created four interactive experiences. RoomAlive supports procedurally texturing the living room, transforming the room into a new environment. Here the living room can be transformed into a holodeck, an indoor factory or can show a river running through the floor with dynamically generated raindrops. Virtual critters can also be procedurally generated to appear around the living room. >> So here you see actually an example, where a whole lot of other ideas have been brought together into one Interesting setup. It's a projector camera system, except they use def cameras, like connect combined to a projector, which allows them to extract much more detail, 3D information, a depth map of the scene. And then , the light is shown based on the knowledge of the 3D scene. So, and , the calibration process is very similar to what we had seen before. An interesting example of creating augmented reality experiences like this in much more of a spacious do, domain with various types of augmentations in the display space with a variety of things like, you know, colors and also objects showing up. Again, please look at the website for more details on this one.

15 - Summary.srt

So just to quickly summarize, I showed you a variety of different videos of a rather practical way of actually combining cameras and projectors together. So , we are talking about how we can actually take controlled light sources and couple them with a camera, where the camera knows what light is actually being displayed, and actually can control it to figure out what light actually should be displayed in a different types of contexts and scenes. , what we described was projector camera system, or procams. This is a growing discipline that's actually been now actually getting a lot of attention. You saw examples of something as practical as, you know, controllable headlights for cars to augmented reality experiences. Again, the intention here was to introduce to you this concept and show you a variety of examples. For further details look at these papers that I'm listing here that go through all aspects of how Light Stage works to being able to then use different types of ways of calibrating cameras and projector systems. And then also, going towards how we can actually try out the kinds of systems that we saw in some of the later videos. I showed you a whole lot of videos, and actually you can see all of them on your own, in their full detail, from these sites that actually will also be presented to you with the lectures.

# 07-03 Coded Photography.txt

01 - Introduction.srt

Welcome back. In this lecture, I'm going to talk about the concept of coded photography. Recall, again, the whole pipeline of computational photography. We have illuminated scene that goes through a series of different processes to be able to capture the light to generate an image. Now, one concept we looked at was, how do we control different aspects of this pipeline? We looked at how to control light, the illumination sources. We also want to now start thinking about how we can actually change things inside a camera and the optics and such to be able to generate newer forms of images. This leads us to the concept of coded photography. We actually have looked at some of these ideas briefly when we talked about the concepts of epsilon photography. Epsilon photography was when we said, let's take a series of pictures of something, just different in one parameter. Now, actually, let's talk about how we can create those changes in the camera itself. And the whole concept of coded photography is being able to put something in a camera that would let us capture this information right on the camera itself. And all of the variations that come in, perhaps, on things like epsilon photography, but done on-board on a camera. We're going to talk about how to modify cameras to support this in principle and actually what you're learn about is, again, the whole principle of coded photography through this lecture

02 - Lesson Objectives.srt

The specific objectives of this lesson are, one, we'll talk a little bit about the concept of Epsilon Photography, and how we can go from that concept to Coded Photography. We'll talk, really, in much more detail about the concept of Coded Photography itself. And we'll talk about coded aperture, and a flutter shutter camera, two specific types of camera that actually can do a little bit more than a standard camera. , both these cameras are examples of computational cameras that very interested building for those support computational photography. Both of these, the coded aperture camera, and the flutter shutter camera are computational cameras that are partly what we have thinking of, when we talk about controllable cameras can be used to capture images to support the computational photography pipeline.

03 - Recall Epsilon Photography.srt

Now, recall, the concept of Epsilon Photography. Something by now, you should be familiar with, because you did an assignment of trying to capture a sequence of pictures. Epsilon Photography aims to capture sequence of different pictures, where , we're changing one parameter by just a minute amount. Very small Epsilon amount to capture the variations in a scene, and then, , fuse the different pictures together, to create a richer representation that captures a scene, and variety of conditions. And that could be used to synthesize novel pictures, taking multiple captures of the single scene, or the single image to generate a newer image. We looked at a variety of examples. We can actually, change the exposure. If you recall, this was an example of what we did to create HDR images. So, by fusing the three different exposures of this image, we were able to generate an HDR image. Another example could be, the viewpoint, right? Where we change the camera parameters by changing the viewpoint, and then fusing them together, we can create a panorama. Many other examples of this exist. Here is an example of being able to capture a focus stack, and then being able to generate a new image that has no focus variations, or, , sometimes we can then controllrably change the variation. But in Epsilon Photography, the goal was to be able to capture all of those images, and in fact, actually capture multiple images. Then those multiple images could be then used to generate a newer image, or perhaps, you want a controllable image that could be generated on the fly. Now, remember, part pipeline for Computational Photography. We want to capture a 3D scene, which is illuminated, the optics focus a light onto a sensor. We can process the images to generate a new output display for a user. If you recall this from our earlier lectures, this was kind of showcased in the following manner, we had a 3D scene, we can illuminate the scene with the computer controlled projector, a controllable light source. And , we can control the light source entirely, or also the parts of it by creating some sort of, a Modular Controllable Aperture. Remember, again, we have looked at this example in the previous lecture when we talked about projector camera systems. Another option, also was to create a camera, and , we would, knew how to do cameras. We learned all there is to, about the updates and the pinhole aspect of a camera, but we can also start thinking that now I can put some sort of, a Controllable Aperture in front of this, that not only controls the amount of light that goes in. But also, which parts of the sensor will be lit based on which parts I either restrict or open. So, that's the basic pipeline that we have looked at. We've looked up projector camera systems. We have looked at how illumination source with a camera can be used to extract, and generate a newer types of images. We have looked at the camera, but we haven't actually looked at how to control different aspects of it, in this way, except, in a few examples like the dual photography example that we looked at.

04 - Coded Photography.srt

So now, using that idea, let's talk about coded photography. Coded photography is really going to get us into depths of kind to now create different ways of looking at the coded patterns. The aperture in this instance, how we can control it, and how can we embed that information to allow the camera to be able to see additional bits of information that will actually allow us to capture images. Some of this should remind you of what we did when we talked about the light field camera. Right? In the light field camera, we wanted to capture more information than just the pixel information on the sensor. We wanted to capture more information about the light field that would allow us to now generate new images. The question now is, can we actually do a lot of this, and again we touched on it a little bit when we talked about the light field cameras, by controlling this, how we can actually capture a novel image representation that would allow us to do more than a standard image would. We've looked at how to do this with projector camera systems for lighting control. So the coded exposure kind of says, is we want to be able to control light that enters a camera in time by using different types of exposure controls. Coded aperture then, actually, also impacts how much light goes in. Remember aperture is the opening, the size, we can control this. This one we controlled the amount of light goes in over time, this one we controlled the amount of light goes in by the size of the aperture. , we can control the illumination, as we know about how to do with among other things, with controllable projectors. We can also do a lot of control right on the sensor itself of the camera. An example of that, that we've actually kind of seen also is when we talked about the camera itself. We looked at the bare sensor, which was actually doing coded information. Remember, the bare sensor, , was coding how RGB was captured at the sensor and then allowed you to read it off with that code itself helping us figuring out how to get the right color at the pixel level. So coded photography is the concept, now how can we actually learn more about how we can code these things it-selves and put them on the image representation that would allow us to showcase newer forms of images. What are these examples or what we mean about that in a bit

05 - Epsilon vs Coded Photography.srt

So now before we carry on, let's spend a little bit of time comparing Epsilon photography and Coded photography. The aim so far, I've set for the concept of Coded Photography is to encode into a single image, information about the environment, the photographic signal from the environment, and then we can add a post capture, after we've captured the image decoded you actually got more information about the scene. We saw this example a little bit when we did Light Field Cameras, right? It captured the stack of pictures into one image representation, and then we were able to do thinks like do things like paralens and also change focal lens. So Coded Photography is, in that, that kind of a process where we , encode into the image itself additional information that would allow us to extract more kinds of images out of it at a later point. Something that could actually capture depth, parallax even focal planes, and perhaps, additional information. Epsilon Photography says, rather than capture one image with all of that information, let's capture a series of images, a sequence of images, a sequential set of images that may have those different variations. Take a picture, change the focal length, take another picture. That means, for example, if there was a fast moving object we would actually have trouble, because we would really like to have a much faster camera. In the case of Coded Photography, since it's doing all of that, and if it's fast enough in computing all of this, it can actually capture the image in one gulp. So that's, one of the big differences here. Now , the space that goes from Coded Photography and Epsilon Photography couldn't be merged, because we can actual combine them to be able to generate null forms of images, too. So, one thing to note about coded photography is that each image that we may capture in a Coded Photography signal would mean that the neighborhood pixels may have different radiations. One pixel would have focal lens at something, the other one may actually have it at different one, and knowing the code that relates both of them, and knowing one pixel at the left or right was captured, we can actually decode it, and generate something new and interesting. Again, an example that comes to mind is something I mentioned before, bare patterns but knowing the pattern that this one is r, this one is g, and this one is b in a square pattern, allowed us to decode the image. In Epsilon Photography , that variation is in time. Now, all of this allows us to now , create the images that can control light over time or space. Because now we can capture a series of images of one image that actually has those radiations either in space or in time, and we can preserve details about the recorded environment. So, in essence, what I'm trying to get to is there is this big space that we could actually have between Coded Photography and Epsilon Photography. Both of them are useful, and they may even overlap a little bit in terms of what they can do. These are just labels that we're coming up to help us kind of define the space of different types of photography's concepts that are related to Computational Photography. And , have both overlap between both of them, at the end of the day, we're trying to figure out how to capture the best possible way of capturing an image that we can actually render differently?

06 - Coded Photography.srt

Let me actually now start giving you more details about what coded photography could mean. Here, I'm just showing you a single image, one photograph of some bottles, some cans, and a pack of chips. If I just take this one picture, what I'd like to be able to get out of this is some sort of a depth map. Now, we've looked at how to compute depth, and we've talked about how to compute depth by taking two images, a stereo pair. And we can compute something like this. Imagine in one image, we can actually now decode it, to be able to get information that gives you depth in some detail like this. Another example we can play with is where just the same single input image can allow us to now compute a all-focused image. Everything in this, er, all layers of this image are now completely focused. Here you can see a little bit of blurriness in this image, because of defocusing. Or you know, some of this is also defocused. But here each and every part when you cans see there are multiple layers of this one all are focused. You can look at a little bit of close ups, see what I mean by this. More detail, more crispness at all levels here, this one is defocused.

07 - Lens and Defocus.srt

So that starts us wanting to rethink some of the ideas that we've had about lenses and defocus things. Let me review some of that just to kind of get us started, because then we actually going to look at how we're going to change these things. , remember that what we have and we have a lens like this. We have an aperture, the opening, right? The opening is the amount of area that actually goes though the aperture and it hits the sensor after the shutter is open. The shutter is the one that controls the amount of time, the aperture, the amount of light that enters. Now, if you recall some of the simple optic stuff we looked at this before. There's also always a sensor, a lens and the light , from the focal plane goes through. And it's for the object on the focal plane, what happens is the image is formed on the camera sensor and that's exactly we get a focused image. And , the image of the point light source, if there is a point light source at this point here where the object is we will get Something like this. And , the point spread function would just be a short line like this. So, , this is inverted. The white bright one shows this is my point spread function here for this feature. This is the optics we have looked at many times in this class. Now let's actually start thinking about what defocus does in the context of this simple lens system, which means now is I'm going to move the object at a different distance. So the same light source from here would now mean that the actual convergence of this light is happening a little bit ahead. This is where my camera sensor. So, , it's now getting to be a little bit, forming a little bit a er, before this location. And , there's now divergence showing up, which then results in a point spread function that's much thicker. And , that translates into the image of the defocused point light source would look something like this, much wider as show here. Let's move the object further back. , when we do this, you can notice that the point spread function gets wider. And , we also know that would mean that the image of the defocused point light source would look bigger. Just to kind of continue this exercise, we will actually now do last one, where we move this further away and you know, this is even wider. And , bigger blob is showing up here and it's a defocused point light source, which is point here. So we know about this kind of stuff and with that, gives us a little bit of an idea about how all of this works, something we looked at before. Another further example of this is it gets bigger. And here now, I'm showing you last stage where I move the object further. And , the point spec function is really wide and the defocus point light source is also much larger

08 - Depth and Defocus.srt

So what do these concept of depth and defocus imply here? So here , now I'm showing you this whole image. Because of the fact that things that are at different distances and the fact that you have optics, and the sensor itself, any image I take will have this artifact, right? These objects are a little bit further away, they are out of focus and this frame here, this part obviously that's in focus. So, only a few specific planes, depending on the camera and the aperture and stuff like that would be in focus. So this is in essence is the artifact of a traditional camera system. We will always have scenes at different depths. All of them will have different focus. And , the question comes up is can you actually compute the depth just from knowing this focus planes that are different in this image. So depth from defocus is attempts to infer depth by analyzing the local scale. That is, how is, much is the dif, difference in the frame itself or difference in the focus itself in local parts of the image, and use that to compute a defocus blur. And if I actually can figure out the defocus blur, I can actually kind of claim knowing the lens that I have, how far a certain object is. That's what you saw in the last example. The more I moved the object, I got a different blur out of it. Now, if I could actually create a relationship between this, I could actually start computing depth. We saw examples of that when we talked about computing depth from images in the lecture on stereo. But the bottom line is, this is an extremely difficult and ill-posed problem. because we actually have a tougher time of actually figuring out the calibration that will allow us to figure out how much defocus blur exists in different parts of the image, that will allow us to then actually compute how far those objects are in that image.

09 - Depth and Defocus Challenges.srt

So let's look at some of the challenges of computing depth from defocus. One, it's extremely hard to discriminate a scene where there is smooth information or there is blur information, specifically defocus blur. And, , once we've figured out if there is defocus blur, it's hard to actually undo it. So here again is example, how do we know this is out of focus? Again, by looking at it we can kind of see it's blurry. And how do we know what kind of blur it is? Remember when we talked about cameras, blurry images can be caused by a variety of reasons, this could have been a blur because the camera shook. How do we know it's actually a blur because of it's out of focus, or the focus plane is in front of or behind it. Another part of it is, how to get rid of blur. Well, there is a vast amount of literature in image processing. So if somebody gave you an image like this, you can actually run an algorithm, specifically the Lucy-Richardson Algorithm, which again has been know since 1972, Which would actually deconvolve this image to be able to generate a sharper, or attempt to create a sharper image. But , if you notice here there is an artifact called ringing, you can see different types of things across the five and the four. This is an artifact that would come in because we're trying to take this information and use that to sharpen an image. And after certain kinds of sharpening effects you'll actually get artifacts like this. Part of the reason is, again, if you have an original image that's blurry like this, you really can't completely generate something out of nothing. There's no information, you can't create it. One of the mantras always used in computational photography is, what you see is all you get. That reduces the space of what you can do with an image. If you actually have a bad image there are other ways you can try to perfect it. And we'll actually talk about those types of things, too.

10 - Possible Approaches.srt

Now how do we actually compute these types of things for images. How can we actually deblur imagines that are actually because of focus. So here are two different sets of examples. These are actually approaches proposed by Levin et al., in 2007. , the two methods they proposed as one exploit, prior information on natural images. Look and model the types of signal deconvolutions that happen from natural images like this. And actually use that to figure out depth discontinuities and discrimination that will allow us to improve deconvolution and also depth estimation. So, you know, natural images or scenes like this, unnatural images would be, you know, random noise types of things. The other method, and this is the one we will actually focus on in this lecture, is come up with different patterns and that use these patterns on the lens itself. So make a defocus pattern that is known. In this case, it's the aperture itself. But change the defocus pattern to have some sort of a characteristic like this and use that discriminate where the defocus is differently for different planes of defocus planes in an image. Again, we'll talk about this in lot more detail. And the idea really comes up is that what we want to do is come up with a coded aperture. An aperture that would have different characteristics like this, and you can put this inside a lens before the image is formed, the amount of light that goes in through the lens, through the sensor. If you can put this in, we can actually perhaps, quantify the defocus on the sensor. And that will actually let us know how to deconvolve it based on the fact that we have actually quantified what the focus plane would be.

11 - Defocus as a local convolution.srt

Now, let's actually try to quantify all of this. Imagine I have an input image which has been defocused, and what we want to do is now look at. Remember the, all those defocused regions that we'd looked at, calibrate these blur kernels at different depths. If can calibrate them, perhaps we can now model the response on an image because of different types, you know, different blur kernels that are coming in. Because again, these are response functions we saw when we moved objects around. If you actually went through an exercise of doing that, what we did of trying to move an object. We might be able to kind of build a model that could be used to generate this information. So, let's look at this a little bit more carefully. So imagine, I could now come up with some sort of a local sub-window and also look at how the calibrated blur kernel for that image would look like at certain depth k. That would actually be fk, so, I have a window, yk, I have a new calibrated blur at different depths k. And we call that fk. And , now, we can come up with a sharp output x on that sub window, by doing what we know how to do which is convolutions, right? So, it means is for a specific depth, I can now come up with different types of images. So, k1, k2 and k3. So, I'm just showing you a simple table here. This is my image, I found different sub-windows. And I can now compute if I know the blur kernel. And, these are again, if you noticed the ones that we had looked at as from the example of images earlier. When we moved the lens moved the object, the light source away from the lens. We've got different spread functions out of it. We can actually use this to now, , compute sharp images. And , no surprise to you how we do this is a simple convolution of the calibrated blur kernel. With the sharp window here to be able to create a local sub-window. Right? So this is how we would actually do it. This is a forward process, if I had this, I would actually be able to compute it. , my interest is in actually computing this, right? I want to be able to actually come up with a sharps sub-window. So the inverse of this process is what we are interested in. But, we can actually do calibration this way.

12 - Coded Aperture.srt

But the idea that I want a test to kind of get through. And this was actually an idea proposed by Levin et al in their paper. Which is also referenced at the end of this lecture and will be available for you to look. Is how can you actually create a coded aperture. So this is my conventional aperture, right? Just the opening. I know the size and all the amount of light that it goes through. Can we actually create a mask? Referred to as a code here and the aperture plane that makes the defocus planes different when they actually register on the sensor. And if they're different they might be easier to discriminate. So , that suggests us, we want to actually add a coded aperture, it , lets in less light, but in a known pattern like this. What's the artifact of this? Let's look at exactly, the example we looked at before of our lens and defocus. Remember, we had looked at this before. We , had an object moved up. And, , it created these patterns from a single light source. So here, , we have an image of a defocused point light source. , when we replace it, this will look different. Right? because now less light will come in. , that says, how do we kind of build this as we now create obstructions on the lens. Like I'm just going to show you one simple example here. , how I'm going to represent these things are simple obstructions shown here. So , this line here is eh, maybe considered to be one of these things. And I'm going to say, if I put an obstruction like this on the lens, means is now this part of the light is not making it to my sensor. And, , that translates into a spread function that would be different. Similarly, I would actually put another one just to kind of see what the effect is. that also goes in there. And translation is that, , now I have a point spread function that also has a different look. You know, maybe like this. This was the output based on this. Again, notice this is what we're trying to do here. The same object far away putting simple to obstructions, we get a newer image or a newer output. But we can now start calibrating this. Recall again, we did this kind of same trick when we were talking about light field cameras. Again, simple example, further, same obstructions. I moved the object closer. Spread function is different. , now I can start calibrating for these types of things as I move closer and closer you can see the end of it. Doesn't matter what it is. If I actually move the light source here, the spread function will come back to what I expected for a regular aperture, because this, everything is in focus. , the final one, when the object is on the focal plane. , the image is created here. Our spread function is there. And while there might be a small impact of the obstructions here for full-focus thing, you'll actually get a spread you know, the image to be much smaller, and a point spread function, but also be like the one we'd expected in case where there was no obstructions.

13 - Benefits of Coded Aperture.srt

Let's analyze some of the benefits of coded aperture here. , we looked at variety of different you know, types of kernels here, larger scale, correct scale, and a smaller scale. And, , we have the coded apertures, , also scaled down. If you had an image like this to be reconstructed on a conventional the, again, Lucy Richardson types of deconvolution method, you'd get a lot of ringing. , if we do it at the correct scale so the size of the kernel matches the kind of signal that's in the original image, you'll do much better. But , if you do it a smaller scale, you actually will add other forms of blur. , the secret is to figure out which of these scales to do this kind of stuff. We've looked at that concept, and we looked all the frequency, domain, fading, and blending stuff. But in case of coded, we have an artifact that we might actually generate something like this, but we actually do get a much better image at the right scale. And , a little bit of blurriness and ringing at the scale that's smaller. I do loo, encourage you to look at the Levinthal paper, that'll gives you more detail about this kind of stuff and how to choose the right scale.

14 - Aperture Occluder.srt

Before we go on I just wanted to have you, kind of, think about one specific thing, and that is that can we determine what kind of pattern gives you the most discriminating information between images at different defocus scales. So, here are the four different types sample aperture patterns and a conventional aperture just simplified to look like this. Now, the score is the which one of them is more discriminative. Remember again, when moving the object away the pattern changes at the, sensor and each one of them is going to give you more detail about how far things are, you know, and also that could be useful figuring out defocus planes and depth. I'm just putting in you, know you, five different examples on the score of which one is more discriminative versus less discriminative. So what I'd like for you to do here is, just put in a number. Which one is the most discriminative, which one is the least discriminative? What would help you to look at is, again, a little bit more on the structure of these obstruction patterns. symmetry is there in most of them, but kind of starts giving you a little bit of x and y, but at the same time, you also want to start looking at, you know, what are the variations in, the diagonals and stuff.

15 - Aperture Occluder.srt

The answer for this should, maybe was obvious, and that is, a pattern like this, which has more circular information going out from the middle. Asymmetries and stuff like that would be quite discriminative, and in fact, when they did the experiments in that paper, they found this to be the most discriminative. This actually came out pretty good too, and this is decent because again, it has more information, but this one actually came out more. But, , a conventional aperture came out as the worst. This could have been a little bit of a toss up, this one was actually a really good aperture design. in that work, they came up with various types of metrics to help you figure out the best possible design too

16 - Depth Estimation.srt

So what can we do with a sensor like this? Let's first talk about depth estimation. Using this sensor and this kind of a coded aperture, they were able to compute depth in one single image just using this coded aperture. This is the output they would get. Not perfect, but this is what you get out of a single image. And then they actually came up with an algorithm that would regularize the depth to able to generate an image like this. I encourage you to look at the concept of regularization, but among other things, it is one of the best ways of smoothing out the information. And that is the way that this kind of detail were kind of gotten rid of. And allow you to kind of generate a much better mask for depth. So , if you look at it after doing regularization you can actually get a much better sense of depth. In this paper they also kind of suggest that the sometimes, by simple manual intervention, you can also improve results. Another example of this kind of stuff, again where you have regularized depth here. Some of the stuff is lost, but again, from a simple camera, you can actually do quite well with these types of things. And here, by just adding a little bit of corrections manually you can actually now get the shapes that you wanted, or the depth that you wanted. And these are the gestures that were actually acquired to help you do this. Let's look at a few exam, other examples. This is an example of focus corrections. So, , this is an input and you can see a lot of focusing, you know, and defocusing artifacts here. it's not completely all defocused here. But, with this method, now we know the things we can actually use this to deconvolve the thing, and now the whole image is completely in focus. Another example to just look at this is, this was the original image and if you just zoom in, in this output image. the coded image captured all of the details, allowing you to after the fact, in a post-editing mode, be able to now actually visualize a focused image. Another close-up is this example here, when given an input image of two people. One you can see is badly focused, and , now the whole image has been focused correctly. Again, the camera captured all the information and you can do this after the fact. And, the example again is the original image had a lot of focus blur, and you can see a much sharper image as the output. , we could do some sort of you know, naive sharpening and stuff like that and get simple examples like this on this image there are methods out there. But again, the point being this was done on the camera itself with the coded aperture.

17 - Comparison with Conventional Aperture.srt

Just now let's look at a bunch of different examples. Here, doing a comparison to conventional aperture corrections, a conventional aperture like this to use for doing corrections. You see a little bit of ringing, that's shown here. You see a lot of defocusing, you can see a little bit of ringing here also. This is the output with this aperture, much crisper image. Here's an example very similar to what we looked at when we did the light fieldwork of being able to de-focus, focus an image in a single image itself. You can see , interactively, we can focus one of, one of the subjects and move around and be able to focus on the other one. Again, this is coded aperture. We used to do these types of captioning or hold in, information on one image to be able to allow you to do this. Refocusing from a single image in this example. Hopefully, you can see this on your compressed videos. Now actually, I'm going to borrow a specific example on how these authors generated this coded aperture. That's it, they did was, open up a lens, I don't recommend doing that to your favorite high end expensive lens by yourselves. But open it up and put that small aperture in there, and then seal the camera back. And when you put the camera the lens back on the camera you have a coded aperture. Again, do this at your own peril. I don't recommend taking off and breaking up your very, very favorite lens to do this kind of stuff.

18 - Coded Aperture Benefits and Drawbacks.srt

Let's look at different types of aspects and benefits of this coded aperture. One, , it allows you to compute, capture an image in depth, just from a single image. And there is no real loss of image resolution for a simple modification to a lens. Again, I don't recommend doing to a very, your favorite lens. Depth is coarse, and we saw examples of that, I mean, again, we are trying to get something out of nothing. If you get a little bit of depth, but not very much, that is because we are trying to get this out of, by creating focus planes by actually changing how the focus looks like differently because of the coded aperture. But, you know, we get depth, , we loose light and we know that, right? Because we have a smaller opening than this one, that means that the light sensitivity in your lens of the camera is going to go down, darker scenes won't work very well. But again, we can actually do deconvolution and increase depth of field after the fact from a single image. So, in essence, this is like a light field camera where we actually get defocusing and depth information by changing in something in front of the lens itself, the opening or on the plane of the lens, rather than changing the sensor, because in light field, that's what we did. Light field camera we changed the sensors to be able to capture different types of lights.

19 - Flutter Shutter Camera.srt

Now before we finish, I want to actually now showcase you, yet another example of similar coded photography. Except this time, we're going to talk about the concept of a flutter shutter camera. The flutter shutter camera, now attempts to do the same thing the coded camera did, except that this time around rather than change the aperture, it wants to change something about the shutter. In coded aperture, we added obstructions to the aperture. In flutter shutter, let's control when the shutter opens and closes. So, you know, again have the whole thing and just figure out how to open and close it in a specific manner. Just to kind of look at this, let's look at this series of examples. A standard camera or traditional camera, when it takes picture, when the shutter is open, you get one frame. Right? And these are ideas come from this Raskar et al 2006 paper that's represented at the end and also we'll provide you the links where to look at in detail. I do encourage you to look at these papers to kind of get a good sense of the exact technical details. I'm just summarizing them and giving you the big overview of these concepts here. So, , when I open the shutter like this. And if the car is moving, what I would get is the car is moved and if I actually have the shutter, you'd actually get some sort of blur. A motion blur in this instance, right? Let's look at it again. When I open the shutter, the car is moving. And , depending how long the shutter is open, I will get a blurry image like this. Well, let's actually do something interesting. Let's actually change how the shutter opens and also control it when it closes. So now what we'll do is we'll open and close the shutter and this is showing you this. Now, if I know exactly how this opens and closes, this is interesting. Now, I do get a motion blur image. But if you notice this one has a very specific sharp lines. Let's look at them next to each other, this was when the shutter was open the whole time and this was when we kind of flutter shuttered it, that is opened and closed it at different rates. Here everything is blurred. In this one, you do see some blur. But actually, there is a little bit of a better signature. So you, for example, see these lines at different you know, there's no, , detail here whatsoever. But here, you actually see different characteristic lines on this image. This actually is much more useful than this. And if I actually know this code, then I actually know a lot more about this image too and that's the basic idea. If I know the code on how and when I open this, I should be able to kind of use this information to model the output that I'm getting after all of this is accumulated or integrated. Here I know all of it, but since I have no control over how the aspects of what happened in the middle everything is blurred. So Raskar et al actually designed this camera by actually just taking an existing camera and putting some sort of a additional controller that would change the control of how the shutter opens and closes with a digital process here. So they actually, could actually figure out the exact code on how the, the shutter would open. So they created this new camera to do this and they did a lot of other experiments with it. Again, I recommend you to look at the paper for more details.

20 - Traditional Camera Box Filter.srt

Now let's look at what we can do with a camera like this. So, again, we know that if somebody gave me this information I would have a camera. And what I'm doing is, to create this burn image, remember our lectures on doing things like convolution. If I'm convolving with a stock function like this, blurring is equivalent to convolution and this would be my output. The size of this kind of suggests the size of the step function that we, you would have. And what comes down, there's a concept of a sinc function, which says that. And sinc function for those of you curious, is showing the information about a signal. And it says a sinc function is a general form here and is a sine wave that decays with an amplitude 1 over x. So as we move away, it gets smaller and smaller. This is one of the standard ways of actually computing discrete Fourier transforms and it's one of the pairs that captures the magnitude of a signal like this. So this is, , one of the sinc functions for a signal like this.

21 - Flutter Shutter Coded Filter.srt

When we move to a signal, which is now pulsed or flutter shuttered like this, we might have a signal like this. , the output we know is convolution of this to this one, and this is also a sinc function. And , the big thing here is it is actually preserving all of the high frequencies. It's not going down into the low frequencies. By being able to do this, we actually now are able to get rid of modeling all of the low frequencies and high frequencies are there. And this is my sinc function. Again, I would get this by trying to compute the discrete Fourier transform of an image in this process. Recall again, that convolution has a lot to do with Fourier transforms, when we looked at all of our signal processing stuff in one of the earlier lectures. Here are the two signals next to each other. Full open shutter, sinc function looks like this. And , our flutter shutter, or a pulse shutter like this, the since function looks like this, never gets to some of the low frequencies always . And again you can see this that there is a lot more low frequency information here. And because of the sharp lines and stuff like that, here we are actually seeing a lot more of the high frequencies. Simply, again, we flutter shuttered or pulsed our sense, our, our shutter to be able to get this image. We are interest in, , is the inversion, right? We are trying to invert this, being able to go from this to an original image. , we can do this by doing inverse Fourier transforms. in this case, we can now look at the inverse filters for doing these types of things, and we look at the two different sine functions. The thing that is, and this is something I'm going to refer you to back to some of the literature on signal processing. This kind of a filter is unstable, while a signal like this is much more stable because again it's only kind of capturing the high or low frequencies in this instance. And we can represent this much better to be able to kind of do the inversion to compute our signals. And that's the basic idea here is that trying to do an inverse discrete Fourier transform, this kind actually would be much more suitable for our reconstruction. Let's look at the examples. In this case I'm going to show short exposure, long exposure and coded exposure. For a long, short exposure, , no light. We get a ray thing, but we get a nice, crisp image. Long exposure, we get a lot of blurriness. Coded exposure we get blurriness with a lot more high frequency information still there. And , we can de-blur this using the approach I kind of talked about, and this is what we would get, much crisper image. If we were to de-blurring this with a variety of different approaches, we might still get some incompleteness. But in this case by doing it the method I was doing with discrete Fourier transforms and stuff like that, we will get more bars. By the way, just to compare, this is the ground truth image that we would have had if we actually had captured it correctly. And this one looks pretty good. There's a little bit of noise, and a little bit of banding going on, but much better than this. And that's the basic idea of this flutter shutter camera. Being able to kind of use this knowledge to be able to now reconstruct an image that knows how the variations happen. Again, we know the code, we know the output, we can deconvolve with the knowledge itself to help us figure out how to actually generate a reconstruction that's much more true to what the original image would be. Same idea as what we looked at for coded aperture, except this time it's a coded shutter. Let's see some examples produced by these authors and their paper. Here is an example using this camera here, of looking at a car. , if this car is moving fast, you cannot actually see the license plate, which is here. But, , if we had a flutter shutter camera, you should be able to reconstruct from this and actually get the license plate number. Another example of the same thing again, somebody says I want more detail here. This is the original image, but if I had a flutter shutter camera, I should be able to generate more. Another example with the cars again, this is the orig image, image. Let's just focus in in there. And actually, the whole car image can be regenerated. And if you notice, we can actually now start looking up, and start even reading some of the numbers and additional detail. Another input image, we can rectify the cropped image here and again create a de-blurred result, much more crisper.

22 - Different Codes.srt

So finally, let's look at a little bit more detail about these types of codes. So this was our all open shutter, the sinc function looked like this. I captured both low and high frequencies. , we can just alternate the shutter, not in a random matter at all, but just much more in a time manner like this. A sinc function also still has low frequencies and high frequencies, it just has a different our, you know, representation doesn't help us much. Random, we can just run it randomly. It can, that would allow us to do this, but again, would have more variations, would have more peaks and lows and stuff like that. What these authors found is some sort of a flutter starter signal like this, which went through small, sharp, small, large, small, large kinds of things. Gave a sinc function like this that captured all of the high frequencies, and that was much more amenable to the kind of stuff they wanted to do. Again, the analysis of this exists in the paper, and I encourage you to look at it.

23 - Summary.srt

So with that I'd just like to close and say, one, we talked about Epsilon photography. Sequence of images allowed us to now capture more additional representation that will allow us to generate newer images. We said, let's look at coded photography, that is, within the same image captured, let's put some sort of additional information in there that could be done by coded aperture or coded shutter and that allow us to create, for example, two different types of images. We looked at coded aperture in detail. We looked at flutter-shutter, which was called a shutter. And all of them are examples of how we can actually do coded photography. Again, Epsilon photography, coded photography, just kinds of similar things in different ways, but all aimed at generating novel images on the camera by controlling how the camera captures light from a scene and in essence are major parts of what we think computational photography should be looking more and more at. You'll be learning more about these things and other ways throughout the class with some additional papers that I'm going to be pointing out to you. To finish, here is the paper that actually gets into the whole concept of Epsilon to coded photography, then the paper that I talked about, coded apertures. This is the paper about fluttered shutter cameras. Just if you're interested in the Lucy Richardson deconvolution methods there's a Wikipedia entry on it that's pretty good. for those of you interested in learning more about the discrete Fourier transform and the sync functions for different types of signals. There's an online book that has a lot of in, information on it. But otherwise find your favorite textbook on digital processing or image processing, they'll tell you how to do these things. Just again, I've also linked the websites of the different projects that I've talked about in this class. I encourage you to look at them, there's a whole lot of additional detail available.

07-04 And Remember.txt

1 - Have fun computing with photographs.srt

All right, we're almost at the end now. Hopefully you've had a lot of fun and interesting experiences playing around with Photocraft and computation at the same time. We've actually had a lot of interesting lectures. Hopefully you've enjoyed all of the material we've covered so far and all of the fun assignments that you've gone through to learn about the building blocks of computational photography. While the lecture part of this class is over now, I actually do want to say that I'm going to be now adding on Piazza and other types of sites, a variety of additional advanced concepts. These will be in forms of research papers, websites and videos. I do want you to look at all of this material, study it and engage with me and the rest of the class in a variety of conversations on what you think about these advanced concepts. These are concepts that could be practical to the kinds of things that you've seen in existing software packages that you use, to much more researchy, kind of pointing a direction towards future of computational photography. I'm going to try to come up with a variety of interesting topics. Again, my request is, engage with this material on Piazza, chat with each other. We're going to have a lot of fun trying to play around with a variety of topics in this area. But again, it's new advanced concepts that I think you'll enjoy. With that, I'd just like to say thank you, it's been a lot of fun and remember, have fun computing with photographs.