

Deep Learning for Computer Vision
Professor Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Image Sampling

The next topic we are going to look at is another important fundamental topic in processing of images which is sampling and interpolation. Before we go there, we will quickly review the questions that we left in the last lecture.

(Refer Slide Time: 0:31)

Cost Improvement using Convolution Theorem?

Convolution Theorem



- Fourier transform of convolution of two functions is product of their Fourier transforms:

$$F[g * h] = F[g]F[h]$$


- Convolution** in spatial domain can be obtained through **multiplication** in frequency domain!

$$g * h = F^{-1}[F[g]F[h]]$$

- Image convolution needs $O(N^2 \cdot k^2)$ time, where $N \times N$ is image size, and $k \times k$ is kernel size
- By performing convolution in Fourier domain, cost is: $O(N^2)$ for a single pass over the image + cost of FFT: $O(N^2 \log N^2)$ for the image and $O(k^2 \log k^2)$ for the kernel $\approx O(N^2 \log N^2 + k^2 \log k^2)$, in total (other terms additive)

Vineeth N B (IIT-H)§1.7 Sampling and Interpolation



So one of the questions we asked in the last lecture was, by implementing convolution by going through, to a Fourier domain that is using the convolution theorem, what cost improvement do you really get? Convolution theorem allows us to define convolution in the spatial domain as a multiplication in the frequency domain and then taking an inverse Fourier transform. So you take the individual Fourier transforms of your g and h which can be our image and your filter or kernel and then you multiply them and then you do the inverse Fourier transform.

Let us take a simple basic case here to explain the cost improvement, so your image convolution takes $O(N^2 k^2)$ time where $N \times N$ is the image size and $k \times k$ is the kernel size. Then by performing the convolution in the Fourier domain you are still going to have a cost of $O(N^2)$ for

doing a single pass over the image. In addition to that to do FFT you will need $O(N^2 \log N^2)$ for the image, recall that when we spoke about the Fast Fourier Transform, we said that if you have N samples, the cost is going to be $N \log N$.

In our case an image has N^2 pixels, so the cost of computing for your fast Fourier Transform is going to be $O(N^2 \log N^2)$. Similarly, $O(k^2 \log k^2)$ kernel which is of course $k \times k$. So together your computation is going to be $O(N^2 \log N^2) + O(k^2 \log k^2)$. Any other computations that you are going to have are going to be additive and this will be the largest component which will overshadow the others.

And clearly you can see that this is cheaper than the cost, the original image convolution operations, especially when k gets bigger. Which means if your filter size or your mask size gets bigger using fast Fourier Transform to implement convolution is going to give you some cost benefits. A lot of the deep learning libraries that you use today, already implement convolution through Fourier transforms, you may not know it, you may just be using a library, a deep learning libraries say, for instance and you just invoke convolution but the backend of this, many of them already use fast Fourier Transforms to implement their convolution which makes it fast.

(Refer Slide Time: 3:09)

Exercise: Match spatial domain image to Fourier magnitude image

The slide displays a matching exercise between spatial domain images and their corresponding Fourier magnitude images. The spatial domain images are labeled A through E, and the Fourier magnitude images are labeled 1 through 5. The images are as follows:

- A:** A small grayscale image of a vertical bar.
- B:** A grayscale image of tulips.
- C:** A grayscale image of a car.
- D:** A grayscale image of a bright spot.
- E:** A grayscale image of people on a beach.
- 1:** A Fourier magnitude image showing a central peak.
- 2:** A Fourier magnitude image showing a central peak with a horizontal line.
- 3:** A Fourier magnitude image showing two peaks.
- 4:** A Fourier magnitude image showing a central peak with a vertical line.
- 5:** A Fourier magnitude image showing a central peak with a cross.

The slide also includes the NPTEL logo and the text "Vineeth N B (IIT-H) §1.7 Sampling and Interpolation".

The other exercise we had was to match spatial domain images, hope you had chance to figure out these answers yourselves in case you had doubts on the answers, let us do the simple ones first.

(Refer Slide Time: 3:25)

Exercise: Match spatial domain image to Fourier magnitude image

The slide displays five Fourier magnitude images labeled 1 through 5 at the top. Below them are five spatial domain images labeled A through E. Red arrows connect the Fourier images to their corresponding spatial domain images: 1 to D, 2 to B, 3 to A, 4 to C, and 5 to E. The slide includes NPTEL logos on the right side and a small video inset of a man in the bottom right corner.

1 would go to D, that is straight forward. 3 would go to A, straight forward and we also had that clue in one of the slides in the earlier lecture. And 4 goes to C mainly you look at 4, you see a very strong vertical set of edges and you see that image C has a lot of horizontal edges in image. So the horizontal edges in the original image would lead to a vertical edge in the Fourier transform. Go back and look at the sample Fourier images that we saw last lecture, you will see that we saw an example of a vertical line in your spatial domain which lead to a horizontal line in your Fourier domain which would be the inverse the other way.

So if you had a horizontal line, if you had a horizontal line in your spatial domain you can get a vertical line in your frequency domain. Once again I think the scope of this course would not allow us to go too deep in to the Fourier domains and related concepts plus, but do read the links that we shared in the last lecture.

(Refer Slide Time: 4:37)

Exercise: Match spatial domain image to Fourier magnitude image

The slide displays five spatial domain images (1-5) and five Fourier magnitude images (A-E). Red arrows indicate the correct matches:

- 1 (Spatial) matches D (Fourier)
- 2 (Spatial) matches E (Fourier)
- 3 (Spatial) matches C (Fourier)
- 4 (Spatial) matches B (Fourier)
- 5 (Spatial) matches A (Fourier)

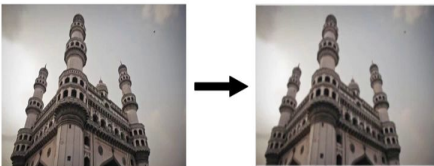
The slide also features NPTEL logos and a video feed of the presenter.

Regarding the other two images, which can be a bit tricky image 2 goes to E and the reason for that is if you look at image 2, there is pretty strong vertical line there and that comes from this horizontal lines at the horizon of the water or even these horizontal lines that you have in the repulse through there are lots of horizontal lines in your repulse, that is translating to this vertical line in the Fourier match.

And the fifth one is a straightforward extension, in the fifth one as you can see that the there are lots of stalks in the flowers that are giving the horizontal aspect of the frequencies and there are also lot of horizontal edges across the different flowers that give you some of these vertical frequencies.

(Refer Slide Time: 5:26)

What sense does a low-resolution image make to us?




Original Subsampled & zoomed

Clues from human perception:

- Early processing in human's filters for various orientations and scales of frequency.
- Perceptual cues in mid-high frequencies dominate perception.
- When we see an image from far away, we are effectively **sub-sampling** it.

Credit: Ron Hansen (Unsplash)

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation



Let us review one of the questions we asked the last time again as to what sense does a low-resolution image make to us? If we had to go back and try to understand this from human perception perspective, it is understood today that early processing in human's filters in the human visual system are filters that look for different orientations and scales of frequency. We will review this bit more carefully later but sometimes people also relate them to what are known as Gabor filter.

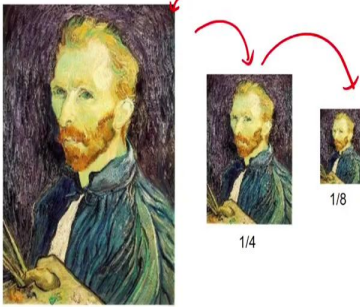
And secondly the perceptual cues in the human visual system are largely based on mid to high frequencies. So remember once again, high frequency means an edge because there is a sudden change in intensity value as you move from one pixel to the next pixel along a row or a column. If it is along a row, you would call that a vertical frequency, if it is along a column, you would call that a horizontal edge.

So perceptual cues depend on mid to high frequencies in general, so but when you see a low-resolution image, it is equivalent to sub-sampling image or if you are seeing an image from very far away you are actually sub-sampling the image, which means you may be losing some of the high frequency information but you do get a lot of the low frequency and mid frequency information which is good enough to get a sense of what the object is but may not give you finer details.

(Refer Slide Time: 6:56)




Sub-sampling

Throw away every other row and column to create a $1/2$ size image.



Credit: S Seitz, R Urtasun

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation




So, how do you sub-sample an image as an operation? The simplest way to sub-sample is given an image such as this, you simply throw away every other row and column. Consider only every alternate pixel both row wise and column wise. That straight away gives you a half size image, you can then go to repeating this to go to lower and lower resolutions.

(Refer Slide Time: 7:27)




Sub-sampling

Why does this look so cruffy?



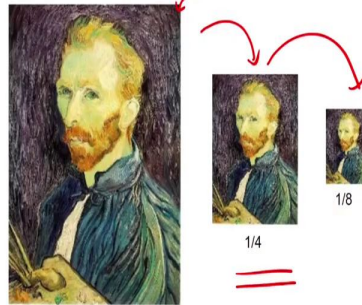
Credit: S Seitz, R Urtasun

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation



Sub-sampling

Throw away every other row and column to create a $1/2$ size image.



Credit: S Seitz, R Urtasun

Vineeth N B. (IIT-H)

§1.7 Sampling and Interpolation



But if you actually zoomed into these low-resolution images, or the sub-sampled images, this is how they look, we are just zooming in to these images here and if you just zoomed in and looked at those images, the half sampled image gets you something like this, the quarter sampled image at a 2x zoom gets you something like this, the 1 by 8th image at a 4x zoom looks like something, something like this.

You may ask, say yes, that is expected, I mean if you sub-sample, why do you, I mean why do you worry about how it looks at close, that is a valid question. But there is a reason for bringing this up here is the last image here looks pretty cruffy, it has too many artifacts, can we at least make it look smooth? We understand that by subsampling you have lost some information but can it appear smooth when you zoom in, is what we are looking for, we understand that you may have lost sharpness but we still want to look, make it look smooth. What do you think? How would you achieve this?

(Refer Slide Time: 8:43)

Sub-sampling

What's happening?



Credit: S Seitz, R Urtasun

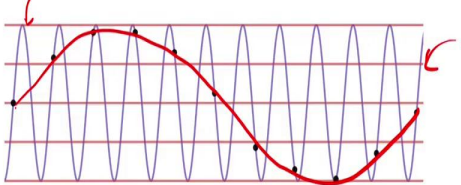
Vineeth N B (IIT-H) §1.7 Sampling and Interpolation





Let us wait for the answer for a moment but let us also see another example of an effect such as this. So do you what is happening in this kind of an image? So this is the original image, we have sub-sampled in some way and you get an effect such as this.

(Refer Slide Time: 9:05)

Aliasing






Aliased



- Occurs when your sampling rate is not high enough to capture the amount of detail in your image.
- To do sampling right, need to understand the structure of your signal/image.
- The minimum sampling rate is called the Nyquist rate.

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation



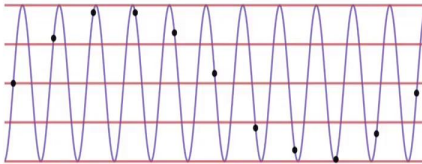
This effect is also known as aliasing. So aliasing generally happens when your sampling rate is not high enough to capture the amount of detail in the image, we are going to see a couple of examples and then re-visit the painting that we saw and how do you ensure that it is not cruffy.

So whenever your sampling rate is not high enough, you will not get the details, example you see a sinusoidal wave form on top here. If you sub-sample, only at these points remember that you are not looking at the downward trend of the sin wave.

So if you only sub-sample at these black points, you may think that this is the actual sin wave. Remember here that the frequency of this red wave that we have just drew is very different from the original wave. And why is it so different, only because we did not sample the original signal, when I say original signal I mean the one here in purple. Because we did not sample it enough, that is the reason why we end up getting this kind of a shape which is very different from the original sinusoid wave.


(Refer Slide Time: 10:25)



Aliasing



- Occurs when your sampling rate is not high enough to capture the amount of detail in your image.
- To do sampling right, need to understand the structure of your signal/image.
- The minimum sampling rate is called the **Nyquist rate**.


Aliased





Vineeth N B (IIT-H)

§1.7 Sampling and Interpolation

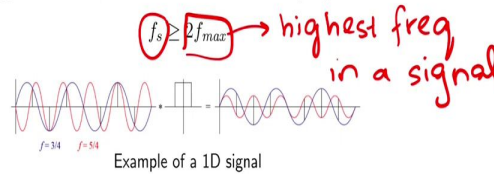


So what you need to do to get the sampling right, ideally you need to know the structure of your original signal and this is where an important concept called the Nyquist rate comes into the picture. We also mentioned this briefly in the previous lecture with the minimum sampling rate that is required to reconstruct your original signal or to capture the original signal in its actual form is called the Nyquist rate. So now what does the Nyquist rate mean?

(Refer Slide Time: 10:58)

Aliasing: Problems

Shannon's Sampling Theorem shows that the minimum sampling is:



Examples

- **Image**
 - Striped shirt's pattern look weird on screen.
- **Video**
 - Wagon Wheel effect: Wheels spins in the opposite direction at high speed.
- **Graphics**
 - Checkerboards disintegrate in ray tracing.

Vineeth N B (IIT-H)

§1.7 Sampling and Interpolation

For accessing this content for free (no charge), visit : nptel.ac.in



NPTEL



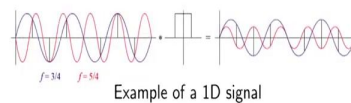
NPTEL



Aliasing: Problems

Shannon's Sampling Theorem shows that the minimum sampling is:

$$f_s \geq 2f_{max}$$



Examples

- **Image**
 - Striped shirt's pattern look weird on screen.
- **Video**
 - Wagon Wheel effect: Wheels spins in the opposite direction at high speed.
- **Graphics**
 - Checkerboards disintegrate in ray tracing.

Vineeth N B (IIT-H)

§1.7 Sampling and Interpolation

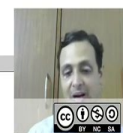
For accessing this content for free (no charge), visit : nptel.ac.in



NPTEL




NPTEL



Nyquist rate goes back to fundamentals from informational theory and signal processing where Shannon whose sampling theorem is very popular, proposed that the minimum sampling rate or the Nyquist frequency is given by f_s which has to be at least $2 * f_{max}$, f_{max} here is the highest frequency in a signal, it is the highest frequency in a signal. And we are saying now and for an image we would say it is an image and we are saying that your sampling rate or Nyquist frequency should be at least twice the highest frequency that you have in your image or signal.

Why is this so, we are going to see in a moment but we can explain the impact of not doing this in an image setting, in a video setting or even in a graphic setting. Let us see couple of examples. (Refer Slide Time: 12:07)






Aliasing: Image



Striped shirt's pattern look weird on screen.

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation

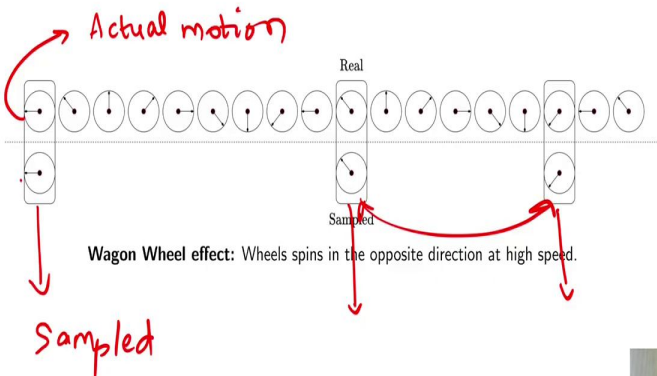
For accessing this content for free (no charge), visit : nptel.ac.in



So in an image setting if you do not sample your original signal at the appropriate frequency which is twice f_{\max} , your shirt is going to look very weird and this is as I said what is known as aliasing.

(Refer Slide Time: 12:21)

Aliasing : Video









Wagon Wheel effect: Wheels spins in the opposite direction at high speed.

Sampled

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation

For accessing this content for free (no charge), visit : nptel.ac.in

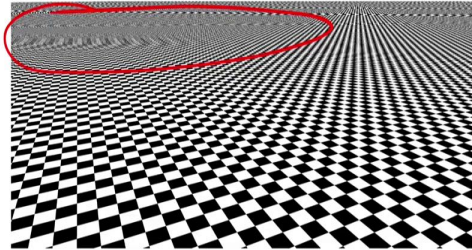


In a video, this is something that you may have observed a lot, sometimes when you go out on a road you see a bicycle, especially it happens with bicycles or car wheels, you may although car is moving forward, when you look at the wheels alone, it may appear that the wheels are circling backward. So this could be first sight bicycle or a car, try observing this if you have not already done this.

If you have not asked yourself why this happens, here is the reason. The reason is if you now take the motion of a wheel to be the top row here, so the top row here is the actual motion. As you can see the top row, you can see that the arrow here is moving in a clockwise direction, so you can assume that this is something like a cycle wheel moving in a clockwise direction. But we have chosen to sample it only at three locations across all of these. So the top row is the actual movement of the wheel but these boxes here are the ones that are actually sampled, they are the ones that are sampled.

So if you looked at only the sampled positions, you see that the arrow is first here, then it is here, then it is here, so on so forth. So probably in fact the second and third one could give you the reason why this effect happens, so you see that if you only took the second and the third one, you would actually think that your wheel is moving counter clockwise, because those are the only two samples you have and the arrows moving in the counter clockwise direction, but originally your wheel was moving in the clockwise direction. This simply means that you are not sampling the wheel at the frequency at which it is rotating or at twice the maximum frequency at which it is rotating.

(Refer Slide Time: 14:16)



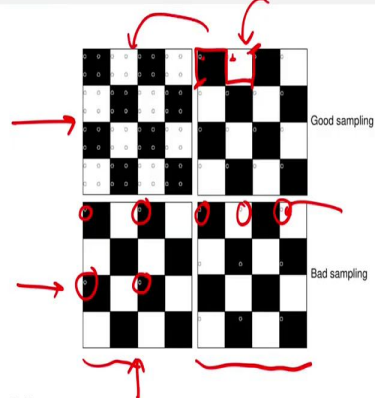
Checkerboards disintegrate in ray tracing.



Here is another example from graphics and the reason is again the same. You can see in this particular case that the aliasing is prominent at these regions where the frequency is very, very high. Remember once again, when we say frequency in an image, we mean how quickly the pixels change between black and white. When we say black and white, it can be dark grey or light grey but going from black and white and that becomes very high, very closer to the horizon of this image. And that is where the frequency is very high, which means your sampling rate has to be twice that frequency to be able to be true to the original image and this is an effect that we often see with graphics.

(Refer Slide Time: 14:57)

Aliasing: Nyquist Limit 2D example



Credit: S Seitz, R Urtasun

Vineeth N B. (IIT-H)

§1.7 Sampling and Interpolation



A more tangible way to understand why you need to sample at twice the maximum frequency, we are not going to derive Shannon sampling theorem here because that is not the focus of this particular course but let us try to intuitively understand why you need to do sample at twice the maximum frequency. Let us take this example of a chess board, so you have a chess board here and we argue now that the top row are examples if good sampling practices and the bottom row are bad sampling practices.

So let us see here that if you see this one here, that is the bottom left, you can see that the samples are these circles here, you can see those circles. And when you see those samples, you can clearly see that we are sampling all of them at only blacks. So if you, if you had only those four samples, you going to assume this entire board is a black board, that there are no white squares in between because you never sampled in the white squares.

So clearly you are not sampling at that appropriate frequency, the frequency here is how quickly does white and black change in how many pixels does it change. Clearly you are not sampling at twice that maximum frequency and it is going to give you poor information. Here is a slightly better sampling but still not complete, so once again here you can see these hallow circles, you can see that you improved your sampling a little bit but once again you see a black, you see a white and again your third sample on that first row is a white again.

So, which means your perception or any further processing is going to imagine that this is black followed by a series of white, that is not checker board or a chess board, but it is just an entire block of white, after the first small square of black it is an entire block of white, which again is poor sampling. So here is a sampling where we actually go at twice the frequency, you may ask why twice, so in this case remember here that the actual frequency is when a signal completely goes through between its variations.

So in this case a signal completes from here till here and we are sampling twice inside those two boxes. So which means that we are sampling twice the maximum frequency. And when you sample like that you have got a sample in every black square and every white square and you now know this is a checker board. Obviously if you sample more than that, that is even better for you. Hopefully that give you an intuitive understanding of why you need to sample any signal, any image at, at least twice the maximum frequency of content in the image.

(Refer Slide Time: 17:53)

Anti-aliasing

Example: Gaussian Pre-filtering

Aliased

Anti-Aliased

Credit: N Snavely, R Urtasun

$\frac{1}{2}$ $\frac{1}{4}$ (2x Zoom) $\frac{1}{8}$ (4x Zoom)

Vineeth N B (IIT-H)
§1.7 Sampling and Interpolation

So a good anti-aliasing technique is Gaussian pre-filtering, we will talk about what Gaussian pre-filtering means in a moment but this is the image that we saw earlier, we said that this image was cruffy and this is what we want to try to achieve, we do understand that if you sub-sample, you going to lose some information but we at least wanted to make it look smooth.

(Refer Slide Time: 18:21)

Subsampling with Gaussian Pre-filtering

$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Gaussian filter
↓
low-pass filter

Credit: N Snavely, R Urtasun
Vineeth N B (IIT-H) §1.7 Sampling and Interpolation

So how do you do Gaussian pre-filtering? You first going to take an image and then do a Gaussian blur on it, you know now what a Gaussian blur is, it is Gaussian smoothing. You can take a 3x3 Gaussian kernel, in case you do not recall what a Gaussian kernel is, remember it look something like $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$.

Similarly, you can also build 5x5 kernels and so on and so forth. So you first apply that kernel on your input image that gives you a blur version of the image. Now, you sub-sample, then before you sub-sample again blur then sub-sample, blur then sub-sample, blur then sub-sample, so on and so forth. Why do you think this works? There is a reason for this. Let me ask you this question, this as we said is a Gaussian filter.

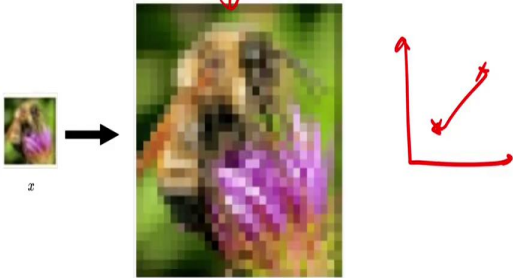
Do you recall what a Gaussian filter is, a low pass filter or a high pass filter? A Gaussian filter is a low pass filter, which means it removes high frequencies and only passes through the low frequencies, relatively lower frequencies which means it is going to remove, remember any smoothing filter removes a certain amount of high frequencies. So by applying the Gaussian blur, you have removed the high frequency which means your sampling rate now can actually be reduced. Remember again the Shannon sampling theorem states that you must sample at least twice the maximum frequency in the image.

If I reduce the maximum frequency in the image, I can sample at a lesser rate. And that is the trick that we employ to be able to sub-sample but still have a smooth effect in your output image.

So you do a Gaussian blur, now your overall frequencies have come to a smaller range, now you can afford to sub-sample without having an aliasing effect or cruftiness in the image and that is what we mean Gaussian pre-filter.


(Refer Slide Time: 20:40)

Upsampling



How to go from left to right? **Interpolation** Simple method: Repeat each row and column 10 times (Nearest Neighbour Interpolation).

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation



So now sub-sampling is one side of the story and the other side of the story is how do you upsample. Sometimes you need to go from a low-resolution image to a high resolution image and the simplest way to do this would be a very standard interpolation operation. The interpolation between any two values, so you have, we are talking about the real axes, the interpolation between any two values here is simply values that lie in the line connecting, at least linear interpolation is on the line joining those two points.

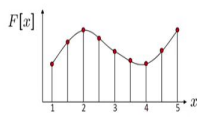
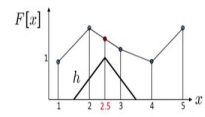
So the simplest form of interpolation that we can do, if we want to upsample an image is what is known as nearest neighbour interpolation, where we simply repeat each row and column 10 times. So every pixel, you repeat it 10 times along the column and 10 times along the row, so it almost becomes like one entire 10 by 10 block has exactly the same value. Then you take the next pixel repeat it for 10 columns, repeat it for 10 rows which means the next 10 by 10 block has the same value so and so forth.

Good news, your resolution goes up; bad news, you are going to have such artifacts across the image because this is exactly what you did, because you took several blocks, several pixels and

simply expanded them into blocks. How can we do better than this? Clearly we do not want to upsample images like this and have a blocky effect in the upsampled image.

(Refer Slide Time: 22:13)

Interpolation

Recall how a digital image is formed,

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$



- It is a discrete point-sampling of a continuous function.
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale.


What if we don't know f ?

- Guess an approximation: Can be done in a principled way via filtering.
- Convert F to a continuous function:

$$f_F(x) = \begin{cases} F(\frac{x}{d}) & \text{if } \frac{x}{d} \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

- Reconstruct: $\hat{f} = h * f_F$

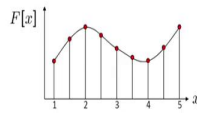



Vineeth N B (IIT-H)
§1.7 Sampling and Interpolation


So the way we can go about doing interpolation or upsampling while retaining some smoothness in the output is, let us try to recall how digital image is formed again, recall an image being a discrete sampled version of the original continuous function or the original continuous signal. So you have an original signal f , you quantize it at certain locations which gives you your image or your signal $F[x, y]$. In our case it is going to be an image.

(Refer Slide Time: 22:51)

Interpolation



Recall how a digital image is formed,

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function.
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale.

What if we don't know f ?

- Guess an approximation: Can be done in a principled way via filtering.
- Convert F to a continuous function:

$$f_F(x) = \begin{cases} F(\frac{x}{h}) & \text{if } \frac{x}{h} \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

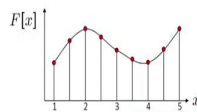
- Reconstruct: $\hat{f} = h * f_F$



So as we already said this is nothing but a discrete point sampling of a continuous function. So when we do interpolation, we are simply asking the question can we somehow reconstruct that original continuous function from the discrete samples. That is the question that we are trying to ask because if we do that, we could interpolate and go to any resolution that we want to go for.

(Refer Slide Time: 23:16)

Interpolation



Recall how a digital image is formed,

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function.
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale.

What if we don't know f ?

- Guess an approximation: Can be done in a principled way via filtering.
- Convert F to a continuous function:

$$f_F(x) = \begin{cases} F(\frac{x}{h}) & \text{if } \frac{x}{h} \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

- Reconstruct: $\hat{f} = h * f_F$



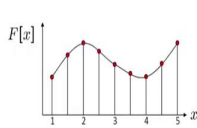
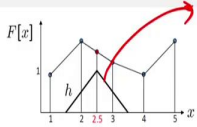
Unfortunately, in practice we do not know the small f which is your continuous function. We do not know that continuous function, we do not know how, what was the real world seeing from

which your image was captured in a sampled manner. However, we can try to approximate this in a principled way and the way we are going to do that is through a convolution approach.

So the first step that we want to do is convert your continuous function, sorry, convert your F , which is your discrete sample version to a continuous function because we do not know the original continuous function we are going to define a pseudo continuous function which simply states that wherever you know the value, you simply put the images value there, everywhere else you just say it is 0. This would be an automatic version of a continuous, this is at least a continuously defined function, let me put it that way, at least for the moment.

(Refer Slide Time: 24:18)

Interpolation

Recall how a digital image is formed,

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$



- It is a discrete point-sampling of a continuous function.
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale.


What if we don't know f ?

- Guess an approximation: Can be done in a principled way via filtering.
- Convert F to a continuous function:

$$f_F(x) = \begin{cases} F(\frac{x}{d}) & \text{if } \frac{x}{d} \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

- Reconstruct $\hat{f} = h * f_F$

Vineeth N B (IIT-H)
§1.7 Sampling and Interpolation


So now we try to reconstruct your \hat{f} which is the resolution at which you want to take f to, by convolving this pseudo continuous image with some filter. So we want to ask if, if we have some, this is not exactly right in terms of the function, so let me erase that. But if we have this function, continuous function, f_F , can we get some kernel h in such a way that we can construct your original, not original but at least an upsampled signal? What do you think 'h' should be?

This image should give you some understanding, maybe a signal such as a tent, in a shape of a tent could help you with the 1D signal perhaps, but that still does not answer how convolution can take us from one resolution to a higher resolution because the way we saw it, convolution can maintain the same resolution if you include padding and if you do not pad, you are going to

be, you probably will lose a few pixels on the boundaries. But now we want to go to a higher resolution then we use convolution.

(Refer Slide Time: 25:36)

Interpolation as Convolution

- To **interpolate** (or upsample) an image to a higher resolution, we need an **interpolation kernel** with which to **convolve** the image:

$$g(i, j) = \sum_{k, l} f(k, l) h(i - rk, j - rl)$$

Above formula similar to discrete convolution^a, except that we replace k and l in $h(\cdot)$ with rk and rl , where r is the upsampling rate.

- Linear interpolator (corresponding to *tent kernel*) produces interpolating piecewise linear curves
- More complex kernels e.g., B-splines.

^a $g = f * h \Rightarrow g(i, j) = \sum_{k, l} f(k, l) h(i - k, j - l)$

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation

Interpolation

Recall how a digital image is formed,

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function.
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale.

What if we don't know f ?

- Guess an approximation: Can be done in a principled way via filtering.
- Convert F to a continuous function:

$$f_F(x) = \begin{cases} F(\frac{x}{d}) & \text{if } \frac{x}{d} \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

- Reconstruct: $\hat{f} = h * f_F$

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation

Now, to be able to perform interpolation and then upsample an image to a higher resolution using convolution, we are going to do it using an interpolation kernel which is defined as what you see here. Remember again that your original convolution is defined something like this,

where you have $g(i,j) = \sum_{k,l} f(k,l)h(i-k, j-l)$ and this is the new definition that we are going to use for interpolation.

And if you observe carefully, the main difference between these two expressions, is the quantity r here, that is multiplying k and l , and r is what we call as the upsampling rate. So if you want to double your image, your r is 2, if you want to quadruple your size of your image, remember we are doing up sampling, so if you want to quadruple the size of your image, r is going to be 4, so on and so forth. Let us try to understand how this actually works.

Let us just for the sake of simplicity, let us look at one-dimension variant of this. Let us say $g(i) = \sum_k f(k)h(i-rk)$. That is the one dimensional variant of the same equation, remember once again that an image is a 2D signal for these purposes. So a lot of these expressions that we write for images, if you remove one-dimension, you can apply that as convolution for one dimensional signals.

So this is what we actually have in one dimension, so it is just easier to explain in one dimension. So let us consider, let us try to understand what is actually happening. So let us consider an x axis, so let us assume that there are some signals $f(k-1)$, that is it, there is another signal $f(k)$, there is another signal here $f(k+1)$, so on and so forth. This is very similar to what you saw in the earlier slide.

So I just drew lines such as these to represent the various values of f that I have with me, that is my input image which I want to increase the resolution of. So now, because I want to increase the resolution, the question I am asking here is if I take some i here in between those values, what would be $g(i)$ but g is the output, f is the input, that is what I am trying to ask. So in the g axis the tricky part here is that the coordinates, the indices in g are different from the indices in f , because f has only a certain number of indices whereas g has double those number of indices.

So this would be $rk-1$ in g indices, this will be rk and this would be $rk+1$ in the indices of g . Now if we try to understand what g of i is trying to do here, it is $\sum_k f(k)h(i-rk)$, as you take many k

values, so of course k can range from $k-1$ to $k+1$. If you take a 3×3 window or a higher depending on k simply defines the length of the window that you have.

So in this case, so what we need now is $h(i-rk)$, so if you again took an example such as this, remember here that the h was a one of the interpolation kernels, in this case it is called the tent kernel because it looks like a tent. And what this kernel does is as you go further out from the central pixel on which you place the kernel, the effect of those pixels fall off linearly, not exponentially but linearly. So that is your, that is what is your tent kernel.

So if you add a hat such as this, we are simply saying that this is f , so if you place the h kernel here, so if your h kernel is placed something like this, so then you are saying $g(i)$ is going to be $f(k)h(i-rk)$, $i-rk$ is this value here, $i-rk$ is going to be this value here with respect to h , which is the green cuff, so you are going to multiply f_k by h of i minus rk f_k plus 1 by h of i minus rk plus 1 which is going to be this quantity here, so on and so forth. So by doing this operation, you are able to interpolate and you are able to, you will obviously sum up those values, they are still in the summation here, you will sum of those values and that will give what the value of g of i is. That is how this interpolation kernels work.

(Refer Slide Time: 30:49)

Interpolation as Convolution

- To **interpolate** (or upsample) an image to a higher resolution, we need an **interpolation kernel** with which to **convolve** the image:

$$g(i, j) = \sum_{k, l} f(k, l) h(i - rk, j - rl)$$

Above formula similar to discrete convolution^a, except that we replace k and l in $h(\cdot)$ with rk and rl , where r is the upsampling rate.

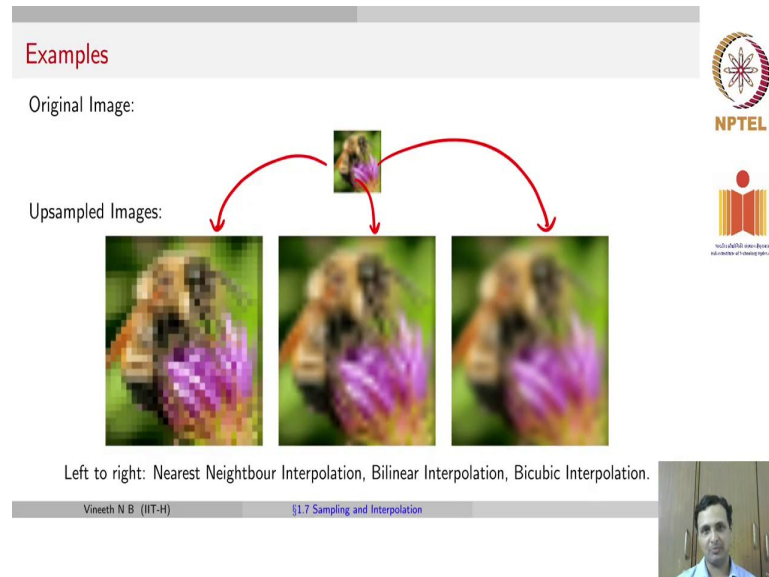
- Linear interpolator (corresponding to tent kernel) produces interpolating piecewise linear curves.
- More complex kernels e.g., B-splines.

^a $g = f * h \Rightarrow g(i, j) = \sum_{k, l} f(k, l) h(i - k, j - l)$



So now linear interpolator is what we saw corresponding to a tent kernel, you can also have more complex kernels such as B-splines, those are one of the most complex kernels.

(Refer Slide Time: 30:59)



But here are some qualitative results of applying this interpolation, so the first one that we already saw is an effect of doing the nearest neighbour interpolation, where you have the blockyness effects. Then if you do bilinear interpolation which is the linear interpolation that we spoke about but along the two axis and you get something like this, if you going to bicubic interpolation you get something like this, so on and so forth.

So the main difference in these interpolations is the h kernel that you use, in one case it is nearest neighbour, in the second case it is bilinear, third case is bicubic. We are not going to define every kernel but you can look at the references at the end of this lecture to be able to understand various kernels that have been defined in practice.

(Refer Slide Time: 31:48)

Interpolation and Decimation




Interpolation
To **interpolate** (or upsample) an image to a higher resolution, we need an **interpolation kernel** with which to convolve the image (r is upsampling rate):

$$g(i, j) = \sum_{k, l} f(k, l) h(i - rk, j - rl)$$

Decimation (Sub-sampling)
To **decimate** (or sub-sample) an image to a lower resolution, we need an **decimation kernel** with which to convolve the image (r is downsampling rate):

$$g(i, j) = \sum_{k, l} f(k, l) h(i - \frac{k}{r}, j - \frac{l}{r})$$

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation



So just like how we defined interpolation, you can also define the complementary operation of it which is decimation. Decimation means sub-sampling, so which means while we saw sub-sampling so far as simply omitting every other pixel where if you do Gaussian pre-filtering, you also ensure that you get a smoother output. You can also achieve sub-sampling using a convolution like operation where it would be very similar, the only thing is instead of rk you would have k/r and l/r here because you now want to sub-sample and not interpolate.

(Refer Slide Time: 32:28)




Homework Readings

Homework

Readings

- Chapter 3 (§3.5.1-3.5.2), Szeliski, *Computer Vision: Algorithms and Applications*
- Chapter 7 (§7.4), Forsyth and Ponce, *Computer Vision: A Modern Approach*

Vineeth N B (IIT-H) §1.7 Sampling and Interpolation



So that is where we going to stop this lecture, to know more interpolation kernels and decimation kernels, you can look at these references that are on your homework, please read them to understand this better and we will continue with the next topic.