Python And OpenCV coast in this course will be talking about everything you need to know to get started with OpenCV and Python we're gonna start up with very basics, that is reading images and video, manipulating news media files with image transformations, and how to draw shapes and put text on those files. Then we're gonna move on to the modem. Bronze part of open CV that is switching between colour spaces, bitwise operators, masking, histograms, edge detection and thresholding and family to somethings up will be talk about face detection and face recognition in open CV. So how to detect and find faces in an image and how to recognise them? Using inbuilt methods in the last video will be building a deep computer version model to classify between the characters in The Simpsons. Best off. Some images of material discussed will be available on my guitar PG and all relevant links will be put up in the description below. So that sounds exciting. Don't forget ahead. River and subscribe to my channel and I'll see you in the coast. Everybody, and welcome to this Python And open TV coast. Over the next couple of videos we're going to talk about using the Open TV library to perform all sorts of image and video related processing and manipulations. Now I won't be delving into what open TV really is, but just be brief. It is a computer vision library that is. Available in Python, C Plus plus in Java. A computer vision is an application of deep learning. They are primarily focuses on driving insights from media files, that is images and video. Now I'm going to assume that you already have Python installed in your system and going to check this is bye going to terminal and typing Python Dash Dash version. Now make sure you're running a version of Python. Of at least 3.7 above. Whatever we do in this coast, one really work in some all the versions of Python And especially pipe on 2. So just make sure that you have the latest version installed. Go ahead to pipe on dialogue. Download the latest version from there. Now assuming that you've done this, we can proceed to installing the packages that we require in this course. First one is open C so go ahead and do a pip install open CV Dash contrib Python. Now sometimes you may find people telling you to install just open CV Dash Python. Well this open team Dash Python is basically the mean package, the main module. Of open CB or BCD dash control Dash Python includes everything in the main module as well as a contribution modules provided by the community. So this is something I recommend you install as it includes all of open C vs functionality. You may also notice that Urban TV trying to install the NUM PY package now known place kind of a scientific computing packaging. Python they have extensively used in matrix and array manipulations, transformations, reshaping, any things like that. Now will be using one point sum of the videos in this coast. But don't worry if you've never used it before, it's simple and relatively easy to get started with. Now the next package I like you to install is here, so go ahead and do a PIP. Install Sierra. Now slide disclaimer. This is a package that I built to basically help you to speed up your workflow. Sierra is basically it said utility functions that will prove super useful to you in your computer vision journey. It has a town of super useful helper functions that will help speed up your workflow. Now, although we're not gonna be using this for a good part of this course. In fact, we only begin to use this and all last video of viscose when we're building a deep computer vision model. I recommend you install it. How so that you dont have to worry about the installation process later on. If you're interested in contributing to this package or just simply want to explode the code base, I leave a link to this. Get a page in the description bored. Okay, that's it for this video. In the next video will be talk about how to read images and video in urgency. So I'll see you guys in the next video. Everybody and welcome back to another video. In this video we're gonna be talk about how to read images and video in open CV. So I have a bunch of images in this photos folder and a couple of videos in this videos folder. In the first half of this video will be talk about how to read in images in open CV and towards the end will be actually talk about how to read in videos. So let's start up by creating a new file in. Call is read it. And the first thing we have to do is actually impose CV 2 and CV. So the way we read in images in open CV is by making use of deceived I'm read method. Now this method basically takes in a path to an image and return that image as a major or pixels. Specifically were gonna be trying to read this image of a cat here. So we're gonna say photos slash cat done jpg. And we're gonna capture this image in a variable call IMG. Yeah, you can also provide absolute paths, but since its photos folder is inside my current working directory, I'm going to reference those images relatively. And once we have read in our image, we can actually display this image by using seaweed. I'm show method. Now this method basically displays the image as a new window. So the 2 parameters we need to pass into this method is actually the name of the window. In this case it's gonna be cat and the actual matrix of pixels. 2 display which in this case is IMG and before we actually move ahead I do want to add an additional line ASI we don't wait Kee Zero. Now the CEO Wiki Zero is basically a keyboard binding function. It waits for a specific delay or time in milliseconds 4A key to be pressed. So if you pay send zero it basically you wait for an infinite amount of time for a keyboard key to be pressed. I don't worry too much about this. It's not really that important for this course, but we will be discussing some part of it towards the end of this video. So lets actually save this end wrong. I'm saying Python read the images displayed in a new window. Cool. Yeah this was a small image. This was an image of a size 640 by 427. I'm gonna try and read in this image of the same cat but a much larger version 2400 by 1600 image. So we gonna say kept on a school lunch.jpg lets see that and run and as you can see this image goes way of screen. The reason for this is because the dimensions of this image were found greater than the dimensions of the monitor that I'm currently working on. Now currently open CV does not have an inbuilt way of dealing with images that are for greater than your computer screen. There are ways to mitigate this issue and we will be discussing them in the next video when we talk about resizing, rescaling frames and images. But now just note that if you have images, if you have large images, it's possibly going to go off screen. So thats it for reading images. We cannot move on to reading videos in open CV. So it's cold reading videos. So what we gonna do is actually going to read in. This video of a dog. And the way we read in videos is buy actually creating a capture variable and setting this equal to seaweed video capture. Now this method either takes in integer arguments like 0123 etc or a path to a video file. Now you will provide an integer argument like 012 and 3 if you are using your webcam or a camera that is connected to your computer. In most cases, your webcam would be referenced by using the integer zero. But if you have multiple cameras connected to your computer, you could reference them by using the appropriate argument. For example, zero would reference your webcam, one word reference the first camera that is connected to computer 2, word reference second camera and so on. But in this video we'll be actually looking at how to read in already existing videos from a file path. Yeah, we specifically relbe reading this dog this video of a dog here and the way we do that is by providing the path. So videos dog Dot MP 4. Here's where reading videos is kind like different from reading images. In the case of reading and videos, we actually use a while loop and read the video frame by frame, so we're going to say while true. And the first thing we want to do inside this loop, say is true and frame is equal to capture dot read. Now this captured read basically reason this video frame by frame. It returns the frame and Boolean that says whether the frame was successfully reading or not. Do you display this video? We can actually display an individual frame, so we do this by saying I'm show. And we call this video and we pass in the frame. And finally for some way to stop the video from playing in. Definitely it was saying if see, we don't. Wheat Key 20 and zero xff is equal to equal to or of B and we want to break out of this while loop and once that's done we can actually release the capture pointer. And we can destroy all Windows. And we can get rid of this. So basically just try camp. The capture variable is an instance of this video capture class. Inside a while loop, we grab the video frame by frame. By utilizing the captured read method we display each frame of the video by using the AM show method. And finally for some way to breakout of this while loop we say if she if you don't wake E 20 if. And zero xff is equal to order D, which basically says that if the letter D expressed, then break out of this loop and stop displaying the video. And finally we release the capture device and we destroy all the windows since we don't need them anymore. So lets say that in run. And we get a video displayed in window like this. But once it's done, you will notice that the video sunny stops and you get this error, most specifically a -215 assertion failed error. Now if you ever getting error like this -215 assertion failed, this would mean in almost all cases is that. Open CV could not find a media file at that particular location that you specified. Now the reason why it happened in a video is because the video ran out of frames. OpenCV could not find any more frames after the last frame in this video. So it unexpectedly broke out of the while loop by itself by. Raising a CV 2 error. And are you the same error if we? Kamala this out we can comment this out and we specify our own path to this image. So lets see we don't wait. Wait, Ki? Zero saver and run. And we get the exact same error. This basically again says that open CV cannot find the image or the video frame add a particular location. Basically it could not be read it. That's what it's saying. So that's pretty much it for this video. We talk about how to read in images in urban savyon, how to read in videos using the video capture class. In the next video will be talk about how to rescale and resize images and video frames in open CV to see you then. Everyone and welcome back in this video we're gonna be talk about how to resize and rescale images and video frames in open CV. Now, we usually resize and rescale video files and images to prevent computational strength. Large media files tend to Stow a lot of information in it, and displaying it takes up a lot of processing needs that your computer needs to be sign. So by resizing, rescaling we are actually trying to get rid of some of that. Information. Rescaling a video implies modifying its height and width to a particular height and width. Generally it's always best practice to downscale or change the width and height of your video files to a smaller value than the original dimensions. The reason for this is because, well, most cameras, your webcam included, do not support going higher than its maximum capability. So for example if your camera shoot. 720 P chances are it's not gonna be able to shoot in 1080 P or higher. So to rescale a video frame or an image we can create a function called deaf rescale friend and we can pass in the frame to be researched and scale value which. D folder gonna said as 0.75. So what I'm going to do next is I'm going to say with is equal to frame dog shape of one of one times. Scale. And I want to copy this. And do the same thing for the heights. Remember premiership of one is basically the width of your frame or your image and frame. Note shape of zero is basically the height of the image. Now since with in hyd are integers, I can actually convert these floating point values to an integer are converting into a 90. And we're gonna be doing is we're going to create a variable call dimensions in said this equal to A to bull of width, height. And we can actually return. See we dont resize. The frame dimensions and we can pass in interpolations of TV dot inter on the school area. Now we talk about TV to resize in upcoming video, but then I will just note that it resize iframe to a particular dimension. So thats all a function does. It takes in the frame and scales that frame by a particular scale value, which by default is 0.75. So lets actually try to see this in action. Let's go back to this reader PY and grab this code. Now what I'm going to do is after that I read in the frame, I'm gonna create a new phone call frame on the scroll resized and set this equal to. Rescale frame of frame and lets leave the scale values 0.75 and we can actually display this. Video resized. By passing in the frame\_resized. Resize. So lets say that in run Python re scale. Top keyboard. It was a narrow. Okay, we dont need this. Let's closeout. Say that and run. This was are originel video and this is actually a resized video with video resize by 0.7575%. We can modify this. By changing the scale value 2. Cheers maybe? 0.2 so we rescaling into 20%. Am I getting even smaller video in a new window? Chillers closer. Now you can also apply this only images so that. Hement that out, change that to can't turn JPG. MV can you received that I'm show. Image in pass in Hindi. Image. And we can create a resize image. I calling rescale dream and we get pass in the IMG. Sasidhar in Rome. And this is a small videos when a concern with that this is actually the big image large image and this is the research version of this image so thats close error. Now there is another way of rescaling or resizing video frame specifically and thats why actually using the capture dot set method. Now this is specifically for videos and work for images, so let's go ahead and try to do that. Let school this death change rest. So what changes are changing the resolution of the image of the video? We can pasina with in hyd. What to do is going to say capture don't set 3. With them are going to do the same thing with captured DOT set full height. Nothing for basically stand for the properties of this capture class. So 3 references the width and 4 references for height. You can also expand this too, maybe change the brightness in the image, and I think you can reference that by sending this to 10, but now we're gonna be interested in the width and height. Now I do want to point out this. This method will work for images, videos and live video, basically everything. You can use this rescale frame method, but the changes function only works for live video that is video reading from. Download camera or your webcam for instance. So video that is going on currently. This is not going to work on stand alone video files, video files that are already exist. Just doesn't walk. So youre trying to change the resolution of live video with then go with this function. If you're trying to change the resolution of an already existing video, then go with this function. So that's pretty much it to this video. So we talk about how to resize an rescale video, friends and images in open CV. In the next video we talk about how to draw shapes and write text on an image so that everything. I'll see you guys in the next video. Paper one and welcome back to another video. In this video we're gonna be talking bout how to draw and write on images. So go ahead and create a new file and call this draw PY. We're going to import CD 2 SSV are going to import the NUM PY package that opens if I had installed previously. I'm gonna input that as an P. We will read in an image by saying I'm G is equal to see I'm read. Pass in photos. Feroz slash cat.jpg in. Display that image in a new window. And we can do received, wait Kee Zero. Now there are 2 ways we can draw on images. I actually drawing on stand alone images like this image of a cat ear or we can create a dummy image or blank image to work with. And the way in which we can create a blank image is by saying blink is equal to zero. Of shape 500 by 500 and give it a data type of UNTH. Uidai is basically an image, the data type of an image. So if you want to try and see this image, save this image looks like. We can see your blank and we can pass in black. Say that and run. Python draw PY. And this is basically the blank image that you can draw on. So we're gonna be using that in instead of drawing on this cat image. Feel free to use this category like. So the first thing we are going to do is trying to paint. They tried to print the image certain colour. We do this is by saying blank and reference all the pixels and set this equal to 02550. So by pending the entire image green and we can display this image that saying green impasse in the blank image. Say that and run. Can I broke as? Yeah okay. You need to give it a shape off 3. Basically we are giving a shape of height, width and the number of colour channels. So just keep that in mind. Say that and this is the green image that we get. Cool we can I can change this Ain trying to change this to read. Zero, 255 say that. Can we get a red image over here? Now you can also colour a certain portion of the image by basically giving it a range of pixels. So we can say 200 to 300 and broom 300 to 400. Say that I'm drunk. And you got a Red Square in this image. The next time we're gonna do is we're going to draw a rectangle. And we do this is by using the we dont rectangle. Method of this method takes in an image to draw a rectangle over, which in this case is blank and it takes in 0.1 0.2 colour, thickness and online type if you like. So the 0.1 will specifically by zero zero which is the origin and we can go all the way across 2250, 250. Let's give the colour of zero, 245, zero which is green. Give it a thickness let's say 2 which is basically same the thickness of the borders. Animal sex down we can display this image is saying. School district angle. Impossible. It has in the blank image. We can comment this out since we dont need this anymore. Can we get a green rectangle that goes away from the origin to 250 colour 250? Can you play run with it if you like? So we could go from 252, maybe 500? Aron goes all the way across the image, so you basically divide the image in half. Now there is a way of filling in this image a certain colour and the way we do this is instead of saying thickness is equal to 2, we say thickness is equal to CB dot build. And that basically fills in the rectangle to get this green rectangle. Now alternatively you can also specify this as -1, -1. Can we get the same result? What we could also do is instead of giving it fixed values like 250 and 500 what we can do is we could say IMG don't shape of zero. Of one divided by divided by 2 and image don't shape of zero. Divided by divided by 2. Let's say that and rum. Images of Accor this is blank in this blank see that are wrong. Can we get a nice little rectangle square, if you will, in this image? What are basically did is its killed the rectangle from instead of being these this entire square. This rectangle basically has dimensions half of that of the original image. So, moving on, let's draw and draw a circle. Draw circle. This is also fairly straightforward. We do a CV dont circle. Can we pass in the blank image and we given a center which is basically the coordinates of the centre? For now let's set this to the midpoint of this image by saying 250 common 250. Alternatively, you could also get this. Let's give it a radius of 40 pixels, give the colour of zero, 0255 which is read BGR and give the thickness of let's a 3. Weekend display this image. Say circle is equal to black. We get a nice little circle over here that has its center at 2:50, 250 and radius of 40 pixels. Again, you could also fill in this image by giving a thickness of -1. Can we get a nice little dot here in the middle? Cool. Now there's something else that I forgot and that is how to draw a line as standalone line on the image. That again is fairly straightforward SADR Online. We use a CV dot in line method and this text though the image to draw the line on and 2 points. That's just copy these. Points a square everything. Alice, basically you draw the point from 002 half these image dimensions, so thats 25250 and then it draws. Align of colour 0255, zero let's set this to a full white 2552558255 and its thickness you can specify at 3:00. How we display this image see you don't line show called this line. Roll the line. Blank image. Can we get a line that goes all across from 0.002250, 250? Let's try play wrong with this. Let's draw a line from 100 to maybe 250 and then it goes all the way to. 300. 2. Bhojani, save apple. Can you get a line that goes from 100 100 to 300, 400? Cool. And finally, the last thing that we will discuss in this video is how to write text on an image. That's so thats right text. Ohh name is. Now the way we do this is very straightforward. You see we do is. I will put. Text. And this will put text on the blank image we specify what we want to put on. So lets say hello. We can given origin which is basically where do we want to draw the image from. Let's set this 2225 \* 25. And we can also specify. Phone face now open CB comes with inbuilt phone faces and we will be using the see we dont fault unscrew Hershey. Amsco will be using the \*\*\*. You have complex, you have duplex, you have plane, you have script simplex and a lot of in bold fonts but for now. Let's use the triplex. Let's give this a phone scale, which is basically how much do you want to scale the front by. Let's to 1.0 you want to scale, appoint its, give it a colour of zero, 255, zero, and give it a thickness of 2. Ask on. Scrum enter out. And we can display this image. Received. I'm sure its called as text and pass in the blank image. Am I get some text that is placed on the image you play, run with it and say hello my name is Jason. Saver and run. Anna goes off screen. When we dealing with large images, but we can. There's no way of actually handling this except for me by changing the margins hear a bit. So we can do that by saying. Says singer, 225. Start from zero and says hello my name is Jason. So thats it for this video we talked about how to draw shapes, how to draw lines, rectangles, circles and how to write text on an image. Now in the next video we talked about basic functions in open CV that you most likely gonna come across whatever projects in computer vision you end up doing. So if thats it. I'll see you guys in the next video. Everyone and welcome back to another video. In this video, we talk about the most basic functions in open CV that you're going to come across in whatever computer vision project you end up building. So let's start up with first function. And that is converting image to grayscale. So we read an image and we display that image in new window and currently this is a BGR image, 3 channel Blue, green and red image. Now there are ways in open CV to essentially convert those BGR images to grayscale so that you only see the intensity distribution of pixels rather than the colour itself. So the way we do that is by saying grey is equal to UC. We dont sbt kollam. BA pass in the image that we want to convert from which is IMG and we specify a colour code. Now this kind of code is see we dont colour and scope. We are 2 great singer converting a BGR image to grayscale image and we can go ahead and display this image. I think Siri damn show. Grey. Great image. Say that in run. Basic dot PY. And this was the original image. And this is the grayscale image. Let's try this with another image. If. I know this. Yes, image of a park in Boston, say that. Maybe change that to Boston. And this is the BGR image in open CV and this is its corresponding grayscale image. So nothing too fancy, we just converted from a BGR image to a grayscale image. The next function we're going to discuss is how to blur an image. I'm blurring image, essentially remove some of the noise that exists in an image. For example in an image that may be some extra elements that were there because of bad lighting when the image was taken, or maybe some issues with the camera sensor and so on, and some of the ways we can actually reduce this noise. He's by applying a slight blow. Their way too many blurring techniques which we will get into in the advanced part of this coast. But for now we're just gonna use the Gaussian blur. So what we gonna do is going to create a blurred image by saying blue is equal to CB dot Gaussian. Blood and this image will take in a source image which is the IMG. It will take in a kernel science which is actually a 2 by 2 table which is basically the window size that opens up users to compute the blown. The image will get into this in the advanced part of the school, so don't worry too much about this, just know that. This kernel size has to be an odd number so so let's start a Brill simple and keep the kernel science 23 by 3. And another thing that we have to specify is see we dont border on school default so go ahead and try to display this image saying blur and pass in blue. Now you will be able to notice some of the differences in this image and that is because of the blower that is applied on it right this. The people in the background pretty clear this image and over here there slightly blurred out to increase a blow in this image, we can essentially increase the kernel science joke from 3 by 327 by 7. Say that and run. And this is the image that is way more blood than the previous image. So thats it. The next function we going to discuss is how to create an EJB cascade which is basically trying to find the edges that are present in the image. Now again there are many edge cascade that are available but for this video we're gonna be using the canny edge detector which is pretty famous in the computer vision. Well, essentially its a multi step process that involves a lot of blurring and that involves lot of gradient competitions and stuff like that. So we're going to take any. K is equal to Kennett will you passing the image. We pass in 2 threshold values which phone I'm going to 725 and 1:07 o'clock. Let's go ahead and try to display this image. Geek any changes and we can pass it Canon. Say that and run. And these were the edges that were found in this image. As you can see that about the any edges found in the sky, but lot of features in the trees and the buildings and quite a few, you know, features and answers in the grass and stuff. We can reduce some of these edges by century blurring the image in the way we do that is instead of passing the IMG, we pass in the blue. Say, then run. After for less edges that are found in the image and this is the way you can basically reduce the amount of edges that were found by a lot by applying a lot of low or get rid of some of the edges by applying a slight blow. Now the next function we going to discuss is how to dilate an image using a specific structuring element. Now the structuring element that we are going to use is actually these edges week any edges that were found. So we're going to say darling the image and the way we do that is my saying. Dilated is equal to see we don't dilate and this will take in the structuring element which is basically the Kenny. Edges animal taking a kernel science which we will specify as 3 by 3 for now and it will also taken and iterations of one. Now dilation can be applied using several iterations at the time, but for now we're just gonna stick with one. So go ahead and try to display this image by saying I'm show. Call this dilated and we can pass in guided. See there and run. And if these were these were edges, these are the dilated edges, we can may be increase the kernel science to me by 7 by 7. In tried to see what that does hobo. And nothing much. Who is done? How much difference was there? Let's try to increase the number of iterations, 2, maybe 3? And it's definitely we Sikar. Are you gonna see subtle differences with new model features and edges that you find? There is a way of eroding this dilated image to get back this structuring element. Not be perfect, but it will work in some cases. So what can I say? Call this roading. Can we call this eroded? Is equal to UCB dot. Erod. It will take in the dilated image passing dilated. It will take a kernel science of let's start up with 3 by 3 and given iterations of one just for now. And we can display. Show cause eroded. Favourite. And if this was your structuring element. And this was your dilated image. This is basically the result you get from eroding this image. Now it is not the same as structuring element, but you can just about make the features there. But you can see that between this and this there is a subject change in the address in the thickness of these edges. We can maybe try to match these values so that we attempt. So thats the reason attempt to get back this is cascade. Yes, we got the edges back. Now you can see that you compare these 2, they look pretty much the same. We entered the same. So essentially if you follow the same steps you can in most cases get back the same edge cascade. Andromeda last function are going to discuss its how to resize and crop an image. So I'm gonna start with resize. So we come to resizing video frames and images in the previous video in one of the previous videos. But which is going to touch on the CV to resize function? Just a bit sore gonna say resized. Resized is equal to resize. This will take an image to be resized and it will take in a destination source which not set this 2500 by 500. And so this is actually takes in this image of the park and resize the image to 500 by 500 ignoring the aspect ratio. So we display this image by saying see that I'm sure resized. And researched. Say that and run. And let's go back to this image. If this is the original image, this is the image that was resized 2500 by 500. Now by default there is an interpolations that occurs in the background and that is see we dont inter\_area. Now this interpolation method is useful if you are shrinking the image to dimensions that are smaller than that of the original dimensions. But in some cases if you are trying to enlarge the image in scale image. Too much larger dimensions. You are probably use the into\_linear or the International School CU. Now cubic is the slowest among them all, but the resulting image that you get is of a much high quality then the inter\_area or the Indian school linear. So let's touch on cropping. And that's basically by utilizing the fact that images are erased and we can employ something called array slicing. We can select a portion of the image on the basis of their pixel values. So we can say cropped is equal to. The image and we can select a region from 52200 and from 200 to 400. Yeah, we can display this image. Call is cropped, passing cropped. And this is a cropped image of let's go back here. Of this original image, you try to super impose M it's probably going to be right there. Yeah, its basically this portion. When is pretty much it this video we talked about the most basic functions in open CV. We talked about converting image to grayscale by applying some blue, by creating an edge cascade by dilating the image, by eroding that dilated image by resizing image and trying to crop an image. Using arrays licence. In the next video we're gonna be talked about image transformations in open CV. Thats translation, rotation, resizing, flipping and chronic. So if you have any questions leave them in the comments below, otherwise I'll see you guys in the next video. Everyone, and welcome back to this Python in OpenCV coast. In this section we're gonna cover basic image transformations. Now these are common techniques that you likely applying to images, including translation, rotation, resizing, flipping, cropping. So what started with translation? Translation is basically shifting an image along the X&Y axis. So using translation you can shift in image up, down, left, right or with any combination of the above. So. So to translate an image we can create a translating function are going to call this deaf translate. And this translation function will take in an image to translate and take in X&YX&Y basically stand for the number of pixels you want to shift along the X axis and y axis respectively. So do you translate an image, we need to create a translation matrix, so we're going to call this transmet. Is equal to MP dot float 32 and this will take in a list with 2 lists inside of it. In the first list we're going to say one common 0X and zero one, Why? Answers for using Napa we can import nampally. Implode Naam Hai hasin P. And once we've created a translation matrix, we can essentially get the dimensions of the image. Same dimensions which is a table of image don't shape of one which is the with an image not shape of zero which is the height. And we can return see we dont warp affine. This will take in the image matrix Sauternes met and I will take in a dimensions. And with that down, we can essentially translator image. Now before we do that, I do want to mention that if you have negative values for Xu, essentially translating the image to the left negative. Negative Y values implies shifting it up. Pause of X values implies shifting it to the right and as you guessed, positive Y values shifted down. So let's create after first translated image. Sending this equal to translate were gonna pass in the image. Image image right by 100 pixels and down by 100 pixels. Has to receive it online issue. Translated and translate. Saverin run Python transformations doc EY. And this is your translated image. It was shifted down by 100 pixels and shifted to the right by 100 pixels. So let's change that. Lets shift the image left by 100 pixels and down by 100 pixels. So we pass a negative values for X. L it moved to left. Feel free to play around these values as you see fit. Just know that negative X ships to the left, negative winships it up, X shifted to the right, and positive Y values shifted down. Moving on, let's talk about rotation. Rotation is exactly what it sounds like. Rotating an image by sumangal Open TV allows you to specify any point, any rotation point that you like to rotate the image around. Usually it's the center but but with open TV you can specify any arbitrary point, could be any corner, it could be 10 pixels to the right, 40 pixels down and you can shift the image around that point. So to talk to rotate the image we can create a rotating function calls diverted. This will take and image angle to rotate around. And rotation points. Which one is a which when I said it's not. So we're gonna grab the height and width of the image. Boys pressing. By setting this equal to orange you don't shape of the first 2 values. I basically if the rotation point is non we are going to resume that we want to rotate around the center. So we're gonna say raat point is equal to. Who is divided by divided by 2 in point divided by divided by 2. Can we can essentially create the rotation matrix like we did with the translation matrix by setting this equal to root myth is equal to get rotation matrix 2D leader passed in the center the rotation point? Angle to rotate around, which is angle and scale value. Now we are not interested in scaling the image when we rotate it, so we can set this to 1.0. Only we can set a dimensions variable equal to you the width and then the heart and we can return the rotated image which is see we dont warp affine. Image. Raut met. The destination sign switches dimensions and thats it. Don't we need for this function? So we can create a rotated image by sending this airport to rotate, and we can rotate the original image by. 45 degrees. So let's play this image called is rotated and passing rotated. Sailor in rock. And this is your rotated image. As you can see it was rotated counterclockwise by 45 degrees. If somehow you want to do root it this image clockwise. Just specify negative values for this angle and involve take the image around. Rotated clockwise. Now you can also rotate or rotated image, that is take this image and rotated by 45 degrees further. So let's call is rotated. Rotated rotated is equal to rotate or rotate TDD. Air we can rotate this image by another. 45 is so weird. Rotating it clockwise and we can see that I'm sure call this rotated, rotated and we can pass in rotated, rotated lot of root edits. And this is your roti rotated image. Now the reason why these black lines were included is because if there's no image in it, if there is no part of the image in it, it's going to be black by default. So when you took this image and rotated it by 45 degrees, you essentially rotate the image. But introduced these black triangles. Have you tried to rotate this image further buy some angle. You are also trying to rotate these black triangles along with it. So thats why you get these kind of skewed image. So this additional triangles are included over here. But see yourself the trouble and. Basically add up these angles and you will get the final angle. So we can change that to 90 and rotate the original image by -90. And this is essentially the image that we were trying to go for. Take this image rotated 45 degrees. Clockwise and rotate this 45 degrees image by further 45 degrees TV South trouble and add those 2 angle values. So so far we've covered to image transformations, translation and rotation. Now we're going to explode how to resize an image. Now this is nothing 2 different from what we discussed previously. But let's touch it just a bit sore resizing. And we can create a resized variable in SED is equal to see we dont try science, we can pass on the imagery science and the destination science of maybe 505 100 and by default the interpolation is inter\_area. I'm you could maybe change this to\_linear or inter\_cubic. Definitely amount of preference depending on radio enlarging or shrinking the image. If you are shrinking the image you will probably go for Inter\_area or stick with default. If you're enlarging the image you could probably use the entire\_linear or the Indian school cu cu a slower, but the resulting image is. Fed up with have a high quality again, I think 2 different from what we discussed before so we can display this image. I say resize. Air passing in resized. Run. Can we got a resize image? Next up we have flipping how to flip an image. So we don't need to define a function for this, we just need to create a variable is at this equal to CV. This will take in an image and a flip code. Now this flip code could either by zero, one or dead one. O basically employs flipping the image of vertically, that is over the X axis. One specifies that you want to flip the image horizontally or the Y axis, and then one basically implies. Flipping the image both vertically as well as horizontally. So let's start with zero flaming in vertically. How sign show called this flip, impulse and lip. Save and run. And this is the image that was flipped vertically. Let's Troy horizontal flip. How to get a horizontal flip? Surely see you when they was a horizontal flip. We can bring these 2 images together. Anything it looks like mirror images then it was flip horizontally. This is kind of a symmetric image, so it's not that obvious, but bring them together and you can maybe find out the difference. We could also try to flip the image vertically and horizontally by specifying -1 as a flip code. And the image was flip both vertically as well as horizontally. Mirror images but reverse mirror images. Everlast method is cropping. Now we discussed dropping against. I'm just gonna touch on it. We can create a very good call cropped and set is equal to IMG and perform sum array slicing so 2102403102. 400 say that and run. We just like this is. Hero and show this call is cropped. Pass in cropped, save and run. And this is the cropped image if you're trying to bring this together. So could be brought together. Current gram holders. Okay. So that's pretty much is this video we talked about. Translating an image, rotating that image, resizing image, flipping an image and cropping those images. We are basically just covering the basic basic image transformations there of course way mode transformation that you could possibly do with OpenCV, but just to keep this code simple, beginner friendly. Time only covering the basic transformations. So that's with this video. In the next video are going to talk about how to identify contours in an image. So if you have any questions, leave in the comments below, otherwise I'll see you guys in the next video. Everyone and welcome back to another video. In this video we're gonna be talk about how to identify contours in open CV. Now contours are basically the boundaries of objects, the line or curve that joins the continuous points along the boundary of an object. Now from a mathematical point of view then not the same is edges for the most part, if you can getaway with thinking of contours as edges, but from a mathematical POV. Contours and edges are 2 different things. Contours are useful tools when you get into shape analysis and object detection and recognition. So in this video I said want to introduce you to the idea of contours and how to identify them in open CV. So the first thing I've done is I've read in a file and image file and display that image using the CV data in Shell method. Then next thing I want to do is convert this image to grayscale by saying great is equal to see CVT. Colour. I'm gcv. Don't. Colour on this krovvidi are too, you're great and we can display this. So just another on the same footing I'm going to run this. Python contours don't PY. Can we get a Gray image over here? Now after this I want to essentially grab the edges of the image using Makani Edge detector. So want to take any is equal to. How can I go to pass on the angry and we're gonna give it 2 threshold values so 125 and 175. We can display this image. Calling this canny edge is possible. Can I say that in run? Weather at in, Saver, Saver, North Run. And these are the edges there were there in the image. Now the way we find the contours of this image is by using the find contours method at. This method basically returns 2 things contours in hierarchies and essentially this is equal to see if I don't find contours this sex in the. Edges, so can I? It takes in a mode in which to find the contours. Now this is either CV dot. Reter on this gotri if you want all the hierarchical contours or the writer external if you want only the external contours or. Oreta list if you want all the cartoons in the image, the next method we pass in is actually the contour approximation method. For now we're gonna set this to UCD chain, unscrew approx and scope no. So lets lets just have a top down look and what this function does. So essentially the CVR find contours method looks at the structuring element or the edges that were found in the image and returns to values the contours which is essentially a Python list of all the coordinates of the contours that are found in the image and hierarchies which is really out of scope of this. Course, but essentially it refers to the hierarchical representation of contours. So for example, if you have a rectangle and inside the rectangle if you have a square instead of that's quite you have a circle. So this hierarchies essentially the representation that open C users to find these quarters. Desi hot metal list essentially is a mode in which this find contours method returns and files. The Contours reading list essentially returns all the countries that find in the image. We also have writer external that we discussed. Graphics download retrieves only the external contours, so all the ones. The outside E returns those reader\_tree. Returns all the hierarchical contours, all the contours that are in a hierarchical system that is returned by reference kothri. For now I'm just gonna set this 21 list to return all the contours in the image. The next one we have is the Core 2 approximation method. This is basically how we want to approximate the contour. So chain approx none does nothing, I just returns all of the contours. Some people prefer to use chain approx symbol which essentially compresses. All the contours that are returned into simple ones that make most sense. So for example if you have a line in an image, if you use chain approx none, you are essentially going to get all the contours, all the coordinates of the points of that line. Jeena Prox simple essentially takes all of those points of that line. Compress is it into the 2 end points only because that makes the most sense of line is defined by only 2 end points. We dont want all the points in between. That in a nutshell is what this entire function is doing. Jobs in coaching is a list. Weekend assembly find the number of contours they were found by finding the length of this less so we can print. Print length of his less and we can say. Ah, fair. We can say. Okay, so you these many contours. Sound. Okay. So lets say that and run. And we found 2794 contours in the image. This is huge. This is a lot of contours ever found in the image. So let's do a couple of things. Let's try to change this. Change approx symbol to change proxy in London and see what that does. See how that affects on length? Now there isn't any difference between those 2 because I'm guessing that there were no points to compress, and since there are lot of edges in points in this image, so there was a lot of compression. So let's change the back to simple. And actually I want to do is I wanna blow this image before I find the edges. So let's do this, let's do a blow. Is equal to receive dot Gaussian blur passing the great image and we can give the kernel size of its. Let's do a lot of know so 5 by 5 and maybe we can give it body defaulters of border on screw default. And we can if you want to go and we can display this image. Call this blow and passing loo. Hey, we can find the edges on this blood image, so let's call is below. And as you can see the significant reduction in the number of contours that will found just by blurring the image all away from 2794, 2380 that's close to 7 times just blurring image with kernel size of 5 by 5. Okay, now there is another way of finding the contours is that instead of using this canny edge detector we can use another function in open CV and that is threshold. So I'm just going to connect this out. And down hear what I'm gonna do is I'm going to say Ret thresh is equal to see we dont threshold. This will take in the great image and will take in a threshold value of 125 and a maximum value of 25. Don't worry too much about potential in for now, just know that thresholding essentially looks at an image and tries to binarize that image. So if a particular pixel is below 125, if the intensity of a pixel is below 125, it's going to be set 20 or black. If it is Bob 125, it is sent to White. 4 to 5 thats only does and in the final countries method we can assembly pass in the thrush value. So lets say that lets close this out and try to run that. Time okay threshold missing that time. Okay. I think I forgot pond part. We just specify a threshold link type. So this is CV dope fresh and escrow binary Bina rising the image basically. Okay. That's wrong, bats. And there were 839 contours that were found. We can visualise that. Let's print how to display this thrush image. Passing Athresh say that Enron. And this was a threshold that image. You're using 125. Close this out using 125 as threshold value and 255 as a maximum value. We got this threshold image and when we trying to find the contours on this image, we got 839 contours. Now don't worry too much about this threshold in business. We will discuss this in the advanced section of this course more in depth. Just know that thresholding attempts to binarize an image taken image and convert it into binary form that is either zero or black or white or 2.5. Now what school in open CV is that? You can actually visualise the contours that were found. On the image, by essentially drawing over the image they want to do real quick, he is actually imposed. Naam hai. Naam Hai as MP. And after this I'm going to create a blank. Variable and static equal to MP don't zeros of image don't shape of the first values and maybe give it a data type off. I don't want let's UNT 8. We can display this image. Cools blink. Prasan blink. Just to visualise and have a blank image to work with. Let's say that and go to a blank image. This is of the same dimensions as original caps image. So what I'm gonna do is I'm gonna draw these contours on that blank image so that we know what kind of countries that opens if I found. So the way we do that is by using the. See we dont draw contours method, it takes in an image to draw over, so blank it takes in the contours which has to be a list which is in this case it's just the quantities list. It takes in a quanto index which are basically how many contacts do you want. In the image since we want all of them since we want to draw all of them we can specify -1 give the colour that said this to BGR. So let said this to read 00255 and we can give the thickness of maybe 2 and we can display the blank image. So thats called this. Contours drawn. Every can pass in black. Saverin run. Okay, there was an error. Think this has to be shape. Okay, so these were the countries that were drawn on the image if you take a look at the threshold value. A threshold image it's not the same thing. What I believe it attempted to do is instead it found the edges of this image. All the edges of this image in attempted to draw it out on this blank image. So let's set this so we have started thickness to maybe one so that we have a crisp of you. Okay, so with these were the contours that were drawn in the image. In fact, you are trying to visualise it with the Kenny. Let's actually visualise that with Kenny. Uncomment that out. Run. Blogs on the point. Okay that has to be image. K What's Luke Kenny? Let's look at this. Okay, it's not the same thing. And that make sense because of fine cottons method and use Kenny as a basis of detecting and finding the contours. But we can do that. Let's. Not user thresholding method and instead let's use Kenny so we can possum can hear. Say that and run. And okay, that's pretty much the same thing. Read. It's basically a mirror image of these 2, like I said. You can getaway with thinking of contours. Edges are not the same thing, but but you can think of a message begins from a programming point of view. They kinda like the edges of the image, right? They are the boundaries. There curbs that join the points along the boundary. Those are basically edges. Let's try to block that image. What's on coming that out? Let's see what that does. I don't think that hear any effect because we can pass in blue. K 380 contours ever found and mirror images of each other. So generally what I recommend is that you use scanning method first and then try to find the contours using that rather than trying to threshold the image and then finally contours on that. Because like we will discuss in the advanced section, this type of thresholding, the simple thresholding has its disadvantages. Maybe because we're passing in a simple just one value broadband rise the image using this threshold value right. It's not the most ideal, but in some cases, in most cases it is most favorite kind of threshold because its simplest and it doesn't jump pretty well. So that's pretty much it. For this video we talk about how to identify contours in open CV. But in 2 methods, first trying to find the edge cascades of the image using wicket taker and try to find the quantities using that and also trying to binarize that image using acetyl threshold and finding the contours on that. So if you have any questions, leave them in the comments below. I'll be sure to check him out, otherwise has always. I'll see you guys in the next video. Everyone, and welcome back to another video. We are now at the advanced section of the schools where we are going to discuss the advanced concepts in open. Say we. So what we gonna be doing in this video is actually discussing how to switch between colour spaces in open CV. How? Colour space is basically a space of colours. A system of representing an array of pixel colours. RGB is a kind of space, grey scale is colour space. We also have all the colour spaces like HSV, lab and many more. So let's start off with trying to convert this. Image to grayscale. So we're going to convert from a BGR image which is open CVS default way of reading and images and we're going to convert that to grayscale. So the way we do that is best saying grey is equal to CBT colour. We pass in the image and we specify a colour code which is CV dot colour\_2. To grab since we are converting from a BGR image format to grayscale format. And we can display this image. I saying great and passing in great. Lets say that in run Python spaces. Veer problem as, save in run. And this is the grayscale version of this BGR image. Cool, pretty cool. Grey scale images basically show you the distribution of pixel intensities at particular locations of your image. So let's start off with trying to convert this image to an HSV format. So from Sir from Dgr to HSV. Hsv is also called her saturation value and is kind of based on how humans think and can see that colour. So the way we convert that dispersing HSV is equal to CV CVT colour we passed in the IMG variable and we specify a colour code which is see we dont colour and go to HSV and we can display this image. Coles HSV in person HSV. Let's say that. And this is the HD version of this BG image. As you can see that there is a lot of green. In this area and the skies are red. Fish. Now we also have another kind of colour space and that is called the lab colour space. So we're going to convert from BGR to LLB. This salt and drop is added as L times 8 times by but but be free to use whatever you want. So IB is equal to. GT Kholo you personally mg and the colour under screw BGR to maybe. Define show commerce lab and pass in now a, B. Just want that. And this is the LLB version of this BGR image. This current looks like a washdown version of this BGR image. But hey, that's the LLB format is not Tutu how humans perceive colour. Now when I start off this cause I mentioned that open save you reads in images in a BGR format that is blue. Green and red and that's not the current system that we used to represent colours outside of open CV. I'll try to conceive you. We use the RGB format which is kind of like the inverse of the BGR format. Now you tried to display this ING image in Python library that's not open TV, you are probably going to see and inversion. Of colours and we can do that real quick. Let's try to import met blocked Lib Dot pipe lot as PLT. We can, we can basically. Run CMD command that out and we can try and display this image variables were gonna say PLC dot I am show passing the image and we can say we can see the PLC don't show me by that comment this out see that and run and this is the image you get now. If you compare with the image that opens TV read this is completely different. These 2 are completely different images and the reason for this is because this image is a BGR image and open CV displays BGR images. But now if you try to take this BG image and try to displayed in map of labour instance. Mapa lived has no idea that this image is a BGR image and display that image as if it were an RGB image. So thats why you see and inversion of colours. So when this read over here UCA blue whale this blue over here UCA red and there are ways to convert this from BGR to RGB and that is. By using. You can see itself, so let's comment that out and let's uncomment this all out. And. Run over here, lets say BGR to RGB. And we're gonna say its RRGB is equal to use CVT colour, we can pass in the BGR image. FC. We can pass in the BGR image and we're going to do is specifier colour code which is see we dont colour on this KO BGR to RGB. And we can try to display this image in in open CV and see what that displays. Rgv and we can also display this in. Naaptol slip so thats pass in the RGB and we can do PLT dot show say that. Koshy. Its its your Python spaces dumpty. Why? Why most interested in is this. And this now again you see and inversion of colours but this time in open CV because now you provided open CV and RGB image and it assumed it was a BGR image and that's why there's an inversion of colours. But we passed in the RGB image to Matplotlib and map of lips default is RGB so thats why it displays. Proper image. So just keep this in mind when you're working with multiple libraries, including open CV and Matplotlib for instance, because do keep in mind the inversion of colours that tends to take place between these 2 backwards. Sir. No, nothing that I want to do is we've essentially converted BGR to grayscale. We have essentially converted video teachers video to LLB, BGR to RGB. What we can do, if we can do the inverse of that, we can convert to grayscale image to BGR, we can convert an issue Sveta BDR. We can convert in LLB to BGR and RGB to video and SOA. But here's one of the downsides. You cannot convert grayscale image to HSV directly. If you wanted to do that, what you have to do is convert to grayscale, to BGR and then from video to HSV. So we gonna do that real quick. So we're going to say HSV 2. So the first thing we do is HSV\_VDR. Basically converting from ages is equal to use CVT Khan. This will taken by HSV image. And the colour code will be colour on disco. Hsv to BGR and we can display this image that's called as HSV 2. Bgr and pass in H the on square BDR. Sounds good you are say that and run. Okay, were not interested in this so lets close this out. But it's actually this is the H3 2 BGR image. If this would be HD image we converted this image to BGR. And we can try this with LLB. So let's call this LLB 2. Maybe it's called maybe and its. Copy this and paste out there. We can get rid of Nepal visit in address in an email. Siva, run. Run. Okay, there was a mistake, we say. Each is the elevates you. Hello baby TV. That was my mistake. Cool. So if this was the LLB version. This is the limited BGR version back from BGM to and from LLB to BGR. So thats very much if for this video we discussed how to convert. We discuss how to convert between colour spaces from BGR to grayscale, HSV, LLB an RGB and if you want to convert from grayscale to LLB for instance, note that there is no. Direct method. What you can do is convert to grayscale to BGR and then from BGR to nabh. That's possible, but directly I don't think there is a way to do that. If open, see if you could come up with a feature like that. It would be good, but it's not gonna hurt you to write extra lines of code, at least 2 or 3 lines of code extra. What do I hurt you? In the next video we talk about how to split and merge colour channels in open CV. If you have any questions, leave in the comments, blow. Otherwise I'll see you guys in the next video. Everyone and welcome back to another video. In this video we're gonna be talk about how to split and merge colour channels in open CV. Now colour image basically consists of multiple channels, red, green and blue. All the images you see around you. All the BGR or the RGB images are basically these 3 colour channels merged together. Now we can see the allows you to split an image into its respective colour channels so you can take BGR. Image and split it into blue, green and red components. So that's what we're gonna be doing in this video. We're gonna be taking this image of the park that we had seen in the previous videos, and we're going to split that into its 3 colour channels. So the way we do that is by saying be coming Kumar with stand for the respective colour channels and stuff is equal to see we don't split. Split of the image so they see that split basically splits the image into blue, green and red. And we can display this image by saying C dot. I'm sure that's called this blue and passing blue. Let's do the same for. Green image in has inj and we can do the same for the red part. 2 are. Yeah we can actually visualise the shape, the shapes of these images. So thats first \*\*\*\*\* image don't shape and print the be done shape and print the genome shape in that print Dr Dot sheet. So basically we're printing the shapes and dimensions of the image. And blue, green, red and we also displaying these images so thats run Python split merged PY. And these are the images that you get back. This is the blue, the blue image, this is the green image, and this is read image. Now these are depicted in displayed as grayscale images that show the distribution of pixel intensities, regions where it's later showing that there is a. Farmer concentration of those pixel values and regions where it's darker represent data little or even know pixels in that region. So take a look at the blue, Blue Channel first. If you can you compare with the original image. You will see that the sky is almost white. This basically shows you that there is a high concentration of blue in the sky and not so much in the trees over the grass. Let's take a look at the green and there is very even distribution of pixel intensities between the between the grass. The tree is and some parts of the sky. And take a look at the red colour channel and you can see that parts of the trees that are red. Whiter in the grass in the sky. I'm not that white in this read image. So this means that there is not much red colour in those regions. Now coming back, let's take a look at the sheep. Sabine is now. This stands for the original image, the BGR image. The additional elements in the turbo here represents. The number of colour channels 3 represents 3 colour channels, blue, green and red. Now we proceeded to display the shapes of BG, ANR components. We don't see a 3:00 in the table. That's because the shape of the component is one is not mentioned here but it is one. Thats why when you try to display this. Image using C with time show it is pleasant as a grayscale image because grey scale images have a shape of one. Now let's try and merge these colours channels together. So the way we do that is by saying that merged image merge images equal to see if you don't merge and what we do is we pass in a list of blue or blue, G. Save that in. Let's display that since I Don show coldest M. Coldest emerged image and we can pass in merged. So lets say that North Run. Can we get back to merge image by basically merging the 3 individual colour channels thread? Green and. Now there is away an additional way of looking at the actual colour there is in that Channel. So instead of showing you grey scale images it shows you the actual colour involved. Do for the blue image you will get the blue colour channel for the red. Channel, you got the red colour for that Channel and the way we do that is we actually have to reconstruct the image. The shapes of these images are basically grey scale images, but what we can do is we can actually create a blank image. Blank image using Naam Hai. In essentially booking ID you can say blank is equal to MP dot zeros and we're gonna set this to the shape. Of the image but only the first 2 values and we can give it a data type of you 88 which basically offer images and to print the blue colour Channel V gonna do is going to say. Down here we use blue is equal to see if you don't merge with upasna list of B, blank come up link. M begin to do the same thing for green in South is equal to see the dog merge of blank, G blank and going to do the same thing for read by setting this equal to use merge of blank, blink, cover red. Basically what I've done is this blank image basically consists of the height and width, not necessarily the number of colours, channels in the image. So by essentially merging the blue image in its respective compartment, so blue, green and red, we are sending the green and the red components to. Black and only displaying the Blue Channel are doing the same thing for the green by setting in the blue and the red components to black, and the same thing for read by setting the blue and green components to black. And we can display this bye saying. Bling. Green and red. At Saverin run. And now you actually get the colour its respective colour channels. Take a look at this. You will not be able to visualise a distribution much better. Here. You can see lot of later portions represent a high distribution. Lighter portions here represent hi distribution of red and hired weather regions. Represent a high distribution of green. So recently if you take these 3 images of these colours channels and merge them together, you essentially get back the merged image that is that the motion image. So that's pretty much it. For this video. We discuss how to split an image into its respective college channels, how to reconstruct the image to display the actual colour involved in that Channel, and how to merge those colour channels back into its original image. In the next video will be talk about how to smooth and blue and image. Using various blurring techniques. If you have any questions, leave in the comments below, otherwise I'll see you guys in the next video. Paper one, and welcome back to another video. In this video, we're going to address the concepts of smoothing and blurring in urgency V. Now before I mentioned that we generally smooth and image when it tends to have a lot of noise and noise that's cause from camera sensors or basically problems in lighting when the images taken and we can essentially smooth out the image or reduce some of the noise by applying some blurring method. Now previously we discussed the Gaussian blow method which is kind of one of the most popular methods in blurring. But generally you going to see that Gaussian blur won't really suit some of your purposes and that's why there are many blurring techniques that we have and that's what we're gonna dress in this video now before we actually. Do that. I do want to address a couple of concepts. Let's actually go to an image and discuss what exactly goes on when you try to apply blow. So essentially the first thing we need to define is something called kernel or window. That is essentially this window that you draw over an image. That. As 2 andhere men, let's join up align there. So this is essentially a window that you draw over a specific portion of an image, and something happens on the pixels in this window that changed to blue. So essentially this window has a size. This size is called kernel size. Now colour size. Basically the number of roads and the number of columns. So over here we have 3 columns and 3 roads. So the kernel size for this is 3 by 3. Now essentially what happens here is that we have multiple methods to apply some blue, so its actually blur is applied to the middle pixel as a result of the pixels around it also call the surrounding pixels. That's. Change that 2 different colour. So something happens here as a result of the pixels around the surrounding pixels. So with that in mind, let's go back and discuss the first method of blurring, which is averaging. So essentially averaging is we define a kernel window over a specific portion of an image. This window will essentially compute the pixel intensity of the middle pixel of the true center as the average of the surrounding pixel intensities. So if this was, suppose if this pixel intensity. Wazwan this word maybe 2 this is 3456780 got the point is actually the new pixel intensity for this region will be the average of all the surrounding pixel intensities. So thats coming up 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 and dividing that bye. It which is essentially the number of surrounding pixels of essentially use that result as a pixel intensity for the middle value or the truth center. And this process happens throughout the image. So this window basically slides to the right and once that's done, its lines down and computer basically for all the pixels. In the image. So let's try to apply and see what this does. So what we gonna do we use average is equal to CV dot blue. Receiving a blow method is a method in which we can apply averaging blow. So we define the source image which is IMG. We give the kernel size of ICA 3 by 3 and thats it. We can display this image call with average. Average blue. Safe, Aaron. Run. Python smoothing da PY. Maps, Ghosh. We have to pass in the average. Say that in Ron. And this is basically the average flow that supplied. So what the algorithm did in the background was essentially define Akola window of the specified size 3 by 3 and add computed the center value for a pixel using the average of all the surrounding pixel intensities and the results of that is we get. A blurred image, so the higher kernel size we specify, the more blur there is going to be in the image. So let's increase that to 7 by 7 and see what that does. Let me get an image with way more blur. So let's move on to the next method, which is the Gaussian blur. So Gaussian basically does the same thing thats averaging, except that instead of computing the average of all of this running pixel intensity, each surrounding pixel is given a particular wait and essentially the average. Of the products of those weights give you the value of 42 centre. Now using this method you tend to get less blurring then compared to the averaging method. But the gaps in blood is more natural as compared to average. So let's print out that's cold is layout. And South is equal to CB dot cabs in blood and this will say can the source image to MG Road Karnal size of 7 by 7 just to compare with the averaging. And another parameter that we need to specify is Sigma X or basically the standard deviation in the X direction, which for now it's just gonna set as zero and we can put that out. Kohli's gaussian. Play impossible Gas Safe for and run. If you compare with this you see that both of them use the same kernel size, but this is less blood as compared to the average method. And the reason this is because a sudden weight value who is Adam in computing the blur. Okay, so let's move on to the next method. And that is meaning blue. So let's go back to my image and meaning. Learning is basically the same thing as averaging, except that instead of finding the average of the surrounding pixels, it finds the median of the surrounding pixels. Generally, medium blowing tends to be more effective in reducing noise in image. As compared to averaging and even Gaussian blur and its very good and removing some salt and pepper noise that may exist in the image in journey people tend to use this image in it bonds computer vision projects that tend to depend on the reduction of substantial amount to noise. So let's go back here. And the way we applying blow it by saying it's called is median in SAP is equal to CV dot median. Blurred image and this kernel size will not be October or 3 by 3 but instead just an integer so 3. And the reason for this is because you have been seen me automatically assumes that this kernel size will be a 3 by 3, just best office integer. Erik and print assert its called this median. Blue & million. L. Satpathy 7. And comparing the grid Gaussian blur. An average in blur. You tend to look at this end. You can make out some differences between the 2 images. So its like as if this was your painting and it was still drying and you take something and smudge over the image and you get something like this. Gallery medium blurring is not meant for high kernel size like 7 or even 5 in some cases and its more effective in reducing some noise in the urge so thats thats change this Paul 23 by 3. Copy that changed 33.3. And we can change that to 3. Now let's have a comparison between the 3. This is your Gaussian blur. This is your average in blue. This is your medium low. So compare with these 2 you can see that. Aryan. Kind of less blurring when Gaussian. But you can sort of make out the differences between the 2. It's very subtle, but there are a couple of differences between the 2 fairly. The last method we going to discuss is bilateral blurring. Commerce, bye. Metro. Bilateral bearing is the most effective and sometimes used in a lot of advanced computer vision projects, essentially because of how it blows. Now traditional blowing methods basically blow the image without looking at whether you whether you're reducing edges. The image or not? Bilateral blurring applies blurring but retains the edges in the image. So you have a blood image, but you get to retain the edges as well. So let's call this vine lateral. Insert row. NCRT SQL to CB DOT bilateral filter and we pass in the image. We give it a diameter of the Pixel mod. Now notice this isn't a kernel size but in fact the diameter. So let's set this 25 for now. Give it a Sigma colour, which is basically the colour Sigma. Sigma colour. A larger value for this colour Sigma means that there are more colours in the neighborhood that will be considered when the blowers computed. So thats at this 215 for now. And Sigma space is basically your space Sigma. Larger values of this space Sigma means that pixels. Further out from the center Pixel will influence the blurring calculation. So let's at this 250. So let's take a look at that Sigma space thing. So for example in bilateral filtering, if this is a value for this central pixel or the truth center is being computed by giving a larger values. For the Sigma space, you essentially indicating that whether you want pixels from. This 4 way or maybe this far away or even this far away from influencing this particular calculation. So if you give like really huge numbers, then probably a pixel in this region might influence the computation of this pixel value. So let Satish to 15 to know. Let's display this image so called and show this call is bilateral. M percent bilateral. Let's say about Hiran run. And this is your bile natural image. So let's compare with all the previous ones that we had compared with this. Much better compared with averaging. Way much better. S compare with median. The edges are still there, is slightly blood. If you compare with the original image, the kernel look the same thing. Okay come looks like there's no blow Clyde, so maybe lets increase this diameter to 10. And not much was done the edges so where it kind of looks like the original image itself. So let's try Intune one of the other parameters. Let this 23435 at set this 225. Really playing around these it is valid and now you can basically make her that this is starting to look alot like median blow. When you given launch values, it's starting to show you that this is more looking like a smudged painting version of this image, right? There's a little blue applied here. But the cats are looking smudged, so definitely keep that in mind when you are trying to apply blur the image, especially with bilateral median lowering because higher values of this basic mouth for bilateral or the kernel size for medium during and you tend to end with a washed up smudged version. This image, so definitely keep that in mind, but that kind of summarises whatever we we have done in this video we discussed averaging gassy and median and bilateral blurring. So in the next video we talk about bitwise operators in open CV. So again like always if you have any questions. Leave them in the comments below, otherwise I'll see you guys in the next video. Paper one and welcome back to another video. In this video we're gonna be talking about bitwise operators in open CB. Now there are 4 basic bitwise operators and, or, XOR and notes. If you ever taken an introductory CSS codes, you will probably find these terms familiar. Bitwise operators and they are in fact use a lot in image processing, especially when were working with mosques like will do in the next video. So at a very high level, bitwise operators operate in a binary manner. So a pixel is turned off if it has a value of zero, and it's turned on if it has a value of one. So lets actually go ahead and trying to import Naam Hai. Hazem heap and wanna do is going to create a blank variable and set this equal to MP dot zeros. Of size. 400. By 400 and we can get the data type of uin. See it is. What I'm going to do is going to use this blank variable as a basis to draw a rectangle and draw circle. So I'm gonna say rectangle. Is equal to CB dot rectangle. We can say blank copy. And we can pass in. The starting point. So let's give the margin of around 30 pixels on either side. So we're gonna start from 30, 30 and we can go all the way across 2370370 and we can get a colour. Since this is not a colour image but rather binary image, we can just give it one parameter, so 255 white and given thickness of.-1 because we want to fill this image and then I'm going to create another single variable inside this equal to C dot circle. I'm going to say blank the copy. We are going to give it a sensor so the center will be the absolute center so 200 by 200 and thats given radius of given radius of 200 and give the colour of 255 and lets fill in the circle so -1. So let's display this image and see what we've C working with. They will call this rectangle.& direct L I'm going to do the same thing with the circle is called a circle. Impasse in this circle. Say that and run Python bitwise. So we have to images that we're going to work with, this image of rectangle and this image of a circle. So let's start with the first basic bitwise operator, and that is bitwise. And so before actually discuss what bitwise and really is, let me show you what it does. So essentially what I'm doing want to say big points on school ends equal to. Between and. And basically what I have to do is pass into source images that are these 2 images, rectangle and circle. Now we can display this image. Let's call this. Bitwise and its possible bit wines and save run. And essentially you get back. Vis image. So essentially would be your end did was it took these 2 images, place them on top of each other and basically return the intersection. Right, and you can make out when you take this image, put it over this image, you have some triangles, they are not common to both of these images and so those are set to black O while the common regions are returned. Show the next one is basically bitwise or. Now bitwise or real simply returns both the intersecting as well as a non intersecting regions. So let's try this bitwise or is equal to use bitwise and scope or you pass in rectangle, we pass in circle. Then we can print that. Thats course bitwise or pass in bitwise. Apps that was for say that and run. And bitwise or. Boys, poor prime mistake. Supervisor basically return this funky looking this funky looking shape. Essentially what I did is it took these 2 images, put them over each other from the common regions and also found the regions that are not common to both of these images and basically superimpose them. So basically you can just put them together and. Find the resulting shape and this is what you get for this image. Over this and you get this moving on. The next one is bitwise. Xor which basically is good for returning the non intersecting regions. So this found the. The intersecting. Lips, the Internet setting regions. This found. Loan. Intersecting intersecting regions and XOR only finds the known. Intersecting regions. So let's do that. I say bitwise called. This explorer is equal to CV dot bitwise on the school X or V pass in the rectangle. Possible return when we pass in this circle. We can display this COM sure calls bitwise. Xor and you can pass in bitwise XOR. Say that and run. And here we have the no need to cycling regions of these 2 images when you put them over each other. Rico. End just a re camp. This bitwise and again returns the intersecting regions. Bitwise or returns the non is second regions as well as intersecting regions. Bitwise XOR returns the non intersecting regions. So essentially if you take this bitwise XOR and subtracted from bitwise. Or you get bitwise and, and conversely, if you subtract bitwise and from the twice all you get bitwise XOR, just. So essentially that's a good way of visualising what exactly happens with these big points operators. And finally, the last method we can discuss is bitwise. Not. Essentially doesn't return anything. What it does is it inverts the binary colour. So let's do that. So thats called as bitwise knot is equal to use bitwise. Unsecured, not, and this only takes in one source image. So let's set this to the rectangle and we can display this. Coolest rectangle not we can pass in bitwise not. Say that and run. Basically what he did is if you look at this arrange it found all the white regions, all the white pixels in the image and inverted them to black and all the black images it inverted to white sentry it converted white to black in from the ad from black to white. So we try that with this circle thats called this. Circle. We can pass in this circle here. Save and run and the resultant the resultant circle, not that you get is this. This is white hole. This is a black hole physics software. Okay so thats very much for this video. I just wanted to introduce your to the idea of bitwise operations and how it works. In the next video will be actually talk about how to use these bitwise operations in a concept called masking. So if you have any questions leave them in the comments below, otherwise I'll see you guys in the next video. Hirvonen, welcome back. In this video we're going to talk about masking in open CV. Now, in the previous video we discussed bitwise operations. And using those bitwise operations we can essentially perform masking in OpenCV. Masking essentially allows us to focus on certain parts of an image that would like to focus on. So for example, if you have an image of people in it and if you are interested in focusing on the faces of those people. You could essentially apply masking in Moscow with people's faces and remove all the unwanted parts of the image. So that's basically a high level intuition behind this. So lets actually see how this works. In you can say be. So I've basically reading a file and display that image. Young thing I want to do is I want to import Naam Hai Naam Hai Erzan P I'm doing a blank is equal to MP dot. Zeros of size of size image don't shape of the first 2 values. Now this is extremely important. The dimensions of the mask have to be the same size as that. The image. If it isn't, it's not going to work and we can get the data type of you aren't I hate you in it. If you want to display this week and display this, it's just gonna be a black image. This call is blank image in pass in black. Essentially what I'm doing to draw a circle over this blank image and call that my mask. So I'm going to say mask is equal to receive it out circle going to draw a blank image. On the blank image we can give it a centre of this image. So lets say image don't shape of of one divided divided by 2 and image don't shape of 2. Image shape of zero divided by divided by 2 we can give the radius of I don't know let's say. 100 pixels, give it a colour of 255, give it a thickness of -1 and we can visualise a mask as muskan pass in mosque. Let's run that. Python masking. And this is a century are mask. Best Blank image working with and this is the image that we want to mask over. So lets actually create a masked image were gonna say masked. Image is equal to UCV dot bitwise\_end. This source image so ING ING and we specify the parameter mask is equal to mask which is this circle image over here and we can display this image call this masked image and we can pass in masked. C Bar and run. And this is essentially your masked image. You took this image, you took this image, you put this image over it. And from the intersecting region. Okay bye optionally passing the mask is equal to mosque. That's exactly what we doing. Cool. Thats right in you know play around with this. Thats many movies by couple of pixels runs say 45 save and run. Smith down. Okay, this has been 45. Pass 40.71. Can we get the images accounts we can drive, we can draw a circle, we can draw rectangle instead that's pass on Blanc. Eau skip that? It's fate, given that, Andrew. Give it is starting endpoint of. Let's copy this. And add a couple of pics. So maybe 100 pixels this way in 100 pixels this way you can get rid of this me that say that run. This is this is square and this is essentially the masked image. So lets actually try this with. Trying this with a different image. So we have got an image. Let's try it with media scans 2. Scratch kids to say that wrong. And this is the mask that we get by putting these 2 or each other. Aricent, you can play around with this as you feel fit. You could maybe try different shapes, weird shapes and the way you can get these weird shapes. Essentially creating a circle, a rectangle and applying bitwise and you get this weird shape and then you can use that weird shape as your mask. So. Let's just try that, let's let's try that were gonna say. Hey, art school is so cool. Invoke copy. Copy and create a rectangle. Let's just grab it from this we don't read. By the way. From device. Let's map. This. Square rectangle. Copy that. Hey, somewhere along the copy 3030. Okay, going same shape and lets so that's great. Just weird weird shape is equal to bitwise end of this circle, this rectangle. And we dont need to specify anything else. What sort of visualises that's close this out? Chachi receive Sivaram show coldest a weird shape. Pass in the weird shape. Same wrong. Asking Notifiers Moskowitz mosque. Okay. Right. Okay, this is the weird shape that we get. We are not really going for a Half Moon, but hey, whatever, let's close. So. Use this. Real shape is mask. So use. Weird shape has mass, cannot see the final mast image and this is essentially you're weirded Rion shape masked image. Raj College weird shaped mass temperature weird shaped masked image. This little Half Moon here. And sensor you can. You can do pretty much anything you want with this. You can experiment with various shapes and sizes and stuff like that, but just know that the science of your mask has to be at the same dimensions as that of your image. If you want to see why, that's maybe subtract 100 pixels. Not possible that unsupported yeah so thats may be like subtractor tubal 100. Don't know whether what bus? Guesswork. Okay, so let's just say image don't shape off. Call. Okay, let's just give it a different size. Why we using image that's good signs of 300 by 300. Definitely you know the science of this. Can we get this assertion failed M time blah blah blah mask, same size in function whatever. So essentially these need to be at the same science otherwise it's gonna fail and throw you in error. So thats it for this video we talked about masking, again nothing to do different, we have SN. Shall I use the concept of bitwise and from previous video? And you'll see that when we we want to computing histograms in the next video, where masking really comes into play and how masking really affects your histograms. So if you have any questions, again, leave them in the comments below are the wine sounds. See you in the next video. Hey everyone and welcome back to another video. In this video we're gonna be talk about computing histograms in open CV. Now histograms basically allow you to visualise the distribution of pixel intensities in an image. So whether its a colour image or whether its a grayscale image. You can visualise these pixel intensity distributions with the help of histogram which is came like a graph or a plot. I will give you a high level. Integration of the pixel distribution in the image. So we can computer histogram for grayscale images and computer histogram for RGB images. So we're gonna start off with computer histograms for grayscale images and so lets just convert this image to grayscale TV dot TV colour. It's past the image and give it correct of of colour\_BGR to Great Britain street image of course grey. And passing rate. Yeah, to actually compute the grayscale histogram. What we need to do is essentially called this great on school list and set this equal to CB dot calc hist. Which method will essentially compute the histogram for the image that we pass into? Now this images is a list, so we need to pass in a list of images. Now since we are only interested in computing a histogram for one image, let's just pass in the D grayscale image they are saying we have to pass in. Is the number of channels which basically specify the index of the Channel V. Compute a histogram 4, but since we are computer in histogram for grayscale image, that's right, this as a list and pass in zero. Next thing we have to do is provide a mosque. Do we want to computer histogram for specific portion of an image? We will get to this later, but now just have this student, not a his size is basically the number of bills that we want to use for computing histogram. Essentially when we plot histogram, I'll talk about his concept of bins, but essentially for now just sent this to 256. Raptadu list so that's rough as list in. The next thing I want to do is specify the range is the range of all possible pixel values. Now for rks this will be 02256 and thats it. Show to put this image, let's actually use Matplotlib. So input Matlab, Matplotlib Pyplot as PLT and then we can instantiate a PLT figure. A P else figure. Let's give it raining. Its called as grey. Ko histogram. Weekend. Essentially give it a label. Across the X axis and we're gonna call this bins. Let's give this a while. Label. And set this equal to the number of pixels. The number of pixels. Let's go in label. And 5 we can plot plot the. The grayscale histogram. In Delhi, we can essentially give it a limit across the X axis. So PLTS link of a list of 02256 and finally we can display this image to PL TV show. Say that North Run Python histogram to PY and this is the distribution of pixels in this image. As you can see, the number of bins across the X axis basically represent the intervals of pixel intensities. So as you can see that there is a peak. At this region this means that this is closely at 56 ish. So this means that in this image there are close to 4000 pixels that have and intensity of 60. And as you can see that there is a lot of, there's a lot of peaking in this region, so between. Probably 4270. There's a peek pixel intensities or closed you 3000 pixel intensities in this image. So let's try this with a different image. Let's draw in this with their cats.\_2. Say that and run. And there is a be king of pixel values in between 202 25 and this makes sense because most of the images white. So given that reason you can probably deduce that there will be a picking towards white or 255. So this is essentially computing the grayscale histogram for the entire image. What we can do is we can essentially create a mask and then could be the histogram only on that particular mosque. So let's do that. Let's go back to your masking. Let's. It's rap in this, scrap this. It's alright up there. I said this to you. Image don't shape of the first 2 values, sizes of the same let us. Essentially draw a mask which will be circle of. Call blank and we can get the sensor of image in shape of one by divided by 2 an image don't shape of zero divided by divided by 2 give the radius of 100 pixels give you the colour of 255 give you the thickness of -1 week an display a mosque. Let's call this. Of course mask, passing mask, and here's what things get interesting where you complete the grayscale histogram for this mask and we do that is by setting this mask parameter to mask. So instead of non, reset this to mask and let's see what that does to are. Histogram MP is not defined correct. Children rhymes. Can a mistake here? Ohh, that's right. This is the last. This is not exactly the masses circle. This is. This will be a circle. Circle and essentially we need to mouse count the image so we so we do that is better created mosque in telling this equal to CV dot bitcoins. Peter boys on scope and we can pass in the grayscale image to grayscale image and we can pass in the mask which is equal to circle. Now we can use that as the mosque. So let's. Yes, X. Sorry mistake but only thing should be fine right now. So this is the mosque and this is the histogram computed for this particular mask. As you can see that there is a P King of pixel intensity values in this region and there are. Smoking in in these regions dhampur let's try this with another image that spas in weekends. Accounts 2. Accounts to jpg. This is a mask and this is the. There is a Peking in this image. Ram towards 50. Okay so then visit for computer in Grayscale histograms. Let's move on to try to compute a colour histogram that is true. Computer histogram for colour image to an RGB image. So thats called as colour histogram and the way we do that is instead of converting this. Mr Grey scale, let's. Comment all of this out. We will use the mask, but later. Let's. Select all of this out. Face mask by 5 GG. Ncr's pretty much. So what's up with the colour histogram? The way we do that is. Let's define the table of colours in South. Is equal to. B. Then. Table of GHU elements of R and what I'm going to do next is I'm gonna say for I'm home a call in enumerate of colours. What I'm going to do is going to say his. So I'm gonna pop histogram. I'm saying TV dot calc. Hist going to compute it over the image itself? The channels will be I. That means this I over here we are going to provide a mask of non from, give the his size of 256 and give it a ranges of 02256. And then let's do a PLT dot plot. Ahista and give it a colour. Ecourts you call and only we can do a PLT. Excellent. Of 02202256. And for this purpose we can essentially grab. Wish copy bat. On common base out and we can do a PLT don't show. So this should work. Webmusic something I don't think we him were not were not computing this histogram for a mask or will do that next, but lets say that everyone. Cool and lets close it up. I mean mistake. This is a colour. Should make much of a difference, but here. So this is the colour histogram that we get for the original image, not from ask, but in fact this image. So as you can see that this colour image basically computed the plot for Blue channel, the Red Channel and the green Channel as well. So using this you can basically make out that there is a P king of blue pixels that have a pixel intensities of 30. There's a P King of Red, probably run 50. P King of Green will be around 1875 to 80. I'm using this. You can basically make up the distribution of pixel intensities of all 3 colour channels. So let's try and apply a mask by sending this equal to in mosque. Let's see you whether we have everything in order. This is bitwise and mouse, most most famous mosques say about Android. Laskar not the same size? Okay, I finally got the error. So basically them as needs to be a binary format. So instead of passing in this mask. This was should be the masked. Last image begin houses make that mask and we can change the circle to mosque. Now this should work without any. Errors. And we can change that to masked. Yeah that's remember and now we get the colour histogram for this particular mosque. I made a mistake because I use this as my mask to compute the histogram 41 channel. The problem was this masked image was actually a 3 channels and I attempted to use this these 3 channel. Mask to calculate histogram per channel, which isn't allowed in open CV. So that was my mistake. What kind of user on variable names. So confused. But essentially this is it. Your computing the histogram for a particular section of this image and this is a good to get. There is a high Peking of red in this error, high Peking of blue in this error in high Peking off. Green some over here Sir, its actually thats it. But this video history brands actually allow you to analyse the distribution of pixel intensities, weather for grayscale image or or a coloured image. Now these are really helpful in a lot of advanced computer vision projects when you actually trying to analyse the image that you get and maybe try to equalise. The image search new Peking of pixel values here and there in the next video we talk about how to threshold in image and different types of thresholding. As always, if you have any questions leave them in the comments blow. Otherwise I'll see you guys in the next video. Everyone, and welcome back to another video. In this video, we're gonna be talking about thresholding in OpenCV. Now thresholding is the binarization of an image. In general we want to take and image and convert it to a binary image. That is an image where pixels are either zero or black or 255 or white. Now a very simple example of thresholding would be to take an image and take some particular value. Yeah, I'm gonna call the threshold value and compare each pixel of image through this threshold value. In that pixel intensity is less than this rational value, we set that pixel intensity to zero and and if it is above this threshold value we set it to 255 or white. So in this sense. We can essentially create a binary image just from a regular stand alone image. So in this video we're actually going to talk about 2 different types of thresholding, simple thresholding and adaptive thresholding. So lets start up with simple threshold. So in essence what I want to do is before I talk about simple thresholding is that want to convert this BG image to grayscale. So I'm gonna say grey is equal to CV CVT column. We pass in the image, we pass the colour code which is. Bgr. 2 great, we can just play this image. Call this Gray we can pass in, Greg. Cool. So lets start up with this simple thresholding. So essentially 2 to apply this this idea of symbol thresholding we essentially use the seaweed of threshold function. Now this function returns threshold and fresh which is equal to threshold. NDC nations takes in the grayscale image to. Grayscale Image has to be passed in to the threshold in function. Remove we do is we pass in a threshold value. So let's at this 2150 for now we have to specify something called a maximum value. So if that pixel value is greater than that is greater than 150, what do you want to set it to? In this case we want to validate the image, so we sent it to 255. Money, we can specify a thresholding type. Now this thresholding type is essentially CV dot thrush.\_by Eric and what this does is it basically it looks at the image, compares each pixel value to this threshold value and if it is above this value its its its 2255 otherwise it infers that if it follows blow it sent it 20. So essentially returns 2 things, trash which is the threshold, the image of the bio dies image and threshold which is essentially the same value that you passed 150 at the same threshold value passing will be returned to this threshold value. So what actually display this image? So let say see video time show will call this. Threshold. Milk all this simple threshold did and we can pass in fresh. So lets say that in run Python crash. Top EY and this is a threshold with image that you get. Okay, this is nothing 2 different from when we discuss thresholding in one of the previous videos, but this is essentially what do you get. So let's play around with these threshold values. Let's set this 200 and let's see what that does. And as a result, board parts of the image has become white. So and of course if you get a higher value less part of the image will be white. So lets have this 2225. And very few pixels in this threshold image actually have a pixel intensity of greater than 225. So what we can do after this is essentially create an inverse threshold image. So what we can do is we could essentially a copy this. And instead of saying Thresh I'm going to say thrush under school in this Ain I'm going to leave everything else same. Lets have this do 150 and the same thing here and instead of passing in the type of thresholding I'm gonna sexy video thresh\_binary under scope inverse and let school this threshold. Inverse, and we can pass in inverse. So I say that and run. And this is essentially the inverse of this image. Instead of saying pixel intensities that are greater than 150 to 255, it says whatever values that are less than 150 to 255. So that's essentially. What do you get? Right hold the black part of this image would change to white and all the white parts of the image will change to black. Cool, went into simple threshold. Let's move on now. 2 adaptive threshold. Adaptive thresholding. Now as you can imagine, we got different images when we provided different threshold values. Now kind of one of the downsides to this is that we have to manually specify a specific threshold value. Now some cases this might work. In more advanced cases this will not work. So one of the things we can do is. You can essentially let the computer find the optimal threshold value by itself, and using that value that finds it, banner rises over the image so that senescence the entire crux of adaptive thresholding. So let's set up available call adaptive\_with thrush in said this equal to CV dot adaptive. Threshold an inside I want to pass in a source image. So let's set this to your. Great. I'm gonna pass in a maximum value which is 255. Notice there is no threshold limit value, it happened. Method basically tells a machine which method to use when computing the optimal threshold value. So for now we're just gonna set this to the min of some neighbourhood of pixels. So let's chat this. Do you see if you don't? Adaptive on this code thrush and scroll mean\_C. Next, we'll set up a threshold type. This is CV dot thrush.\_binary again I think 2 different for me. This from the first example and 2 are the parameters that I want to specify. Is the block size which essentially the neighbourhood signs of the kernel size which opens. He needs to use to essentially compute the mean to find the optimal threshold value. So for now lets have this too. Fill later. And finally the last method we have to specify is the C value. Now this see value is essentially an integer that is subtracted from the mean allowing us to essentially find June or threshold are. So again don't worry too much about this. You can set this to zero but now it's at this 23. Once that's done, we can go ahead and try to display this image. So let's call this adaptive thresholding and we can pass in adaptive crash. So lets say that Enron. And this is a sensory yo adaptive thresholding method. So recently what we've done is we've a defined a kernel size or window that is drawn up this image. In our case this is 11 by 11. Answer What you can see it does is essentially computes amine with this neighborhood pixels and find the optimal threshold value for that specific part and then it slides over to the right and slides. Lt does the same thing and its lying to down and ask the same thing so that it is actually slides over every part of the image. Select how adaptive thresholding works if you want to find June this week could change this to a threshold scope. Binary and school inverse. You're just to see I was really going on under the hood. Cool. So all the white part of the image will change to black. Anal black parts of the image of a changed quite. So let's play on with these values. Let's set. Festool, probably 13 and see what that does. Okay, definitely some difference from the previous hyper parameter. So let's try it. Let's go with. Let's have this 211 in. Let's set this to me by one. Okay, definitely more white. That said this to me by 5. Really you can play around with these values right? The more you subtract from the mean, the more accurate is right? You can basically make out the edges now in this basket. So what's may be increased 829? And you got less white spots in the image, but it's actually now you can make up the features better. So that was essentially adaptive thresholding, adaptive thresholding that essentially computed the optimal threshold value on the basis of the mean. Now we dont have to stick with amine, we can go with something else. So instead of Millet set this 2 Gaussian. So what? Say that and see what that does. And this is the threshold the image using the Gaussian method. So the only difference that ghost in applied with essentially add a wait to each pixel value and computer the mean across those pixels. So thats why we were able to get a better image. And when we use mean, essentially the double threshold mean works in some cases, the Gaussian works in other cases. There's no real one. Science bits also really. Play around with value, see what you get. But that's essentially all we have to discuss. For this video we talked about 2 different types of thresholding, simple thresholding and adaptive thresholding. In simple thresholding we have to manually specify a threshold value and in adaptive thresholding, open CV does that for us using a specific. Block size or current size? Another computing the threshold value on the basis of amine or on the basis of the Gaussian distribution. So in the next video, the last video in the advanced section of this cause, we're going to be discussing how to compute gradients and edges in an image. So if you have any questions, leave them in the comments below. I'll be sure to check the mark, otherwise I'll see you guys in the next video. Thanks for watching. Everyone, welcome back to another video. In this video we're gonna be talk about a gradient and edge detection in OpenCV. Now you can think of gradients as these edge like regions that are present in an image. Now they're not the same thing. Gradients and edges are completely different things from a mathematical point of view, but you can pretty much. Getaway with thinking ingredients as edges from a programming perspective. Moment. So its not showing in the previous videos we've discussed the canny edge detector which is essentially kind of an advanced edge detection algorithm that is essentially a multi step process. But in this video we're gonna be talk about 2 other ways to compute edges in an image and that is the like placing and the circle method. So let's start up with the Laplace you. So the first thing I want to do is I want to convert this image to grayscale for calling the CVT. See video call colour method. We pass in the image and we say Cedar colour\_BGR to great. We can display this image called this Gray and we can pass in. Everything awesome. Great. So that started with a lot placement, so we're gonna define a variable cold lap in said this equal to see we dont lab lesson. And what this essentially will do is I will take in a source image which is great now and it will take in something called ADI depth or data depth. Now we will sent this to see we dont 64 it is full along with whatever I doing next I'm going to say it lap is equal to MP don't you 98? And insert. I'm gonna pass in MP Absolute and we can pass in lap. And since I'm using dump I I can actually go ahead and port number. As MP. And when I go to display this image kaunasee dollar I'm sure method is called a slap wesha and we can pass in lap. Collapse, save and run Python gradients PY. Invalid syntax if I don't. Okay, its CV dots CV on score 64 F. Send apps. And this is essentially the Laplace in edges in the image car looks like an image that is grown over shop Korean dance man just a bit. But anyway, this is the vibration method. Let's try this with another image. Let's try this word. Hawk. Of course, personal school is the park. Caesar in run. And this essentially it looks like a pencil shading of this image to all the edges that exist in the image. By these most of the edges in the image are essentially drone over with pencil and then likely submerged. So that's essentially the Laplace and edges you could say. So again don't worry too much about why we converted this to renew it. Then we computed the absolute value. But essential application method computes to gradient of this image to grayscale image. Generally this involves a lot of mathematics but its not showing when you transition from black to white and white to black, that's considered a positive and negative slope. Now images itself cannot have negative pixel values. So what we do is recently compute the absolute value of that image. So all the pixel values of the image are converted to the absolute values and then we convert that to a UNT 8. So in image specific data type. So that's basically the crux of what's going on right here. So thats want to the next one that is the subil gradient magnitude representation. So essentially the way this does is that so will compute the gradients in 2 directions, the X&Y. So we're going to say Sobel X which is the gradient that are computed along the X axis. Insert is equal to UC. We do not so. And we can pass in the image flight status to be crazy image we pass on the data depth which is CV CV on school 64. And we can give it a next direction. So let's set this 21 and y direction. We can set that to zero. Let's copy this. And cold sober. Why? Instead of one zero, we can save zero one and we can visualise this. Let's print. Lets hold miss Herbal X and we can pass in sobo X and we can see the alarm showing Sobel. Why? And surface to SAB OY. Collapse. And these are essentially the gradients that are computed. This is over the Y axis so you can see a lot of why horizontal specific gradients and the SAB LX was computed across the Y axis so you can Cy axis specific gradients. Now we can essentially get the combined stubble. Mg by central combining these 2 so blacks and Sawai and the way we do that is we can say combined on combined\_subil and set this equal to. CV drop bitcoins on screw or and we can parcel sober X and Sabo. Why? Yeah, we can display this image. So let's. Call Nayan show we get to combined. So. And we can pass in the combined Sobel. What's wrong are. Eris, essentially the combined Sobel let you get. It is an. Let's go back here. Siri said she took please do appliance bitwise or an essentially got this image. So if you want to compare this with a lap basin. 2 completely different algorithms, so so the result you get will be completely different. Okay, so let's compare both of these location and the Sobel with the canny edge detector. So let's go down here. Lets say Kenny is equal to UTV Dot Kelly. And we can pass in the image. Let's possible grayscale image. Let's give it to threshold values of 150 and 175. And we done thats display this image. School is Kenny we can pass in Kent. So both that gives us. So let's compare that with hi here, here, here. So that's essentially it. This is the Laplace in gradient representation which essentially returns kind of this pencil shading version of the image of the edges in the image. Combined, Sobel computes the gradients in the X&Y direction and we can combine these 2 you with bitwise or Ankeny. It's basically a more advanced algorithm that actually uses Sabo in one of its stages. Like I mentioned, can I is a multi stage process and one of its stages. Is using the Sobel method to compute the gradients of the image. So essentially you see that the Kenny Edge detector is Amole clean of version of the edges that can be found in the image. So thats why in most cases you're going to see the can be used. But in my advanced cases you're probably going to see a sole use a lot, not necessarily libration but so definitely. So that's pretty much in to this video. In fact this video concludes the advanced section of this coast. Moving on to the next section, we will be discussing face detection and face recognition in open CV, actually going to touch on using Haar cascades with to perform some face detection. And in face recognition, we actually have 2 parts of face recognition with OpenCV built in face recogniser and the second part will be actually building own deep learning model to recently recognise some faces in an image. Again, like always if you have any questions leave them in the comments below, otherwise I see you. Guys in the next section. Everyone, and welcome back to another video we are now. It's the last part of this Python in OpenCV coast where we are going to talk about face detection and face recognition in open CV. So what we're gonna be doing in this video is actually discussing how to detect faces in urgency by using something called. R cascade. In the next video we will talk about how to recognise faces using OpenCV build in face recognise and after that we will be implementing Aaron Deep learning model to recognise between the Simpson characters. Were going to create that from scratch and use OpenCV for all the preprocessing. And displaying of images and stuff like that. So let's get into this video now. Face detection is different from face recognition. Face detection merely detects the presence of a face in an image, while face recognition involves identifying whose face is lives. Yeah, will talk more about this later on in this course, but essentially face detection is performed using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive or negative, whether a face is present or not. Now close find needs to be trained in thousands and 10s of thousands of images with and without faces. But unfortunately for U.S. Open see we already comes with a lot of pre trained classifier that we can use in any program. So essentially the 2 main classifier that exist today, arhar cascades and more advanced classifiers call local binary patterns. We're not gonna talk about local binary patterns at all in this course, but essentially there more advanced hall cascade classifiers, they don't ask prone to noise in an image as compared to the heart cascades. So IM currently at the open seas get Home page where they store there. Haan Cascade there Hall casket cost of fires and as you can see there are plenty of Haar cascades that opens up mix available to the general public. You have a heart cascade for an eye friendly cat face, frontal best default full body your left away a Russian license plate, Russian plate number I think the same thing ah Ha Casca to detect smile hawkers get for detection of. The upper body and things like that. Feel free to use whatever you want, but in this video we're gonna be performing face detection and for this we are going to use the haar cascade on school frontal paste on the score default data XML. So when you go ahead and open that you're gonna get about 33,000 lines of XML code. So all of this. So what you have to do is essentially. Go to this role button annual get all this raw XML code olive to do is click control orca Monday if you're on a Mac and click control C or command C and then go to your VS code or your editor and create a new file and we're gonna call this horror and scary face.in XML. And inside this I want peace in those 33,000 lines of XML code. Go ahead and say that ANTA classifier is ready so we can go ahead and close this out. So we're gonna be using this har cascade classifier to essentially detect faces that are present in an image. So in this file call face detect face\_detector, pie imported open CD. I'm basically read in an image of a lady, a person that is this. Image over here and we can go real quick and slave is. So let's run Python face. Face on the score detect. WR and we get a image in a new window. Cool. So lets actually implement barcode. The first thing I want to do is convert this image to grayscale. Now face detection does not involve skin tone or the colours that are present in an image. These are cascades. Essentially look at and. Object in an image in using the edges tries to determine whether its a face or not. So we really don't need colour in R image and we can go ahead and convert that to grayscale. TV DOT TV T Kholo Pass on the image in BGR to grow and we can display this call is grey. Call this great person. Would impasse in IMG let's say, then and run. And we have 2 parcels, the great. Okay, we have a great person over here. So let's move onto essentially reading in this horoscope based on XML file. So the way we do that is by essentially create a haar cascade variable. So let's set this too hard. On the scroll cascade I'm going to set this equal to see we dont cascade. Classifier in inside what ISN shall I want to do is is passed in the path to this heart. To this XML file so that is South Simple Singh har on square face that XML. So this kaise classifier class will essentially read in those 33,000 lines of XML code and store that in a variable kolhar\_cascade. So now that we've read know are cascade file, let's actually trying to detect the face in this image over here. So what I'm going to do is essentially say faces on scroll rect is equal to har\_cascade detect multiscale and Intel gonna pass in the image that we want to detect face on. So this is great, we're gonna pass on a scale factor. Now lets have this 21.1. Give it a variable cold minimum neighbors, which essentially is a parameter that specifies the number of neighbors or rectangle should have to be called a face. So let's at this 23 phone app. So thats it, thats all we have to do. An essentially what this does is this detect multiscale. An instance of the Cascade classifier class will essentially taken this image, use these variables, call scale factor and minimum neighbours 2 assembly, detect a face and return essentially the rectangular coordinates. Fat face have a List 2 faces\_wrecked. That's exactly why we are giving it faces on school fete rat to rectangle. You can essentially print the number of faces that were found in this image by essentially printing the length of this faces on this corrective variable. So let's do that. Let's print. Say number number of faces. Sound is equal to. Impossible length of fish is on this correct? So lets say that in Ron and you could see that the number of faces that were found one. Thats true because. There is only one person in this image. Cool. Now utilizing the fact that this faces on school rect, he's essentially the rectangular coordinates for the faces that are present in the image. What we can do is we can essentially loop over this list and essentially grab the coordinates of those images and draw a rectangle. Over the detected faces. So let's do that. So the way we do that is by saying 4X com Wycombe WAH. H in. Faces on this correct? What we're going to do is going to draw rectangle. Serial rectangle. Over the original image, so IMG give the 0.1. This 0.1 is essentially X. And 0.2 is essentially X plus W, Why plus H? Let's give the colour at this to green, 02550 give it a thickness of 2. Thats it and we can. Print this and we can just like this image. So let's add this to you. Detected basis. And we can person or mg. If you look at this image you can essentially see if the rectangle that was drawn over this image. So this is essence is the face that open saved Har cascades found in this image. So let's try this with another image. So what I have hear a couple of people, couple of other people and image of 5 people. So we're going to use that image and trying to see how many phases are cascades could attempt in this image. So lets have this 2 group 2. And we can change that to your poop. Of by people say that holds great people. 71. Yeah, I wanna point real quick that the number of faces that were found who actually 7 and we know that there are 5 people in this image so lets actually see what happens if I thought was a face so we can go real quick. So actually detected all the phases in this image, all the 5 people. But it also detected to other okay so stomach in part of a neck. But this is to be expected because Haar cascades are really sensitive to noise in an image. So if you have something that pretty much looks like a face like her neck. Looks like a face. It has the same structure as a typical face would have. I know why his stomach was recognises face, but again this is to be expected. So one way we can try to minimise the sensitivity to noise is essentially modifying these scale factor in minimum neighbors. So let's increase the minimum age to me by 6 or 7. Say that in run. In educate see now 6 faces were found. So I guess by increasing the minimum neighbors Brahman OVA sensually stop open CV from detecting her stomach as face. Try this with on another more complex image. Couple of people in Group one. So I change that to Group One. Same right? Now you can see that the number of faces were never found was 6 and we know that this is not 6. So lets actually change this minimum, minimum neighbors just a bit a changes first 23 and see how many faces were found. How we got 14? Okay, some people at the back want chosen because either the faces are not perfectly perpendicular to the camera or they wearing some accessories on the face, for example, eyeglasses. Do you wearing hat? This is doing to bring out cap and stuff like that. So lets actually change this to one and lets you with that gets change that to one, so run. And now we got 19 faces that was found in this image. So if something about looping through these values by changing these values, by tweaking these values you can assumption get a more robust result. But of course bye bye minimising these values yourself for making open CVR Cascades more prone to noise. So thats a trader. You need to consider now again, heart cascades are not the most effective in detecting faces that popular, but they are not are the most advanced. They are probably not what you would use if you were to build more advanced computer vision projects. I I think for that D lips face recogniser. Is more effective in less sensitive to noise then open CBSR cascades it as per your news case. Hard case kids are more more popular, easy to use and they require minimal setup. And if you wanted to extend this to videos you could all you have to do is essentially detect. Hot cascades on each individual frame of a video. Now I'm skipping that because it's pretty self explanatory. So that's pretty much it. For this video we discussed how to detect faces in urgency by using OpenCV Haar Cascades. In the next video we will actually talk about how to recognise faces in open CV using open CVS bills on face recogniser. So like always, if you have any questions, comments, concerns, whatever, leave them in the comments blow. Otherwise I'll see you in the next video. Everyone in welcome back to another video. In this video we'll learn how to build a face recognition model in open CV using OpenCV build in face recogniser. Now in the previous video we dealt with detecting faces in open CV using Haar Cascades. This video will actually cover how to recognise faces in an image. So what? Have you are 5 folders of 5 different people. Inside each folder I have about 20 images of that particular person. So Jerry has 21 images. Amazon hair 17 Mini killing has 22. Ben Affleck has 14 answer what So what I'm essentially gonna do is it gonna use open CVS bills in face recogniser and we're going to train that recogniser on all of these images in these white folders. And this is sort of like building a mini sized deep learning model, except that when I'm gonna build any model from scratch we're gonna use open TVs bills in face recogniser or gonna do is actually going to pass in these close 90 images. And we're going to train that recognise are all these 90 images. So let's create a new file and going to call this faces Amsco train dog pee. Why? We're going to import OS are going to import CD, 2 hour see we are going to import Numpy as MP. So the first thing I want to do is essentially create a list of all the people in the image. So this is essentially the names of the folders of these particular people. What you can do is you can manually type those in or you could essentially create an empty list. Let's call with P and we can loop over every folder. In. This. Folder. Not at this to ashram and we can say P dot append only we can print P. Say that him run. Python places on screw and skirt trend one. Can we get the same list that we got over here? So thats one way of doing it. And what do I do next? Is a messenger going to create a variable colder and set this equal to this base folder. That is this folder which has which contains these 5 folders of these people? Cool. So we had done. What we can do is we can essentially create a function called death create unscrew train. There will essentially loop over every folder in this base folder, and inside that folder it's going to loop over every image and essentially grab the face in that image and essentially add that to a training set. SOA training set will consist of tulips. The first one is called features which are essentially the image to raise of face is. So let's set this to an empty list and the second list will be are corresponding labels. So for every face in this features list, what is its corresponding label whose face? Doesn't belong to. Like one image could belong to Ben Affleck, the second image could belong to Elton John and so on. So let's create a function. So we're going to say going to loop over every person in this people list. Who can grab the path for this person still for every folder in this BC full are going through each folder and grabbing the path to that folder. So that's essentially as simple as sing Koester path, don't join, don't join and we can we can join the dare with person. And what I'm doing, create a labels label very well and set this equal to you people dont index of person. And now the way inside each folder we are going to loop over every image in that folder. So we're gonna say for image to image in OS start list Dir in Puff. We are going to grab the image path, so we use image unsquare path is equal to OS path join. Let's go to sign join. We're gonna join the path variable to the inch. How do we have the path to an image? We're going to read in that image from this path. So we're going to create a variable call on Jian school ring. Is equal to. In read image on the scoop of were gonna convert this image to grayscale saying CBT colour pause annangi on school right and we can pass in see we dont see see we dont colour on the screw BGR to grid. Now now with that done, we can essentially trying to detect the faces in this image. So let's go back to face\_detect and grab the har cascade classifier variable. Hear that space that there? And we can create 8 of faces on scroll rect and said is equal to Harpers Ko Cascade don't detect multiscale. This will take in the great image is scale factor of 1.1 and a minimum neighbors off. How we can loop over every every face in this face direct. So 4 or X com and white comedy comment each in face is wrecked. We are going to grab the bases region of interest and set this equal to and basically crop out the face in the image. Don't say qua whi 2, why? Plus H&X 2X plus W. And now that we have a face is a face region of interest, we can append that to the features list and we can depend the corresponding label to the labels list. So we gonna do features dot append. Going to pass in faces on school ROI and we can do a labels pinned. Label. Yes, label variable is essentially the index of this list. Now the idea behind converting of able to numerical values is essentially reducing the strain that your computer will have by creating some sort of mapping between string and the numerical label. Now the mapping of we're gonna do is essentially the index of that particular list. So lets say that I grab the first image which is an image of Ben Affleck. The label for that would be zero because Ben Affleck is at the Earth index of this people list. Similarly Elton John. Image of Elton John would have a label of one because it is at the second position or the first index in this peoples list. So that's essentially the idea behind us. Now with that done we can essentially trying to run this and see whether we got any errors or not and we can bring the length of the features. So lets say length. Point. Of the features, worst is equal to. The length of features. And we can do the same thing thats. It was copy this. Blank sample labels list sentence to write the labels. So that shouldn't give us any error. So what's wrong? That and we get the length of the features 100 and length of the labels 100. So we sent you what we have 100 face is an 100 corresponding labels to those faces. So we dont need this anymore. What we can do is we can essentially use this features and labels list now that it's appended to train our recognise or on it. So the way we do that is we instantiate our face recogniser. Call is as an instance of DOT face dot. LBPH face recogniser\_create class. Hey Vishwa century, instantiate the face recogniser. Now we can actually train the recogniser. On. On the features. List and labels. And the labels list. So the way we do that is breaking faisons correctness of dot train. And we can pass in the features list and we can pass in full labels list. And before I should do that, I do want to convert this features and labels list to Numpy rates, so we're gonna do. So we're gonna say features is equal to MP dot ray of. Features. And we can say labels is equal to MP dot ray labels and see once Indrani Okay do you have objects? WhatsApp this to detect object. Horse in Hindi type. SQL object. And we can actually. Print When this is down. So lets say training dump. And we can actually go ahead and save this features and labels list. I'm gonna say MP dot save are going to call this features dot MPY and we can pass the features. And we get to impedance happy dogs name labels don't MPW and we can pass it the labels. So let say that him run. Cool. So essentially now the face recogniser is trained and we can now use this. But the problem here is that if we plan to use this face recogniser in another file, we will have to separately and manually repeat this process. This whole process of. Adding those images to a list and getting the corresponding labels and converting that to number as and then training all over again. What we can do and what open CV allows us to do is essentially save this trained model so that we can use it in another file in another directory in another part of the world. Just by using that particular Yamil so smart. So we're gonna repeat this process again, but the only change that I'm going to do is I want to say face, recognise the dots safe and give the path to a Yamil sex while we're gonna say face on screw trend dault Yamil. So let's repeat this process again brings down. And now you must that you have a face on scope trained their file in this directory as well as faces as well as the features that MPW and labels. So lets actually use this train model to recognise faces in an image. So lets close this out. Create a new file and get a call this face on school recognition. Very simply, we going to import Naam Hai. As MP in CB 2 AC we we dont need OS anymore because we are not looping over directories. We can essentially create or her on this code cascade file. So let's do it. Let's go up here. Crap this. We can load features and label re using Isai. Features is equal to load features dot MPY. How we can save Ables? Is equal to MP download. Let's call this labels Dot MPY and we can essentially now read in this face on this code train that yellow file. So let's go over here, let's grab this line. And lets say face recognise a dog breed and gonna give me the path to the Zee Anmol source file. So face on face on the screw trend don't I am. So that's pretty much all we need. Now we need to get the mapping, so let's grab this list as well. Then. So that's pretty much all we have to do. So let's create a variable image that said this. Do you see if you don't M read, give it a path. What's create that's gram one from this validation? Abuan, from which gravitation have one of Ben Affleck's. So let's try this with grab that peace out there. Angrand in this image. Prabhat. Hello peace out there in out to JPG file. Yeah, we can convert that image to grayscale. TV DOT TV T colour palette image CV no colour\_bheegi bheegi are too great. Search despite cement C kolah the person on identified person, it's painting. So what we gonna do is we're going to first detect the face in the image. So what we do that is by saying features on on scroll rect is equal to her\_Cascade dot detect multiscale we pass in the Gray image we passed in the scale factor. Which is 1.1. Give it a minimum neighbors of FO and we can leave over every face in this face isn't correct. Soo. 24 XH in faces. Faces on correct. We can grab the region of interest towards interested in finding. JY 2 why plus H and? X2 X plus H. And now we can predict using this face recogniser, so we get the label Anna confidence of value. And we say face recogniser dot predict. And we predict on the spaces on school ROI. Let's print. It's called. This label is equal to. Memo. With. Confidence of. Confidence. And since we are using numerical values, we can public. We can probably say people off label, okay, and we can. Essentially what can do is we can put some text on this image just to show us what's really going on. We can put this on the image. We can create a string. Verbal of people. Of label so the person involved in that image given in origin or say 10. Sorry, 20 by 20. Give the phone face of CV dot font on. Screw her, she on school complex. Give it to. Phone scale of one point of 1.08, colour of zero, 2550. And give it a thickness of 2 and we can draw rectangle over the image. Over the face. Yeah, this is we do not disturb me image. We give it X2 Y&X plus W call Y plus each. We give the colour of 0255, zero. Yeah, we can give it a thickness of 2. So with that done we can found this display this image call does the detected. Please. We can pass in the image. And finally we can do a CB DOT wait Kee Zero. So lets say even see what we gets Python. Python face. Funskool Rukh egg finish. Nation like you, I cannot be allowed in love. Pickles equals fools Ghosh. Where is that? You probably don't need this anymore, so lets come at that out. That if you wanted to use these again you could essentially use MP download since the data types objects you can basically stay allow pickle is equal to true. Taxes whenever user so lets come at that out so. Okay, we get Ben Affleck with a confidence of 60%, so that's pretty good. 60% is good given the fact that we only train in this recogniser on 100 images. So can try this with another image of Ben Affleck. Maybe this image copy of that right across here. Haan. Where? Okay, this again is Ben Affleck with confidence of 94%, pretty good. Let's go back. Let's go to maybe to another person. Let's go to McDonnell. Let's grab this. It's a pain reliever. I will strangers to Madonna. Hey, let's grab. This person am not sure whether able to take the face. AA. Because of the hair. How much space that anyway? Now this is where you find that open TV is best recogniser built on face recogniser is not the best it currently detects. It currently detects that this person in the image is actually Jerry Seinfeld and that too with confidence of 110%. Maybe there's an error somewhere, I'm not sure why they went to 111. But pretty sure there's an error somewhere. But essentially this is where the discrepancies lights. Not the best so its not gonna give you accurate results. So let's try this with another image. Let's go about to maybe so much. Copy that. Okay, this is Madonna with confidence of 96.8%. Okay, let's move onto Elton John. Watson had problems with Elton John. I'm giving the fact that he look pretty similar to Ben Affleck for some reason. Copy that changed at you. Helps in on school, John. Imprint thats. Okay else, enjoy the confidence of 67%. Pretty good, okay, so not bad. This is more accurate than what I predicted. Before filming this video, I did a couple of trial runs and I got very results. For example, Elton John was continually detected as Jerry Sine field. Ben Affleck Madonna was detected outspending work Ban Affleck was detected as Mindy Kaling, mini killing was detected as Elton John and whole bunch of weird results. So I guess that we did something right. I must have done something wrong in the trial runs boat. We get good results and that's pretty good. I'm not sure why that gave a confidence of 111%. Maybe there's an hour somewhere with the training sense, but I guess for the most part you can ignore that. Keeping the fact that we get pretty good results. So that's pretty much it for this video we discussed face recognition in open CV V essentially built are features list and labels this and we train a recogniser on those 2 list and we save a model. As I am also smiling in another file we essentially read in that saved model. Saved Yamil so smile and we essentially predictions on an image. Henson in the next video, which will actually the last video in this course, we will discuss how to build a deep learning model to detect and classify between 10 Simpson characters. So if you have any questions, comments, concerns, whatever, leave in the comments below. Otherwise, I'll see you in the next video. Everyone and welcome to the last video in this Python And open TV coats. Previously we have seen how to detect and recognise faces purely in open CV and the results we got married. Now there are a couple of reasons for that. One is the fact that we only had 100 images to trains recogniser on. This is a significantly small number, especially when your training recognizers and building models idea that you don't want to have at least a couple of 1000 images power class. The second reason lies in the fact that we weren't using a deep learning market. Now you got deeper into especially computer vision. You will see that there are very few things that can actually beat a deep learning model. So thats what we gonna be doing in this video building a deep computer vision model. Do you classify between the Simpson characters? Now January open TV is used for preprocessing the data that is performing some sort of. Image normalization mean subtraction and things like that. But in this video we're gonna be building a very simple model. So when I'm gonna be using any of those techniques, in fact we only by using the open CV library to read an image and resize them to a particular size before feeding it into the network. Now don't worry if you've never used to built a deep learning model before, but this video will be using tensor close implementation of carers. Now I want to keep this video real simple just so you have an idea of what really goes on in more advanced computer vision projects. In character comes with a lot of boilerplate code, so if you never built a deep learning model before. Don't worry, carriage will handle that for you. So kind of one of the prerak visits to building a deep learning model is actually having a GPU. Now GPU is basically a graphical processing unit that will help speed up the training process of a network. But if you don't have one, again don't worry because will be using candle. A platform which actually offers free GPS for us to use. Real simple. Before we get started, we need a couple of packages installed. So if you haven't already installed Sarita beginning of this coast, go ahead and do a PIP install Siri. The next package you require is Canaro, and this is a package that I built specifically for deep learning models build with Keras, and this will actually appear surprisingly useful to you if you plan to go deeper into building deep computer vision models. Now, installing this package on your system will only make sense if you already have a GPU. Play machine. If you don't, then you can basically skip this part so we can do a PIP install Karo. Install tensor flow by default, so just keep that amount. So with all the installations out of the way, let's actually move on to the data there going to be using. So they said they were going to be using is The Simpsons character dataset that's available on Kaggle. Show the. So the actual data that were interested in lies in The Simpsons on Scott dataset folder. This basically consists of a number of folders with several images inside each subfolder. So Maggie Simpson has about 12. 128 images whom Simpson has about 2200 images, Abraham has about 913 images. So basically we do is we are going to use these images and feed them into R model to recently classify between these characters. So. I think I wanna do is go to cancel.com slash notebooks, go ahead and create a new notebook. And advanced settings make sure that the GPU is selected since we're gonna be using GPO after that click creates. And we should get a notebook. So we're gonna rename this to Simpsons. Aaron, nothing I want to do is enable the Internet, since we're gonna be installing a couple of packages over the Internet. So to use The Simpsons character dataset in R notebook, you need to go ahead to add data search for Simpsons. In the first one by Alex ITR should pop up. Go ahead and click add. And we can now use this data set inside on notebook. So the first thing I want to do is going to PIP install. Share. Now the reason why I'm doing this yet again. Now the reason why I'm doing this again is because candle does not come pre installed with CRN Canary. Now I did tell her to install it on your machine and the reason for that is because you all can work with it an experiment with. So once its done go ahead to a new cell and lets import all the packages that. Open it. So we're gonna import OS, we're going to import Sierra, going to import Karo. Going to import dump I is NP import CD 2 and CV and going to import GC or garbage collection. When next one one do is in. Basically when building deep computer vision models, your model expects all your data or your image data to be of the same size. Soil sensor working with image data this size is the image science. So all the data all the images in Rdlc set. Well, actually have to be resized to a particular science before we can actually feed that into the network. Now with a lot of experiments, I found that an image size of 80 by 80 works well, especially for this Simpsons dataset. Okay, the next variable we need is channels. So how many channels do we want to know? Image. And since we do not require colour image, we're going to set this 21. Basically craze game. So run that. Movie next is going to say Karana scope of. Is equal to the base path where all the data. Well where all the actual data lines and that is in The Simpsons on school days. Sir, this is the best folder where all the images are stored in. Sewing to copy this file path and paste that in there. Cool so essential were gonna be doing now is recently going to grab the top 10 characters which have the most number of images for that class. The weather gonna do that is we are going to go through every folder inside The Simpsons\_dataset get the number of images that are stored in that dataset store. All of that information it sounded dictionary. So that dictionary in descending order and then grab the first 10. Elements first 10 elements in the dictionary help that make sense. So what we gonna do is going to say create an empty dictionary. Going to say 4 character in OS list der Kar Puff. We're gonna say Karan school dekhte of car is equal to length of OS list of OS path Dot join join the car on scroll path with car. So essentially all over doing is we're going through every folder or grabbing the name of the folder and we're getting the number of. Images in that folder and restoring all that information inside a dictionary call Karana scolded. Once that's done, we can actually sort this dictionary in descending order. No, said sending order. And the way we do that is we check our unschooled dekhte is equal to CR dot sort\_dekhte of car and Skoda Dekhte and we said descending equals to true. If only weekend, print the dictionary that we get. So this is the dictionary that we have to educate the home Simpson has the most number of images 8 + 2 2300 and we go all the way down to Lionel who has only 3 images in the dates. So what we gonna do is now that we have this dictionary, what we're gonna do is we are going to grab the names of the first 10 elements in this dictionary. And store that in a list of characters list so we use characters. So gonna say characters is equal, 2 is equal to empty list and we're going to say 4 in car on score dekhte. Gunasekaran trs dot append and we're gonna pen the name so we say of zero. If count is greater than or equal to 10 we can break. Then we need to specify account of zero and increment that counts. Okay, one second. Let Sprint what characters looks like. So we have. So shall I just grab the names of the characters? So we're not done. We can actually go ahead and create the training data. Educator training data is as simple as in train is equal to share data preprocess. From there we pass in the car and scope of it characters. The number of channels. The image science. Image size. Is shuffle. Equals. So essential what this will do is it will go through every folder inside car on the scope of witches, Simpsons\_dataset and we look at every element inside characters. So essentially it is going to look for a Homer Simpson inside The Simpsons\_dataset if it finds homeless Simpson or as pharmacist. If you find HomeSense and its going to go through inside that folder and grab all the images inside that folder and essentially add them to our training set. Now as you make a call in the previous video. Training set was essentially a list. Each element in a twist was another list of the imagery and the corresponding label. Now the label that we had was basically the index of that particular string into characters list. So thats the same type of mapping that we're going to use. So whom Simpsons gonna have? Of label of Zero, Net will have label of one, these they will have a label of 3 and so on. So once its done go ahead and run this. Now basically basically the progress is displayed to the terminal. If you don't want anything output it to the terminal. You can basically just set set Baba City 20. But I'm gonna leave things just as it is. Since there are lot of images in scientist dataset, this may take a while depending on how powerful your machine is. So thats only took about a minute or so to preprocessor data. So essentially that's trying to. So lets essentially try to see how many images there are. In this training set, we do that by saying the length of trip, and we have 13,811 images inside this training center. So lets actually try to visualise the images that are present in this dataset. So we're going to import Matplotlib. Dial pipe lot as PLT we gonna do a PS I don't figure. I'm gonna give it, I'm gonna give it up. Big size off 30 by 38 zero PLT. I'm show we can pass in the first the first element in this training sets zero and then zero. Yeah, we can give it a colour map of grey. And we can display this image. Now the reason why I'm not using open TV to display this image is because for some reason open TV does not display properly in Jupiter notebook. So thats why by using Matplotlib. So this is basically the image that we get. It's not very legible but. Auto machine this is a valid image. Okay, the next thing we want to do is we want to separate the training set into the features and labels. Right now the training set basically is a list of 13,811 list inside it. Inside each of that sub lists of 2 elements the actual array. And labels itself. So we're going to separate feature set for the erase and the labels into separate list. The way we do that is matching feature set. And labels is equal to share a step on score train. We are going to separate the training sets. And given image size. Of image size L apps and equals to. So basically what is going to do is going to separate the training set into the feature set an labels and also reshape this feature set into a 4 dimensional tensor so that I can be fed into the model with no restrictions whatsoever. So go ahead and run that and once it's done lets actually trying to normalise. There are beaches at. Chachi, we are going to normalise the data to be in the range of. Give me the range of zero one. And the reason for this is because if you normalise the data, network will be able to learn the features much faster than you know, not normalizing the data. So you are going to save pictures at. Is equal to Sierra don't normalise and when pass in pgcet. Now we don't have to normalise the labels, but we do need to one hot encoder that is converted from numerical integers to binary class vectors, and we do. That is not saying from 10 support dog carers. Dot U tools import to unschool categorical and we can say labels is equal to 2 categorical and we can possible labels and the number of categories which is basically the length of this characters list. Ohh so once that's done so once its done we can actually move ahead and try to create our training and validation data. Now don't worry too much if you doing in what these are, but basically the models going to train on the training data and test itself on the validation data am going to say exons Coltrane, exons Coldwell. And why unscored Kreem and WION school bell is equal to Sear dog train. Gel split. And why don't split the feature sets? And the labels using a particular validation ratio. Which bigger said as 0.2? So that's basically we doing with splitting the features said and the labels into into training sets and validation sets. With using a particular validation ratio. 20% of this data will go to the validation set, an 80% will go to the training sets. Okay, now just to save on some memory we can actually remove and delete some of the variables. I'm not gonna be using, so we do that by saying Dell train down. Feature sets. DO labels and we can collect this bye saying GT dont correct. Cool. Yeah. We need to create an image data generator and this is basically an image generator that will essentially synthesise new images from already existing images to help introduce some randomness to our network and make it perform better. So we're gonna say data Jenn is equal to Canaro. Don generators. Dog image data generator. Add this basically instantiates a very simple image generator from the Karas using the carrots library and one sedan. Let's create a training generator by sending this equal to date is n doped flu. And we can pass in ex train. In wine train. And give it a batch size. Equal to or batch sons. So lets actually create some variables here. That's set my bag size. 232 and maybe lets dream the network fool. Tni box. So once it's done that's wrong that so we that done we can actually proceed to building model. So what school does creating the model? And before making this video I actually tried and tested out a couple of models and found that one actually provide me with highest level of accuracy. So thats the same model, the same model architecture that gonna be using Jio were gonna say model is equal to Canaro. Dot models create Simpsons model. We're gonna possibly image science which is equal to the image science. We're gonna say said the number of channels equal to the number of channels. We can say we can set the output dimensions to 210, which is basically the length of a characters. Then we can. Now we can specify a loss which is equal to binary. Binary cross entropy. Cricket Satta dikhaye of east of E to the -6 power we can set a learning rate equal 2.001. Weekend sets. Mum. Momentum. Of 0.9 and we can set nature of true true. So this will essentially create the the model using the same architecture I built and will actually compile the model so that we can use it. So go ahead and run this. And we can go ahead and try to print the summary of this model. And so essentially what we have is a functional model since we are using Caris is functional API. And this is actually has a bunch of layers. Air about 17,000,000 parameters 2 trainer. So I don't think I want to do is create something called a callbacks list. Now this call back list will contain something called learning rate schedule that will essentially schedule the learning rate at specific intervals to that are network can essentially train better. So we gonna say call. Callbacks list is equal to learning rate. Schedule. And we're gonna pass in Canaro. Dot LR on school, LR on a school schedule and we are using learning reschedule. Let's go and import so from tensor flow. Dial kairouz, don't call backs, import learning rate schedule. And that \*\*\*\* back to it. So lets actually go ahead and train the model sewing to say training is equal to model dot fit bit pass on the train gym. Begin story steps fair. Hey, I walk. Is equal to the length of expense Coltrane divided by divided by batch. Science we use epochs. Is equal to epochs. How can I get the validation data? Navigation terror equals reachable of Exxon score well. In WION score well. I want to say validation. Steps. Easy steps is equal to you the length of WION score well divided by divided by. The batch. Pack size in family we can say callbacks is equal to callbacks callback son school list. Steps per epoch rad steps for East Park. In thats begin training and once it's done we end up with a baseline accuracy of close to 70%. So here comes the exciting party wear. Now going to use OpenCV to test how good are model is. So what we're gonna do is we're going to use OpenCV to read in an image at a particular file path and we're going to. Pass that 2 on network and see what the model spits out. So let's go ahead and go to. Is Simpson test set, so let's go ahead and try to search for. Or we doing here? Let's go to captures. Let's just printout just to see what characters we trained on. Quotes look for Baat Simpson.