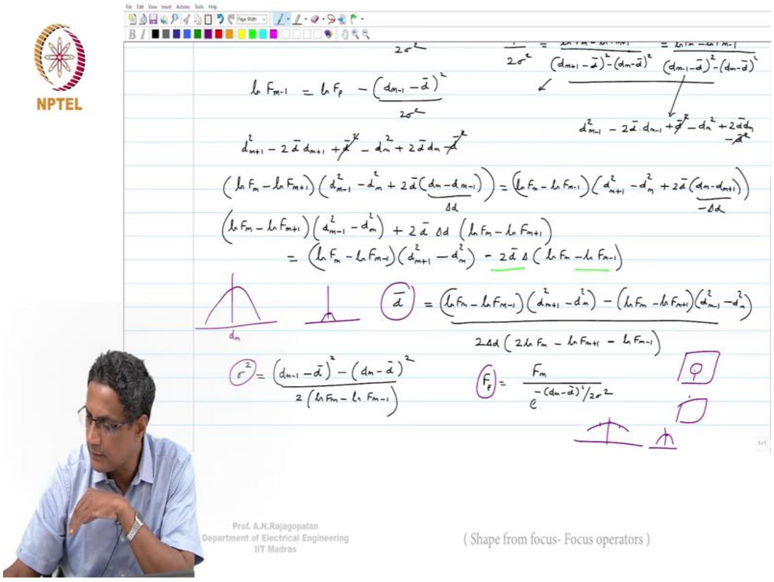
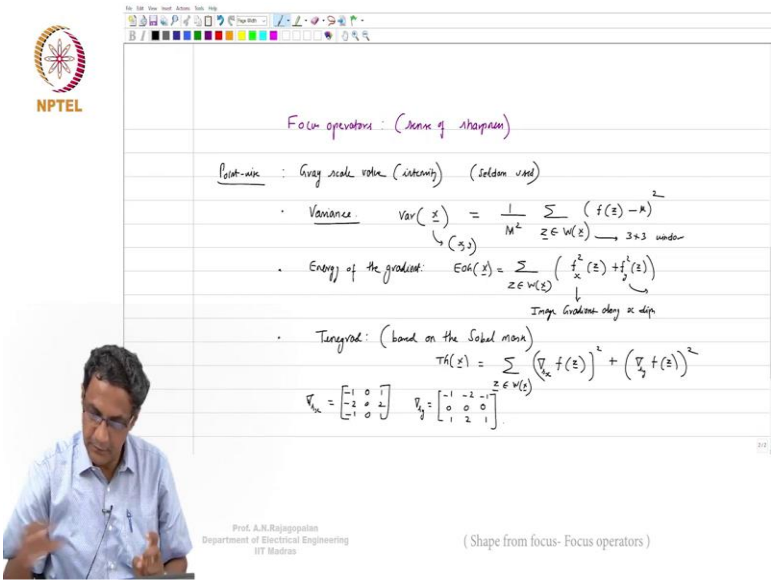
Image Signal Processing
Professor A. N. Rajagopalan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture 31
Shape from focus- Focus operators

(Refer Slide Time 0:27)



So there are no other equations, so we will move to the focus operators. So all the way we have been assuming that this plot this focus measure plot that I have been indicating. So I have been assuming that there is some operator which will act and give me a sense of sharpness.

(Refer Slide Time 0:42)



So this focus operator is something sp focus operator is in fact, so they will give you a sense of sharpness. In fact, these are other than this depth business that I have been talking about. In fact, these focus operators are constantly being used in all your cell phones by the way. So when it goes into that hunting mode, I mean you say autofocus, how does it know that something is in focus? How does it know what lens setting is ought to be brought into focus, how does it know that?

So when you say autofocus, what it is actually doing this, it is capturing a bunch of frames just as we are doing here, but then it is capturing it very fast. We do not even see all of that and we do not typically have access to those images, but you can do something to actually access them. But normally, we do not get to see any of that so it tries to move the lens.

And at that time, it does not care about the parallax and all that, because auto focusing is not about estimating depth and all. So all that it does is captures a bunch of frames, moves your lens, captures a bunch of frames, and then it knows that and then I know it runs a simple focus operator.

Again, each company has its own way of trying to sense the sharpness and then as it runs, this focus operator quickly and where does it run typically in the center. So it assumes that the central region is what we are interested in, so it runs right a quick focus measure operator tries to get a sense of where things are coming into focus or it knows that that is lens setting that it needs to use.

So this is called hunting and in fact many of these cameras they have different ways of speeding up the hunting process and all so there is a Panasonic 2017 model GH 5 or something, I read about it somewhere, Lumix something. So that in fact claims that it is 100 times faster than the other cameras, because it uses something, it uses only 2 frames to kind of lock on to the it calculates depth actually that is the one of the things that calculates, there is only one D there is nothing like a depth map.

This computes the depth for the central portion of the scene and based upon that, maybe if time allows I will talk about it. So I am saying that other than this sense of depth and all depth map and all that which is maybe more involved, but these things are being constantly used, I mean whenever you want to get a sense for sharpness and so on, but then nobody knows which operator is best for what scene that is, like nobody knows.

So there are all these people that have actually I said, that uses but then nobody knows which one of them is the best. So I am going to list down a bunch of these operators, we will kind of go through them one by one and you could use any of them, but there is no guarantee which one of them is going to serve you the best in a particular situation.

So as you would expect, so one of the first thing, the Knave things would be what is called a point wise operation. See, point wise is normally undependable, it is like saying that you are only looking at that point and not even around it. So when you just look at a point, it is typically a grayscale value and all these operators have to be simple.

You are going to have something that is very complex, that is running, which means that it is going to take time and so on. So if you look at these operators, they are very simple, they will not do anything fancy. So they do some simple things in order to get a sense for sharpness.

So grayscale value, is one which is simply what we mean by that is simply like look at the intensity, but then this is seldom used, why? Because of the fact that there is noise and so on, you would end up concluding something very, very wrongly. So, normally not used but this is point twice, this is the only point operator really, the restaurant all they do use a the local information, what is called a neighborhood.

They do not just go by what is happening at that pixel, they also look at what is happening in and around it. So, second is variance, but variance of course will also require you to compute the mean. And the mean of course, we will again use a window. So variance of, let us say, now I am going to write this as x subscript by which I mean that you are at some location x, y. I do not want to get every time write x, y.

So the x means that you are at a location x, y and what you will do is, you will do something like 1 by m square where let us say m number of locations are involved and summed over sum z belonging to a neighborhood of, so this is like window of x, w of x is like the neighborhood of x and then the value of the intensity, value of the image at that z.

So z again like some x, y sort of location, but which is the neighborhood of x. And typically, these are all very clean neighbors, 3 cross 3 you do not use some fancy, irregular neighborhood and all. Typically, regular neighborhood 3 cross 3, 5 cross 5 or something. f of z minus Mu where of course Mu again will be the mean that we can compute as simply 1 by m of that neighborhood and square. So variance seems to make sense, because if something is coming into sharpness then typically a variance will go up because your smoothness is less and therefore variance makes sense to use. But then this is also not so very often used, but yeah, but then this is more sensible.

Energy of the gradient or what is called EOG. So, this is like EOG again at x is given as sum over z belonging to, I mean you can of course use all that 1 by m square and all and use that average, I may not write it every time, it does not matter because m is the same so whether you use that or not.

And then it does f square x z plus x square y z, just add the 2. Now, this is normally a 3 cross 3 window over which you are kind of taking these values and this effects and if I read, these are really this is supposed to be a gradient along x, image gradient along the x direction and similarly f y is like a gradient along the y direction and we need to have some kind of a discrete approximation for that because you are on a discrete grid.

Therefore, f x and f y and all we need to have some operators that will be a discrete approximation of really a continuous differentiation, and all those exist. So, we will talk about 1 and when we will talk about that you will automatically know how these can be found out.

So, it simply means that you are computing this gradient, again this also makes sense because if something is sharp then your gradient will also be higher because your transition is going to be higher, if something is very smooth then your gradient changes will also be less so. So, I mean these are all more or less intuitive, so what you would or I would kind of think about.

Then there is something called a Tenegrad, this is based on something called the sobel mask. If we do edge find and grade that time we look at what is a sobel operator, this is a person is name by the way Sobel, and this is given as Tenegrad of x is equal to summation. So, this is the Sobel operator so this gradient computed using sobel and it is given as f x f of z, again z coming from the window of x square plus.

So the only change between the earlier one and this is that this explicitly uses a sobel mask. Sobel mask is supposed to be better because it tends to read it tends to add a kind of a weightage when you do this sort of a different thing you so that is why, none of these may really matter so much because at the end of the day, it is image quality or the image underlying activity, that will actually decide.

But delt, this is S x so this will be a kernel like this, minus 1, 0, 1, minus 2, 0, 2, minus 1, 0, 1. So, those 2, is that additional weight because you are kind of looking for something around this therefore you just give an added weight right along that row, when you are doing an x gradient. And similarly, when you are doing a y gradient, you do like minus 1, minus 2, minus 1, 0, 0, 0, 1, 2, 1. I will ask you something, when you saw a blur curve, we said that it should satisfy certain things and it has certain interesting properties. What do you think is interesting about these two or these operators I mean Del x?

If I write it this way, what do you kind of notice that is interesting?

Student: Sum.

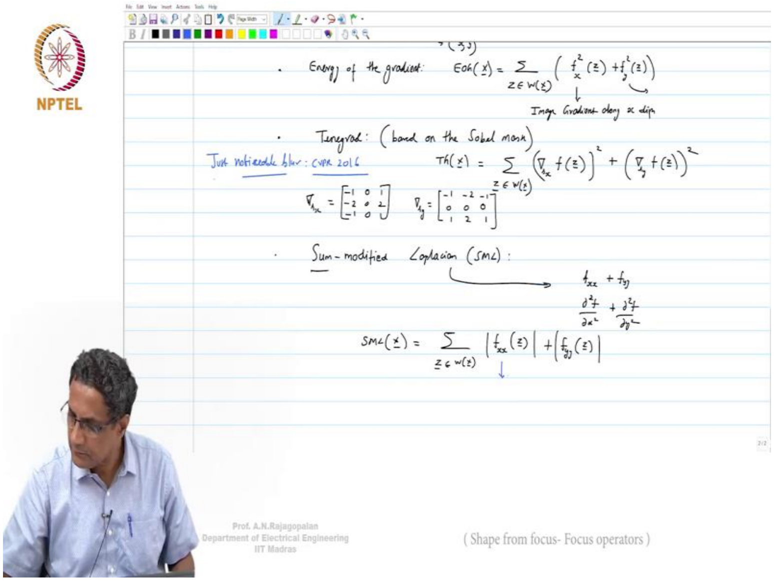
Professor: Sum, yeah they are actually summed to 0, and why should they are summed to 0?

Student: (())(10:28)

Professor; Yeah, exactly. So what it means is if you have an image in which things are totally smooth. So when you blur that is what is expected, when you blur, when you are doing an averaging, what do you do? You actually bring all of them roughly to the same intensity, that is what averaging will do. You keep on doing repeated averaging, that is where you will head. So, the idea is that if things are looking more or less the same, then your focus measure should be such that it should output a low value.

And only when things are rapidly changing inside a local region that is when you want the this f m to be high, the focus value. So, it makes sense that if you have a situation where let us say all intensities are looking very similar, that means either it is very heavy blur or it could also be that the image itself is like that, in either case there is no activity and therefore, your focus evaluation value should be low, so this naturally sum up to 0 and that is the way they ought to be.

(Refer Slide Time 11:37)



Then there is something else that is called sum modified Laplacian. Okay, this is what is most commonly used by the way and this is what Laplacian and I will tell you why it is called SML. And this is going to work pretty well in the sense that not many mathematically optical optimal or something, but this is known to work very well.

So, this is like f x so if you look at Laplacian that if you have a 2D grade, a Laplacian is like f x x plus f y y. So, it is like Dou square f by Dou x square plus Dou square f by Dou y square, this is what is actually a Laplacian. So, this is called a modified Laplacian and there is also some tagged along so, I will explain all that to you. So, what is SML operator does is it works like this, the SML of x is given by again z belonging to w of x and then f x x at z bar and f y y mod, so the mod is taken and then you take the mod of each of these secondary derivatives and then you add them.

And of course, as you keep going higher at the order, you will naturally believe that these have to be more sensitive to any kind of changes in the intensities. But at the same time, they could also become more sensitive to noise that will also happen, but overall it looks like this is a good sort of a compromise between the sensitivity that you need versus the ability to add to sort of handle noise, to kind of say counter noise.

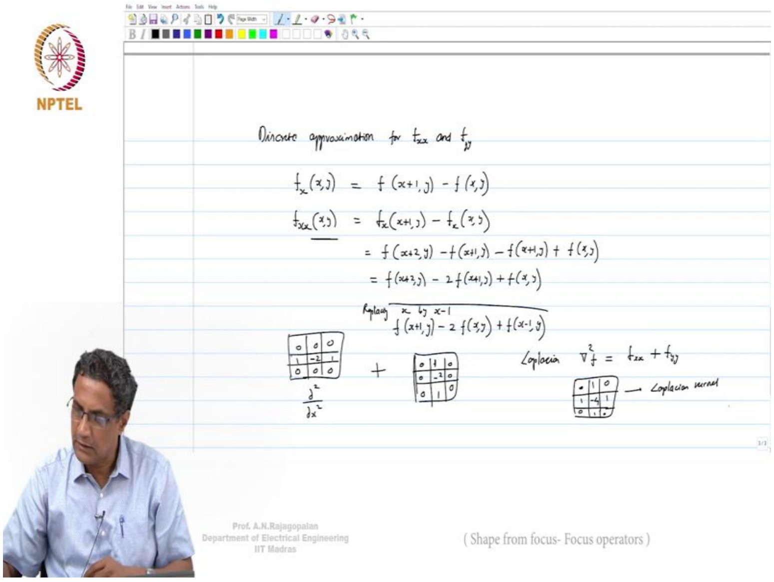
And by the way there are some very interesting papers for those of you who are interested, 2 years ago or something there is a group from Hong Kong that showed that what was called I think the paper title is remember it is called Just noticeable blur.

I think that is how the main this one, it has a title like that, Just noticeable blur. And there they show that even with a kind of regular camera if you to take an image to a humanoid it looks like, everything looks like it is focused. But it turns out that there is a subtle sort of change and difference is in blur across the image that can all act as a cue for depth and these guys do a fantastic job of actually bringing all that out. So why I am saying that is these operators and all, right? Of course they do not do it using this focus operators, they use something else. But I am just saying that this is what you want from these focus operators.

You want them to be extremely sensitive to any changes in blur because that is what will help you tell where the sharpness is exactly happening. So anyway, for those of you (())(14:36) this was from CVPR 2017 or something, or maybe even a little older than that 2016. But it amazing, we look at those images, to your naked eye it looks like they are all pretty much in focus, but then they show that it is not the case and you can actually start inferring depth of a scene just from naturally captured images with a cell phone. And now I will just expand on this operator it a little bit.

So there is some modified Laplacian, so it is called a modified Laplacian for this reason that a Laplacian would have simply summed up rate effects, or has been simply a sum of f x x. And this f y y, whereas in this case, it takes f x x, but then takes the mod of that, right? And f y y takes the mod and then sum sub the 2 mod, that is the way they call it, that is why they call it some modified Laplacian. And if you think about it, I mean, suppose we wanted a discrete approximation, just to start with, we will just do it for this case, f x x what would it look like? So these are all standard things, but we will just do it here. So a discrete approximation because finally rate, we need some kind of a kernel that you want to run on the image.

(Refer Slide Time 16:04)



So a discrete approximation for f x x m and f y y, whereby f i mean the image, so I do not know we have been calling g as the image, so if you look at this is g x x, g y y. Now, all right, I think g is what I have been using right all along so I will say, g y y. No, no, in those expressions I think I have been using F, it does not matter use f or g okay f x x, f y y, alright. So, f x, so the normal, the very standard approximation is take it as f of x plus 1 y minus f of x comma y. I mean there are various approximation, one very standard base f of x plus 1 minus f of x, y.

And which then means that if we are looking at F x x of x, y then you should kind of be looking at F x x of x plus 1 comma y minus F x of x y. It will be the f x x, but then f x of x plus 1, it will turn out that from the first one it will turn out as f of x plus 2, y minus f of x plus 1, y then what is it, f x of x plus 1. So, f x of x plus 1 and x plus 2 and then minus f x of x y that is f of x plus 1 y plus f of x y. So, which then means that you get some like f of x plus 2, y minus 2 times f of x plus 1 comma y plus f of x, y. Now, since this is centered about x plus 1 and not about so like I said that you want to wait at x, y, not at some other location.

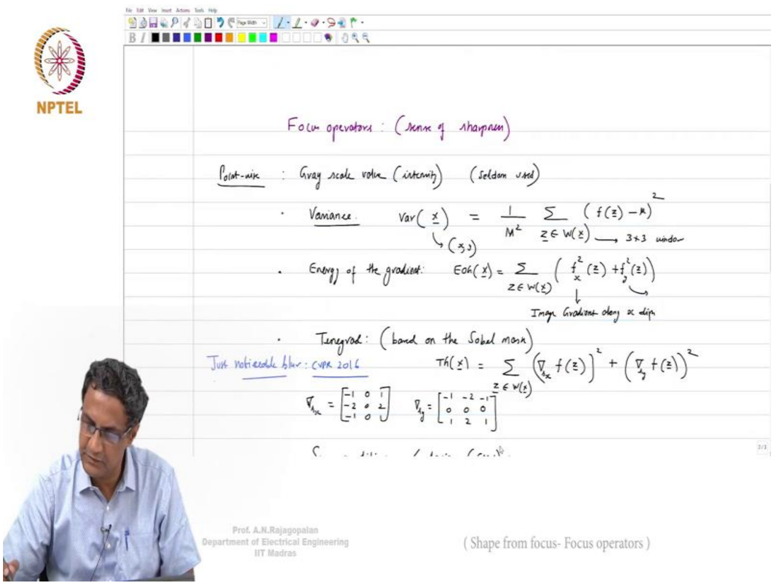
And you are looking at looking at finding f x x at x comma y, whereas this is putting more emphasis that actually x plus 1 than x, therefore, what is standard practice is to replace x by x minus 1 in this expression so that what you end up getting is x of x plus 1 y, so replacing x by x minus y in the above expression, that is purely done to in order to give more weightage to x, y where you are trying to evaluate the second gradient, minus 2 times f of x plus 2, x comma y plus f of x minus 1 comma y. So, this is in fact a very standard, so if you try to write it down in a kind of a 3 cross 3 form, then what it will mean is you will have a minus 2 wait here, you will have 1, 1 here and then 0 0 0 0 here.

So, this will something like Dou square by Dou x square or what they call as f x x. And then along y, you can similarly show that if you do the same thing along y, you will end up with this what is it, 1 here, minus 2 here, 1 here and 0 0 0 0 0 here. And a standard Laplacian when you write Dou square f which is a Laplacian that is f x x plus f y y, a Laplacian typical symbol is this.

And then if you add these two kernels, then what happens is what you will end up getting for a typical Laplacian is minus 4 1 1 1 1 and then 0 0 0 0, so this is a standard sort of a discrete approximation for a Laplacian kernel, but what we want is actually a modified Laplacian, but this is what is really a Laplacian kernel, the simplest approximation that you can make for a Laplacian kernel.

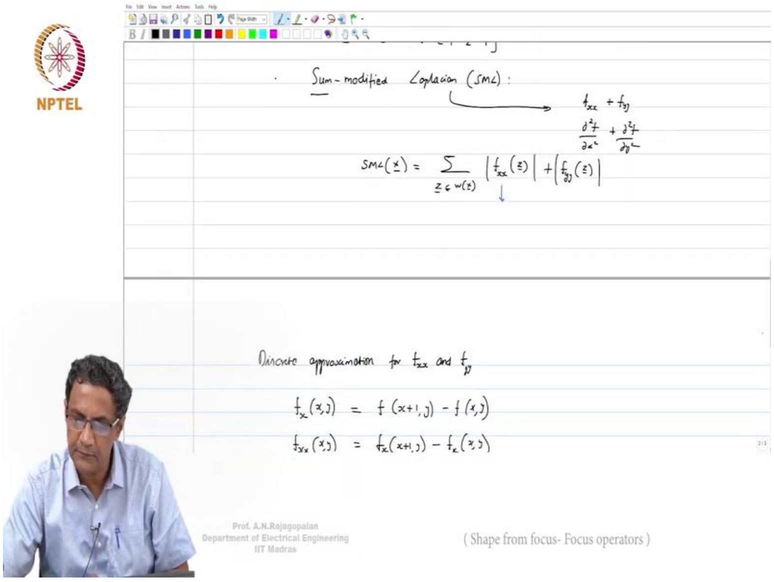
So again this will also add up to 0 and all that because this is also supposed to give out a value which will be low is if all the intensities are similar looking, and high if there is a lot of activities, all of that this will also follow. Coming back to that modified Laplacian, so what it actually means is that you if you go back to that thing, so what you had was this one.

(Refer Slide Time 20:37)



So where is that?

(Refer Slide Time 20:42)

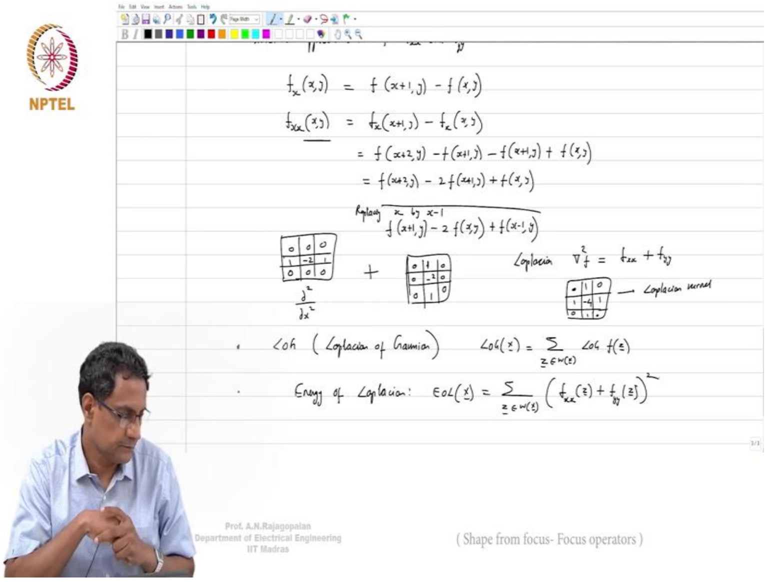


So see if you see here, so this is F x x, so F x x, so what it will mean is so you will replace it with this kernel, it will be minus 2 1 1 and then 0 elsewhere. Then, so this will be a kernel that will be, so the way they operate is like take these individually, and then operate on the image to get the gradient, and then take the absolute value, then again take this kernel, it operated on F, which is like doing a simple convolution operated on F, and then get the second gradient along y, and take the model S of that and then add the 2.

So, ideally you will get one array that comes out of this, one array that comes out of this, another array that comes out of this, you are just taking the mod, of course after you take the mod you will get these two arrays and then you just do this adding of those values to get another array that will that is supposed to give you the value of SML at all locations.

We will talk about the implementation a little later, okay. So the way we actually implement is like we do not go like take one pixel at a time and then do this, we would rather operate on the whole grid at one at one go, because doing a convolution is fast, we just operate in one go, get all the values and then do all these local things.

(Refer Slide Time 22:22)



Then there is one more operator, let me also write that down. This you may be familiar with what is called a Laplacian of the Gaussian what is called log. Log of x, so summation again z belonging to w of x log of anyway, so I am not writing this. So for those of you who are interested, you can find out, again, it is a sort of a convolution.

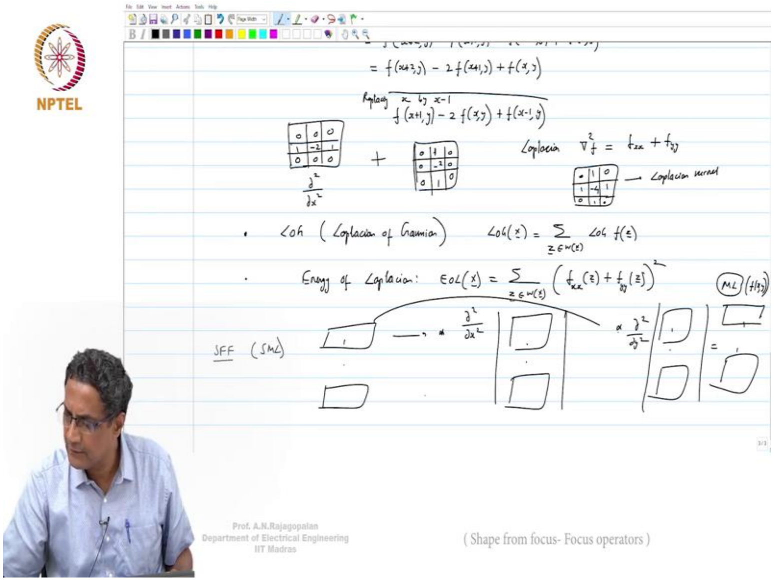
So what it will mean is, did we talk about how to I think, you know what, we did that. So, if I gave you a sigma, then we know that how to actually have a kernel for that a discrete kernel. So Laplacian of the Gaussian is like you have a Laplacian kernel, which I showed you just now. And then you have an image. So log of the Laplacian of the Gaussian is like, you take the Gaussian kernel, blur the image first with that kernel and then apply this operator Laplacian which is then going to convolve.

So, first is a convolve with the Gaussian, so it means that it is going to probably blur that image a bit and then on that you operate a Laplacian kernel. There are also direct kernels that can give you the log itself directly, but the simplest way to think about is blur first with actually a Gaussian and then follow it up with the Laplacian kernel.

You know how a Gaussian kernel can be written, similarly you know now, how the Laplacian kernel can be written. So, those of you right over there who just want out of interest to do something like this, you can still go it, I am not writing those expressions here, because simply means one kernel for the Gaussian followed by another kernel for the Laplacian.

Then there is something called energy of Laplacian. So this is called EOL of x and this is given by summation belonging w of x, and f x x of z plus f y y of z the whole square. S if you see that, this is not the same as we modified Laplacian where you took each one of them individual mod and added up, here it is like you are just taking like f x x z plus f y t. So all these, and in all of this right we do not know which one of these, nobody knows for sure which one of them is most ideal to use and so on.

(Refer Slide Time 24:35)

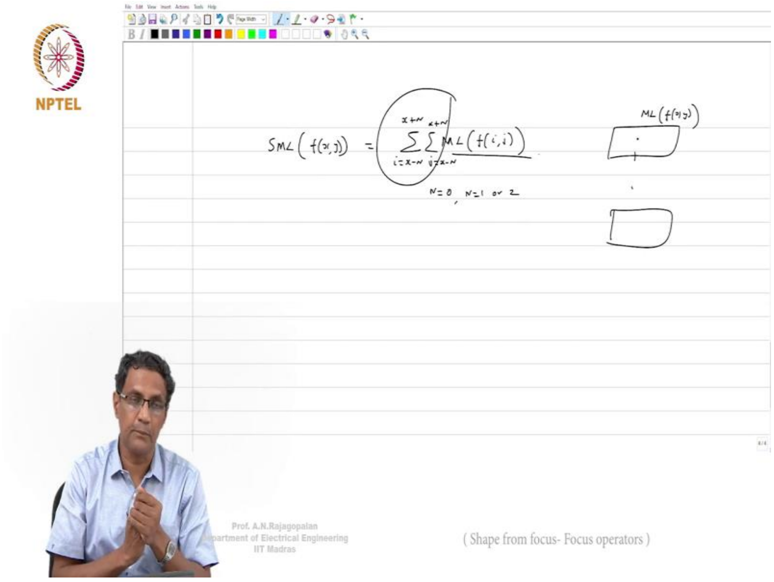


Let me talk about some part. So it is like this, so modified Laplacian because of the fact that what you do, so you will take every frame and act let us say you are an operator Dou square by Dou x square which is this kernel that I wrote down for a second derivative. We will simply do a convolution with each frame, output will be another frame, so you will get again another stack of frames if you have to do convolution.

Similarly, you will use the same frames and then you will also do a convolution with Dou square by Dou y square, which is again another kernel that we have whatever, 1, minus 2, 1 and then the other case is 1, minus 2, 1. So you convolve and then again you will have, let us say one more stack.

Now, what you do is you take the modulus of this, you take the modulus of all these values, because that is a way to do it fast, you do not want to travel one pixel at a time and all. So if we take this modulus, and then if you add them, then we will get one more array or one more stack, which will have all these ml values. That is called like, mlf if you want to call f of xy, then all the modified Laplacian values are all actually sitting here and the sum modified Laplacian looks like this. So, when we say sum modified it, what we really mean is this.

(Refer Slide Time 26:11)



So, when you write SML, suppose I say at f of x, y because finally I want some x, y what is the sum modified Laplacian value, then what I will do is, I will take it to be ml, now this will be a double sum over let us say i is equal to x minus n, 2 x plus n and j is equal to x minus n, 2 x plus n, I will tell you what this n is. And then what you have is a ml of f of i , j. So, that is why it is called a sum modified Laplacian. So, what it is saying is, see you have this array, you have this stack that you got which is actually ml of f x y, at all x y you have the values now.

So what it does is, in order to be a little more robust to some noise and other things, so what it is saying is, it is kind of add up the values in and around your x, y. So for example, if you take n equal to 0, then it means that it simply takes the ml value right there, and then uses it as the value for SML.

But normally, what is done is we will take less than equal to 1 or 2 or something, because 1 will mean that it is looking 3 cross 3 neighborhood, adding up all those values. Whereas if it is n equal to 2, then we will be looking at a 5 cross 5 neighborhood, adding up all those values in order to tell that but then the point is that you cannot make it too big because you have a depth that is actually varying, and therefore it will not make sense to take a very large neighborhood.

So you can either go with n equal to 1 which is what is standard, typically go up to n equal to 2 or something but not beyond 5 cross 5 and all because otherwise you will end up smoothing your depth plot. So this is again, this is another way of implicitly accounting for noise. So if there is some variations that you are not able to see pick up, or let us say D bar itself for some reason, okay? Is there is, if you feel that just trying to get values at one location is going to be a problem, then you could actually look at a small little sort of a neighborhood, that is why it is called some modified, so the sum is coming from this sum okay, some and then the sum of the modified Laplacian values, that is what it means sum modified Laplace.

And then what this will mean is that once you have the some modified Laplacian value, you can actually plot it for every x comma y now. It will just traverse through that stack and you have the SML at x comma y coming out from every stack, and then you have that f m plot then you can actually do this Gaussian interpolation, then you can get your d bar. So if you use one operator now, the point is right I mean who knows which one is the best and how does one, so there are a few ways out of that.