Deep Learning for Computer Vision
Professor. Vineeth Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture – 06
Image in Frequency Domain

Moving on from the last lecture we will now talk about representing images in in a very different way looking at it as a two dimensional signal composed of multiple frequencies.
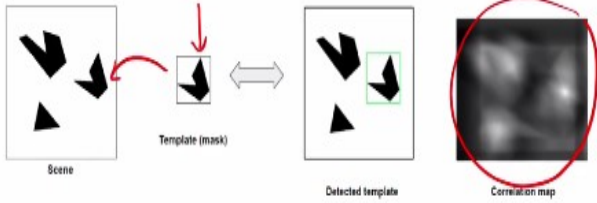
(Refer Slide Time: 00:31)

Before we go there, I think we left two questions from the last lecture which was, do we need cross-correlation at all or is convolution sufficient for all applications of image processing and the second question was, do filters always have to be linear. Hope you had a chance to dig up a bit about those questions.

(Refer Slide Time: 00:55)

Regarding correlation while convolution has mathematical properties that make it useful for various reasons, correlation can still be useful for a tasks such as template matching. Template matching is a task where you have given a template and you have to find that template in a larger image. For example, you could be looking for this template in an image such as this and your goal is to highlight the regions in the image where that particular template is present. In tasks such as this correlation becomes a more direct choice.

(Refer Slide Time: 01:39)

Another example is finding an object in a more real world image. Here again a point to note is that your template and the object in the scene may not exactly match. It may not exactly

match, but as long as they are similar in scale, in orientation, and general appearance you will still be able to template matching to be able to catch the object. As you can see if the car in the scene was from a different pose angle, a different size, then this may not have worked as this. There are ways to improve upon it. We will talk about that later, but as is correlation may not work. But template matching is a task where correlation can still be useful.

(Refer Slide Time: 02:31)

Regarding linear and nonlinear filters, let us try to give an example of a non-linear filter using a practical use case. We all know that there are different types of noise that is possible in images, so this was your original image. You could have salt and pepper noise, you could have what is known as impulse noise. Salt and pepper noise has black and white specks pepper and salt. Impulse noise has only white specks and you could also have a Gaussian noise where you add some Gaussian to the signal at each locations and things you have some noise due to that noise distribution. So, let us consider one particular noise case. Let us take the salt and pepper noise which all of you I am sure will agree is present in a lot of images that we see. So, salt and pepper noise looks like this. And if you now use a Gaussian filter which we saw on the earlier lecture and convolve this image with a Gaussian filter with a 3×3 Gaussian filter you would get this output, 5×5 this output, 7×7 this output. Do you see the problem? Irrespective of the size of the Gaussian filter, you see that we are simply smudging around the salt and pepper noise and not really removing the noise. What do we do?

(Refer Slide Time: 04:04)

Non-Linear Filters: Median Filter

- Replace each pixel with MEDIAN value of all pixels in neighbourhood

- Properties:
  - Non-linear
  - Does not spread noise
  - Can remove spike noise
  - Robust to outliers, but not good for Gaussian noise
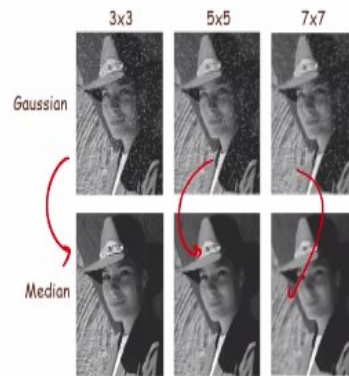
Credit: J Košecká, George Mason University

The answer is obviously in a nonlinear filter and in this case the nonlinear filter is going to be a median filter. So, we did see an average filter of mean filter last lecture. Median filter is simply a variant of the mean filter where you take that 3×3 window, sort them all out and simply take the median of the sorted list of values. If you simply use this as the filter, you see that in these cases things become much better in the 3×3, 5×5 and 7×7 cases and the salt and pepper noise is almost removed. Why does this happen?

(Refer Slide Time: 04:49)

Median Filter: Example
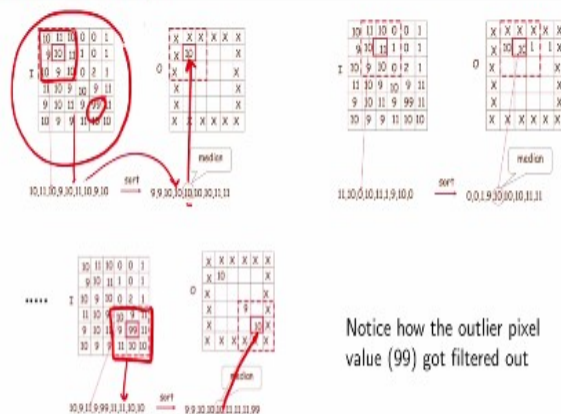
Notice how the outlier pixel value (99) got filtered out

Credit: J Košecká, George Mason University

If you take a more numerical example, so let us assume this was your image so to speak. So, if you going to take one particular window, take those values, sort them out and take the median which is at 10 in this case and put that median back here and you keep repeating this

process all through your image. So in this particular example, 99 here is a salt noise has a very high white value compared to the remaining set of values that you have in this image.

And when you come to that particular portion of the image, you list out your elements, you sort, take the median and when you do the median, it becomes an automatic way of discarding the outliers in that sequence of values. If you had done the mean filter, the 99 would have been used to compute the mean and hence the value in that location would have still been high.

But because you did median that values is no more relevant to you and you are going to get only 10 in that particular location and this helps you eliminate the salt and pepper noise. Clearly median you cannot get the mean in median through a linear combination of your input values so it is a nonlinear filter.

(Refer Slide Time: 06:14)

Another example, which you may have seen in several places is what is known as bilateral filtering. An example output from bilateral filtering looks like this you may have seen these images. So, where as you can see it almost looks like given a photo you have taken particular regions of the face and smoothened out pixels in those regions.

For example if you take the cheek region, if you just take the region there the pixel values in that region are smooth and the pixel values in some other region are smooth by themselves, the pixel values in this region are smooth by themselves and they do not bleed into each other. Now how do you achieve such an effect using a filter? You may have seen this in

image reading software such as Adobe Photoshop 2. So, bilateral filtering is a nonlinear edge preserving smoothing and let us see how this is achieved.
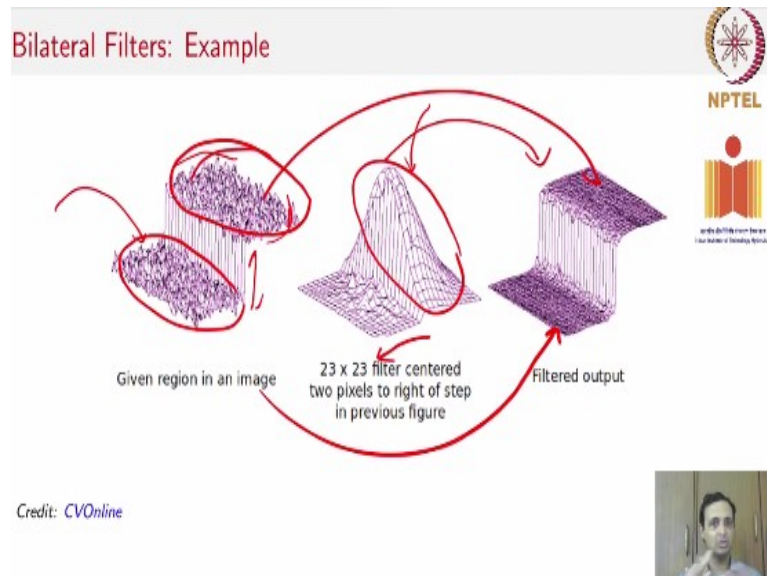
So, our main objective here is going to be, if a pixel differs too much from the central pixel value in its intensity, we do not want to consider that pixel. How do we implement it? It is going to be again a filter. If you remember that, this is what you have. I is your image, your filter is going to be a bunch of weights that you associate with each of those filter locations (i, j) or (k, l) in this particular place, but now and the denominator is simply normalization factor.

We already know that normalizing a filter is important especially for an averaging filter, but you do see that there are 4 coordinates here. What do you think those 4 coordinates mean? Let us see that using a specification of w(i, j, k, l). You have to only define w(i, j, k, l) as it is like a Gaussian filer. We say that if the coordinate location is far from (i, j) then weight it lesser that is one part of the term. The first term is very similar to Gaussian filter. If the neighborhood location is far from the central pixel, weight that a bit lesser, that part is same as a Gaussian filter. But you have an additional term here which also says, I do not want to decide this based on only the location of the pixel. I also want to decide the filter value by what value is there in the image at that location.

Remember so far in all our filters, the filters only where relevant to the position of a pixel in an image. The filter coefficients did not change for different locations in the image, but here we are talking about changing the filter coefficients for different locations and how do we change? We say that if an image pixel at (k, l) which is one of those locations in the neighborhood around (i, j) is going to be very far from the intensity value at (i, j) at the central pixel you want to weight that lesser.

So even if a pixel is an immediate neighbor of that central pixel that you are evaluating at, if it had an intensity that was very different from the central pixel, you are going to weight that lesser. That is what this function does and we call this bilateral because you are doing it along the coordinates as well as along with intensities.

(Refer Slide Time: 09:43)

Bilateral Filters: Example

Given region in an image

23 x 23 filter centered
two pixels to right of step
in previous figure

Filtered output

Credit: CVOnline

Why does this work? You see that on this slide you can see on the left which is the representation of the image. So, these are a bunch of intensity values in a vicinity of a particular number and then there is a large separation before a neighborhood which has a different set of intensity values with some variations and this is how a filter will look. Remember again that this filter will vary for different parts of the image based on the intensities in that local neighborhood.

So, here you see that the filter in this part of the neighborhood varies like a Gaussian. As we said if the intensities are similar the only thing that affects is the coordinate location and then the value of the filter drops off significantly because there is a huge change in intensity between a value here and a value down there. So, now what is the result if you convolve this filter with this input you get something like this where all the intensity values in this range get smoothened out here. And all the intensity values at the bottom range get smoothened out here and that is how you got the boy's image with a smooth cheek area and smooth regions in other parts of the face and neck.

Moving on to the main topic of this lecture, let us start by asking ourselves what do we really lose in the low resolution image. So, if you actually look at these two images perhaps our eye has the same response in both these images. We will probably seen a very similar scene in both cases. So what did we really lose when we reduced the resolution? You all know that we probably lost some sharpness or some edges in the image, but what does this really mean is what we are going to talk more about today.

So, when we talk about images in the frequency domain we mean Fourier domain and when we say Fourier we go back to the scientist Fourier who lived in the eighteenth century and

early nineteenth century who proposed way back in 1807 that any univariate function can be written as a weighted sum of sines and cosines of different frequencies. Unfortunately for this time, at that time you could say of course what is new you probably heard about it in some place.

But at his time many scientist including Lagrange, Laplace, Poisson and many others did not believe the idea and this is statement that they gave. They felt that it really was not complete and the work was not even translated to English to until 1878 that was 70 years after he invented the idea and much, much after he passed away. But today we know that this is mostly true. And this is called the Fourier series and has some subtle restrictions, but is more or less true across signals that we know and we are going to talk about viewing images as 2D signals now.

(Refer Slide Time: 13:01)

But before we go that, let us talk about the Fourier series itself. So, for simplicity we are going to talk about the Fourier series as representing any time varying 1D signal as a sum of sinusoids. We will move to images after some time. So, the building blocks are sin waves and w is the frequency, φ is a phase and A is an amplitude of the signal. So the idea here is that with these building blocks and with signals of different frequencies and phases, you could approximate any function. So, let us see a more tangible example for this.
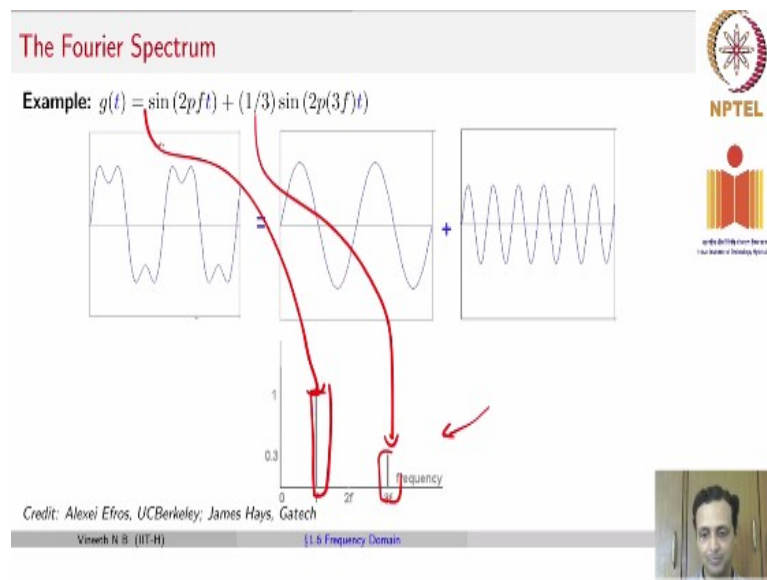
Let us take this particular example here. This function here g(t) what you see as I said the time varying one dimensional signal can be rewritten as sin(2pft)+(1/3)sin(2p(3f)t). So, there are two frequencies f and 3f. f is the frequency that gives you this signal and 3f is the frequency that gives you the other signal. Now let us say this is the input signal to us. We can write is as a sum of two frequencies because we know the actual definition.

Now in the Fourier domain this kind of a signal can be represented as, so you had along the x axis you have different frequencies and what we tell now is that the frequency f you have a certain amplitude and at the frequency 3f you have a certain amplitude. Here 1/3 at 3f and you have 1 at f. So, that is how you represent this signal in the frequency domain.

Now consider a slightly more complicated waveform such as this square waveform which is something like a periodic step function. Now let us try to see how this can be written using a Fourier series. So, we could we know already that by using frequencies f and 3f we can get a signal such as this. To this output signal we can add one more signal of an even higher frequency and then you would get this signal.

And to that signal you add a signal of even more higher frequency and you would get this signal and you keep repeating this process over and over again until you go closer and closer to your original square wave function. As you can see as you keep adding higher and higher frequencies, but in lesser and lesser amplitudes, you get closer and closer to a square wave which means in the Fourier domain the final square wave can be written in this form, where frequency f has the highest amplitude, the next frequency say 3f has a slightly lesser amplitude, 5f has an even lesser amplitude so on and so forth and as you keep adding more and more frequencies your waveform will get closer and closer to your original square wave.
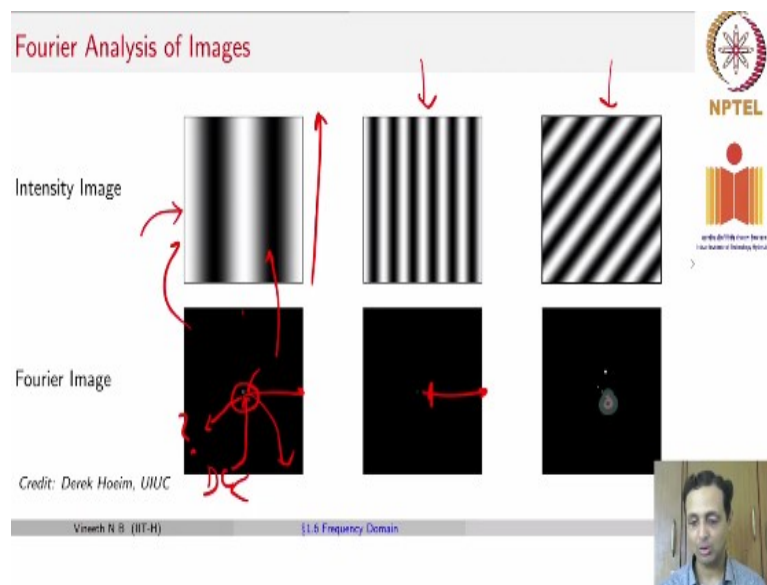
Now, let us come, that was in 1D example. So, if you come to images how would this look? So, this is slightly more nontrivial we will not get into every detail here, but hopefully I will try to explain the basics. So, if your original intensity image was something like this. So, what you see here is there is no variation along the y axis or there is no variation in any particular column. All the variations are along the rows so which means there are only horizontal frequencies here, no vertical frequencies.

Remember we are talking about 2D signals so there are going to be signals in both directions. But here in this particular example, there is no change along the columns, so there is only a horizontal frequency. So, typically in the Fourier image if you observe this carefully here there are 3 dots. The central dot is actually called the DC component or it simply the average intensity of your original spatial domain image.

The dot on the right here is the actual frequency. So you can now represent, let me try to draw it on this one it is easier for you to see. Imagine that to be your axis, imagine the central point to be your origin and that origin as I said is your DC component and on the x axis this particular point refers to one kind of a frequency and you can see a white dot there. White dot represents the amplitude of that frequency in your spatial domain image.

So, the Fourier domain image is simply telling us that there is this much amount of that frequency in this image. Now you could ask me what is the third dot here on the left hand side of your DC component, you may ask what is this? Just hold on to it and we will talk

about it on the next slide. Similarly, on the second column you see that it is very similar to the first column, but the frequency has got a bit higher.

And because the frequency is got a bit higher let me erase these lines. You can now see that on the x axis we now have a slightly higher frequencies present in this input dimension. Remember here that if you consider that to be in axis this had a lower value of the frequency, this has a higher value of the frequency and if you take the third column you can clearly see that there are frequencies both in the horizontal direction and vertical direction which boils down to points, lines somewhere here.

So, if you again thought of this as an axis in the Fourier domain. So, you have now some frequency along the x axis and some frequency along the y axis and that give you that dot. We will come to the third point in a couple of slides from now.

(Refer Slide Time: 19:27)

So, another way of summarizing what we just said is that in the Fourier domain an image can be represented as in terms of the presence of different frequencies of the signals starting from your zero frequency which is nothing, but your average intensity of image. Zero frequency means it is a constant value. So, what is the average intensity of the image is your constant value on top of which you are going to add your frequency changes.

And f is a particular frequency where you record what amplitude of that frequency is present in this given image and you go the, the x axis goes all the way until what is known as Nyquist frequency and we will come to that in the next lecture. For the moment you can just assume

that is something called Nyquist frequency. So, a signal is plotted as a single peak at point f if there is only one frequency in that image. And the height of that peak corresponds to the amplitude or the contrast of your sinusoidal signal. Now, if you noticed on these images we did say that there is a third pixel on these images and I said I would answer it.

(Refer Slide Time: 20:40)

So, it happens that when you plot your Fourier domain version, a spatial domain version of an image or a time varying signal for that matter, you typically mirror your entire representation along your origin. Why you mirror it? I think you may not have the scope to get into it in this course, but if you are interested, please look at this course to understand why is the Fourier domain representation mirrored?

So, the third dot that we saw on the slide couple of minutes back is just the mirrored version of the frequency that we saw along the x axis.

(Refer Slide Time: 21:21)

So, here are more examples. So, this is one kind of harmonic frequency and this is something like 3f, this is 5f. Note that along the x axis this dot here is actually moving forward and forward because the frequency is higher you are moving further along the x axis, this is 7f so on and so forth. Now you can combine these frequencies this is again 1f, this is (1+3)f which means now you get dots in both those locations. This is (1+3+5)f you get dots in all of those locations this is (1+3+5+7)f dots in all of these locations.

(Refer Slide Time: 21:59)

And we will keep doing this further and further and further if you had a single image as a white dot that is going to get transformed into a horizontal line in the Fourier transform. This

could involve slightly more complex understanding of Fourier transforms, but for those of you were comfortable remember that a discrete signal in an original domain becomes periodic and continuous in your Fourier domain. Again we will not get into those details keeping the focus of this course, but I will share references with you through which you can learn more about this.

(Refer Slide Time: 22:43)
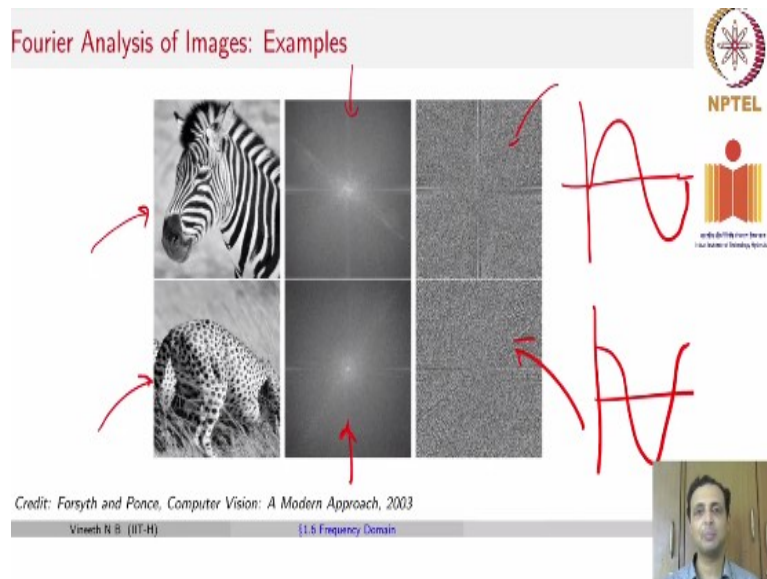
Here are some examples of how real world images look in the Fourier domain. So you can see here that this is a real world image. These are two animals. This is the log of the magnitude spectrum and what you see on the third column is the phase of the image. So, as we already mentioned, remember we said that in the Fourier domain you write out any signal as a sum of sinusoids with different frequencies and phases.

Phases are where the sinusoid starts. For example so this is a certain sinusoid with frequency. This is a sinusoid with the same frequency, but with a slightly different phase. So, the third column represents the phase component of the signals and we will try to talk about what it means over the next few slides.

(Refer Slide Time: 23:39)

So, Fourier transform consist of both magnitude and phase. So, in practice this is often written in terms of real and complex numbers. So the amplitude or the magnitude of the Fourier signal is written this way where φ is the actual representation and the magnitude is simply $\sqrt{Re(\varphi)^2 + Im(\varphi)^2}$, but the phase is given by $\tan^{-1}\dfrac{Im(\varphi)}{Re(\varphi)}$ tan inverse of imaginary by real. Once again we would not get into the depth of this, but I would share references where you can understand this in more detail.

(Refer Slide Time: 24:17)

To complete this discussion, so if you take a continuous signal such as this and if you take a Fourier transform of that continuous signal, its Fourier magnitude spectrum would look something like this which simply says that this is the DC component. You have certain frequencies with different amplitudes so on and so forth. If you now sample that signal and get a discrete version remember that when we talk about images this is what we have.

We do not know the real signal across the continuous signal across the image what we have is the signal sampled at different pixels. Remember this goes back to the representation of an image as a function. If you take a Fourier transform of a sample signal, your output would be a periodic continuous Fourier spectrum. These are concepts for which you may need to do a little bit more reading of Fourier analysis.

And I will again point you the references, but you can just keep in mind that with your input is a discrete aperiodic signal, your Fourier domain is going to be a periodic continuous signal and this is the kind of a signal that you would end up having and this will be very similar to copying and shifting the magnitude spectrum for your continuous notion of a signal.

So mathematically speaking, the continuous Fourier transform is given by $\int\limits_{-\infty}^{\infty} h(x)\,\mathrm{e}^{-jwx}\,dx$, where h(x) is your original signal and the discrete version of it is where you go from 0 to up to N-1 sample or total of N samples and that is what is the discrete version of the Fourier transform.

(Refer Slide Time: 26:15)

So, if you want to learn more about the Fourier transform a highly recommended resource is this first one "An Interactive Guide to the Fourier Transform", it gives a very intuitive explanation. There are also other references here if you would like to know more you can click on those links to know more.

(Refer Slide Time: 26:32)

Let us come back to why we are talking about the Fourier transform in this lecture and why did we have to deviate to talk about something that is for which you need to read a lot more. There is a particular reason for that and that reason is what is known as the convolution theorem. The convolution theorem states that the Fourier transform of the convolution of two functions is a product of their Fourier transforms.

Rather if you take g convolution h two signals and you take the Fourier transform of them. It happens that will be the Fourier transform of the individual signals and the product of the Fourier transforms of the individual signals. So, what does that mean? If we want to do convolution in spatial domain we can obtain it through multiplication in the frequency domain.

And this can lead to some computational savings not in all cases, but in certain cases especially large filters and large images which can be useful and we will talk about that in more detail and that is the reason we needed to talk about the frequency domain here beyond just getting a fundamental understanding.

(Refer Slide Time: 27:39)

Properties of Fourier Transform

Property	Signal	Transform
superposition	$f_1(x) + f_2(x)$	$F_1(\omega) + F_2(\omega)$
shift	$f(x - x_0)$	$F(\omega)e^{-j\omega x_0}$
reversal	$f(-x)$	$F^*(\omega)$
convolution	$f(x) * h(x)$	$F(\omega)H(\omega)$
correlation	$f(x) \otimes h(x)$	$F(\omega)H^*(\omega)$
multiplication	$f(x)h(x)$	$F(\omega) * H(\omega)$
differentiation	$f'(x)$	$j\omega F(\omega)$
domain scaling	$f(ax)$	$1/aF(\omega/a)$
real images	$f(x) = f^*(x) \Leftrightarrow$	$F(\omega) = F(-\omega)$
Parseval's Theorem	$\sum_x [f(x)]^2 =$	$\sum_\omega [F(\omega)]^2$

Credit: Szeliski, Computer Vision: Algorithms and Applications, 2010

Vineeth N B (IIT-H)   §1.6 Frequency Domain

The Fourier transform has a lot of interesting properties which nicely fit into convolution, superposition, shift, reversal, convolution, multiplication, differentiation, scaling and so on and so forth we are not going to walk into all this, but there are lot of useful properties of the Fourier transform.

(Refer Slide Time: 27:57)

Fast Fourier Transform

- Number of arithmetic operations to compute Fourier transform of N numbers (i.e., function defined at N points) is.... proportional to N^2
- Possible to reduce this to $N \log N$ using **Fast Fourier Transform (FFT)**
- FFT is a **recursive divide-and-conquer algorithm** for computing DFT

For more, see https://www.karlsims.com/fft.html

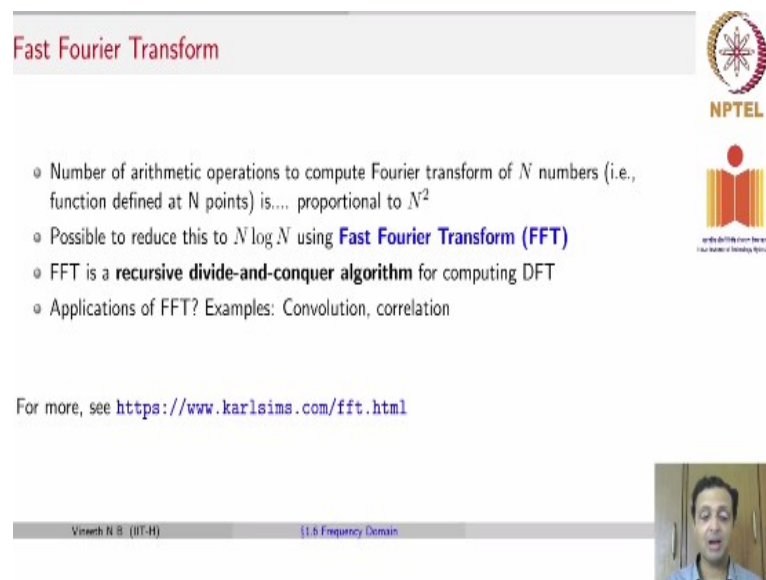Vineeth N B (IIT-H)   §1.6 Frequency Domain

So, if you now ask me the question how does it really matter looks like the Fourier transform also will take the same number of operations as convolution? Let us try to analyze this. So, the number of arithmetic operations to compute a Fourier transform of N numbers let us say a function was defined with N samples. What do you think would be the number of operations?

It would actually be proportional to N^2 that would be the Fourier transform just by the very definition of the Fourier transform itself.

But it happens that there is an algorithm called the Fast Fourier Transform which helps reduce this computation to NlogN. Fast Fourier Transform is a recursive divide and conquer algorithm simply does this by dividing the set of samples that you have in a different portions and then computing the overall Fourier transform. You can see this link to understand this more intuitively.

But this reduces the number of computations from N^2 to NlogN and this is what is going to help us in making convolution more feasible by doing the operations in the frequency domain and this is the trick that is often used to this day when convolution is implemented in libraries which you may just call when you learn deep learning.

(Refer Slide Time: 29:18)

Applications of FFT are in convolution and correlation.

So, let us see an example of this. So, if you have an intensity image and you take such a filter this is an example of an edge filter you would get an output such as this where only the edges are highlighted. This is your normal spatial convolution. So, what does this mean to go to the frequency domain?

To go to the frequency domain means this, you take the original image, take the Fourier domain version of this particular image. As you can see the Fourier domain version is still symmetric above that central (0,0) points. Remember we talked about it being mirrored

version, but there are lot of complex frequencies now because to look at an image like this and be able to eyeball it and find the frequency is hard so there are methods to do that.

So, you get the frequency domain representation, you take your edge filter, get the frequency domain representation. Multiply these two across point wise and you get this output then you do an inverse Fourier transform and you can get back your original image. Remember again if you go back here, this step is your inverse Fourier transform which is what we implement using an inverse FFT. Now coming back to the slide.

Now can we ask the question in this particular case what specific cost improvement did the use of convolution theorem really give? I am going to leave this as a question for you to think about. In this particular example can you try to think what actual cost improvement would the use of the Fourier transform or doing this operation in the frequency domain, frequency and Fourier are synonymous terms in this case would it give? Think about it and we will discuss the answer in the next lecture.

(Refer Slide Time: 31:28)

One more important concept here is the concept of what is known as low pass filters and high pass filters. So, when we talked about the filters or the masks or the kernels, it is common practise to refer to some such filters as low pass filters which means that they allow low frequencies to go through and block high frequencies. What would be an example? your Gaussian filter because it does smoothening.

Remember that when you smoothen an image you remove certain edge information which are high frequency components. High frequency components simply mean that there is a significant change between one pixel and the pixel immediately next to it that is what we mean by a high frequency in an image and high pass filters are filters that allow high frequencies to pass through and block low frequencies.

Can you think of an example here? It would be an edge filter because an edge is a high frequency component and it allows edges to go through, but blocks out your low low frequency components across your image. Now if you went back to the first slide that we used to start this entire discussion. When you go to a low resolution image from a high resolution image you actually lose your high frequency components.

But it happens that the human eye is very good at working with medium frequency components that you can still make out the meaning in an image, but that is what you lose when you go to the lower resolution image.

(Refer Slide Time: 33:11)

One last question before we wind up this discussion which is also an interesting topic is we said that there is a magnitude component and a phase component in every Fourier version of an image and this is how the magnitude is something that we said is easy to interpret the phase is a little harder to describe so we will not get into it now, but this is how it looks. Which do you think has more information?

We will try to understand this by perhaps swapping the magnitude in the phases and doing an inverse Fourier transform and seeing what you get. These are the kind of things that you would get. So, if you now take the magnitude in this cace I think yes it is the magnitude of your leopard or a cheetah and your phase from the zebra and similarly you take the magnitude of the zebra and take the phase from the leopard and you get these images.

What does this tell us? This tells us that the texture information comes from phase whereas the frequencies actually come from your magnitudes.

(Refer Slide Time: 34:28)

So, here is a popular example of doing this magnitude and phase switch using a method called hybrid images where you exactly do what I talk about. So you take an image, you simply first get a low resolution version of that by applying a Gaussian filter. You in fact can do it at multiple resolutions we will come to that a bit later and then you swap your magnitudes and phases of different images and you can get pretty looking images that can be interesting. These are called hybrid images and this was an interesting fact way back in the mid-2000s.

(Refer Slide Time: 35:11)

One more exercise before we stop. Here are a bunch of spatial domain images, spatial domain images, and corresponding frequency domain images on top. Your task is to match which one go where try this and we will talk about this in the next lecture.

(Refer Slide Time: 35:30)

We will stop here please do follow up on the readings and the exercies given to you.