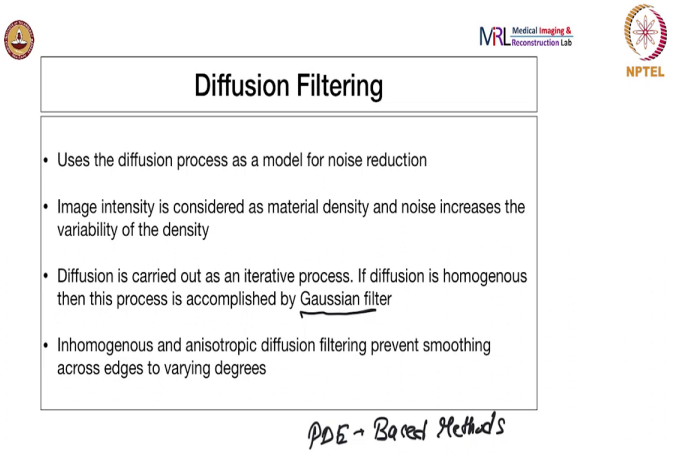
Medical Image Analysis
Professor Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology Madras
Lecture 11

Diffusion Filtering

(Refer Slide Time: 0:15)



So, welcome back. So, in this video we are going to look at what is called diffusion filtering. So far we looked at the best option that we have, which is a median filter. So, what it does is it does a good job of preserving edges but sometimes it may also be outrageous because it does preserve edges only if I had just represented a certain orientation they have straight lines and it also depends on the intensity values.

So, a diffusion filtering is an alternative to median filtering or other filtering approaches wherein we can prevent edges from disappearing and we can also allow for some smoothing near the edges. So, diffusion filtering comes under the purview of the PDE based method so-called sorry.

So, it is one of those PDE based methods in image processing. We look at this topic later in more detail. We will try to cover this also at that time but now it is good to have an introduction to this whole thing. So, either way it works if it treats the denoising process like a diffusion process. So, basically what it means is that it treats the intensity as density.

So, if you look at diffusion you think of material flowing from regions of high concentration to that of low concentration. So, if you think of an image you have noise in the image noise is basically some region or some a bunch of pixels with abnormally high densities. So, you would expect that if you can diffuse those high intensities out that will be the equivalent of denoising.

Another aspect of this process is that now we would like to do this diffusion of this abnormally high values but we do not want to do this across edges. So, we want to preserve edges while doing this. So, one way to know where we are at the edge because we do not know where the edges are beforehand. So, there must be some way of figuring out where the edges are. So, there could be abnormally high values near the edges also.

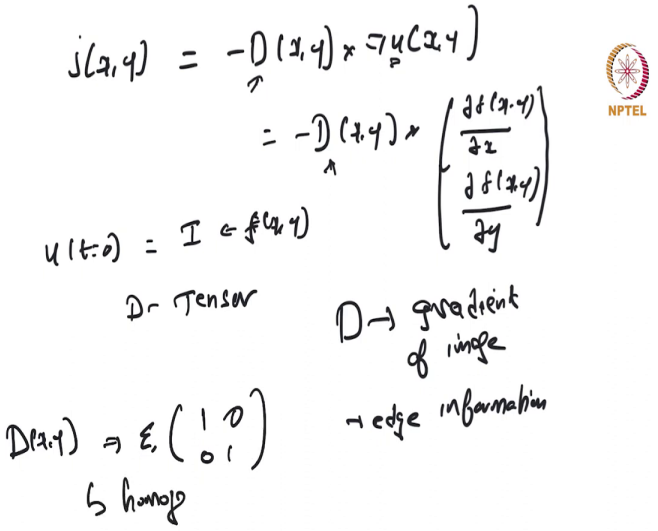
But then the gradient directions are along or similar. So, we can actually restrict this diffusion process near the boundaries and have them and have it go on away from the boundaries or in regions of or in nearly homogeneous regions in the image now if we assume that that if we do not distinguish between edge and non-edge pixel those are means then this this diffusion if it carried out is the equivalent of a gaussian filter.

This corresponds to what this is what you can call a constant diffusion coefficient. Inhomogeneous and anisotropic diffusion filters prevent smoothing across edges to varying degrees. So, anisotropic division and anisotropic diffusion allows for the smoothing along a boundary but not perpendicular to it. So, it makes use of the gradient direction.

So, we know the gradient and the direction of the gradient at the boundary seventh but gradient is a vector we solve it as two components. So, you can figure out the direction of the gradient at the graded point. So, you figure out which is the tangent to the boundary and which is the normal to the boundary and you only allow for diffusion along the boundaries. So, that way you do not do the diffusion across the boundary.

So, inhomogeneous diffusion does not care about this direction but it does treat the boundary like some kind of a semi-permeable thing. So, basically the degree of smoothing is lower across boundaries when you are doing a diffusion filter. So, let us look at the formulation of this.

(Refer Slide Time: 4:09)



So, this is formulated in terms of diffusion equation starting from fixed law a fixed law of diffusion where you know the flux j is given by the following equation $j(x, y) = - D(x, y) * \nabla u(x, y)$. So, which is just to make sure that we understand D(x,y) multiplied by this u gradient of u, u is nothing but f. So, we will see where that f comes from and f is nothing but the image but then represented in the continuous domain we can of course discretize all of this which is how it is typically solved.

So, this is a gradient and it has two components. So, the u is basically at time t equal 0. So, at some point we will also see that this is actually stepped across in time. So, u is nothing but u at time t equal to 0 at this point I will just say it is just your input image which we actually represent by f of x.

So, see the type of diffusion that we talked about where inhomogeneous or anisotropic diffusion is described by this diffusion coefficient now this is what is called a tensor. Now it has multiple components. You can have so, diffusion tensor of multiple components or multiple elements in the matrix that specifies it.

So, what it does is it tells you for a given gradient u density gradient u this is basically remember this is the image at some time what are the directions different directions and how much

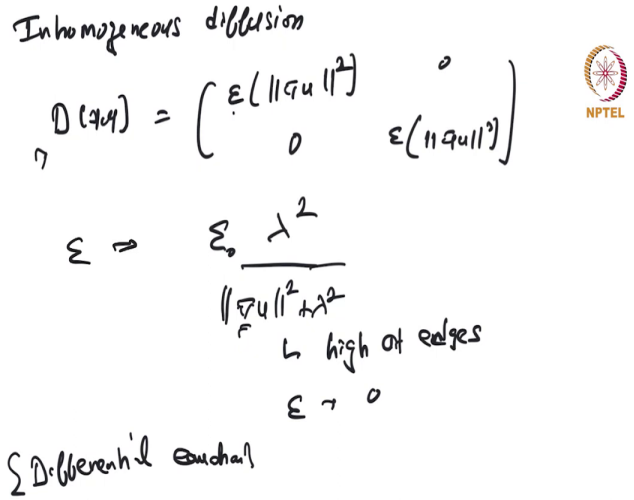
diffusion is allowed across different directions. So, that is what this D specifies. It tells you whether smoothing has a degree of smoothing allowed in a different direction.

So, basically what eventually happens is that is very similar to what we call this box car averaging filter there is some averaging of the neighborhood of a pixel and then replacement of that value but what this D does is to modulate that averaging by saying along this direction you cannot average too much but if you go left to across pixels then you can replace with the averages and you so, that is how it prevents smoothing across edges.

Because what is this D? D is derived from the gradient of the image. Actually, they are in the gradient of image. So, it actually has edge information and network information is what is used to modulate the smoothing. So, the homogeneous diffusion which we talked about which is equivalent to just doing or Gaussian smoothing is independent of the strength and direction of the gradient it is just some constant.

So, D is basically in this case 1. So, for instance if you are looking in two directions then D will be a 2 cross 2 Matrix. So, D of X Y in the case of homogeneous diffusion will just be some constant times your identity Matrix this is homogeneous in the sense it is the same across all directions.

(Refer Slide Time: 7:36)



On the other hand, inhomogeneous diffusion depends on the strength gradient strength and so it is basically some function of the gradient and so we can write in the case of inhomogeneous diffusion. We can write D(x,y) as some epsilon, some function of the gradient of your image, usually it is squared. So, it is the same here: some function of the gradient of u squared again this becomes 0.

Once again there is so it is in this case that what we are looking at is the magnitude of the gradient and not at the direction. So, a typical function this epsilon is basically a function which is $\epsilon = \epsilon_0 \frac{\lambda^2}{||\nabla u||^2+\lambda^2}$. So, what it does is so if this is basically your diffusion coefficient. So, the diffusion coefficient is very low when gradient is very high, that is what this formula tells you.

Now, whenever the gradient is very high, the gradient is very high at the edges; this denominator is high at the edges. So, pretty much at those points this epsilon goes to 0. So, there is hardly any diffusion across edges but then it is allowed. So, if you have a very large lambda then this gradient of u does not matter. So, it just becomes epsilon naught. So, the degree of this innovation rate depends on lambda.

So, but this so-called tensor that we have here for D we say it is still a diagonal. So, there is no constraint based on the direction of the gradient of u. So, because the rate of u is a vector. So, there is no constraint based on gradient u. So, for let us say for anisotropic diffusion where the diffusion at is defined this this D is again defined at every point in the image every pixel in the image.

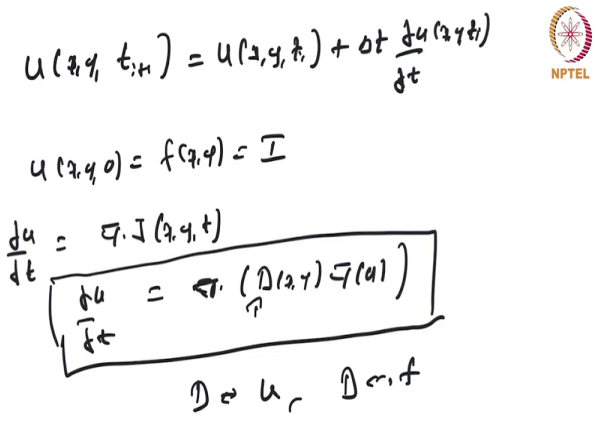
And for anisotropic diffusion the gradient the division coefficient is such that in the gradient direction along the what is it along the edges which is which you can get from the gradient direction there is a defusing it is fine while at diffusion perpendicular to the gradient is not a sorry in the sense direction along the edges. So, let me repeat this along the edges it is allowed diffusion is allowed and perpendicular to the edges diffusion is not allowed.

So, the gradient which you calculate in the image correctly is basically perpendicular to the edges and. So, along that direction there is no diffusion while perpendicular to the gradient which is along the edge there is diffusion. See what typically a person does here is to do the so-called eigenvalue decomposition of D.

So, and then from there you can find out the direction of the largest eigenvector and then you do a diffusion along that direction. However, we can write down a formulation for that but right now we hold off on this particular aspect. So, what all we can say is that D is defined to be a tensor and for anisotropic diffusion we calculate the directions along which where diffusion coefficients are low and directional longitudes are high and we solve the diffusion equation appropriately.

So, this diffusion but we keep saying that we have to solve it involves solving a PDE or a differential equation. And so the form of the PDE let me write that down and in this case when I said if you did a different differential equation we have to solve it iteratively. So, how are you going to solve that.

(Refer Slide Time: 12:12)



So, we are going to say let me add this I am going to say that this $u(x, y, t) = u(x, y, t) + \Delta t \frac{\partial u}{\partial t} u(x, y, t)$ and we saw that u(x,y,0) is your image input image but what is $\frac{\partial u}{\partial t} = \nabla . J(x, y, t)$ and the divergence of your flux is which is we saw here is nothing but which is equal to $\nabla(D(x, y)\nabla u)$.

So, d u by delta, so, this equation delta u by delta t this is what you solve with appropriately chosen D capital D. So, given this we can go ahead and solve this. So, there are many ways

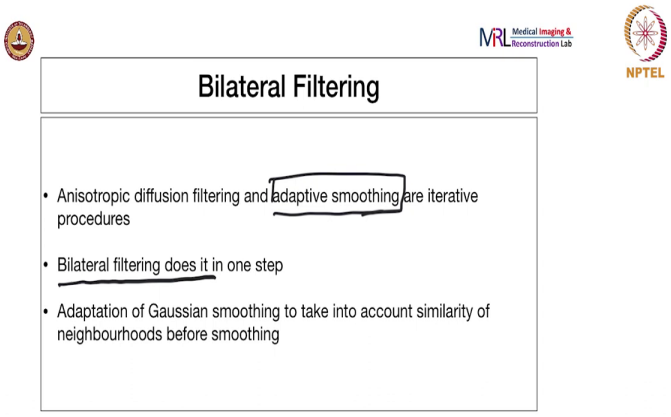
where you can bring this out. So, for instance even though this is especially valuing this equation can be made on a linear or non-linear by your by how you compute D.

So, for instance the most general form D is computed from u every at every iteration you can also compute D from f. So, it does not change every iteration. So, both are possible once again. How many iterations do we have to do that depends on the type of your problem, some of the parameters in the diffusion equation etc . So, we look at this if time permits in more detail. So, but just to give an understanding so everything depends on how you choose D.

So, you choose d t by tensor which has preferential directions then you get an isotropic diffusion wherein you get smoothing along the edges but not perpendicular to it because D depends on the gradient and if you choose due to be some constant then it averages across everything but you can choose D to b proportional to the magnitude of the gradient alone but not worry about the direction then you get in homogeneous diffusion there is some smoothing of the edges but not as much.

So, this is one of the techniques that is often used for quite some time still in use as it requires a lot of fine tuning but it works very well to come from sound principles.

(Refer Slide Time: 15:02)



So, the next topic is bilateral filtering. So, here this anisotropic diffusion filter an adaptive smoothing they are iterative and this I have talked about which will exclude this topic or iterative procedures but bilateral filtering does it in one step and what it does is basically some weighted averaging but the weighting factor takes into account similarities of neighborhood.

So, what do you mean by similarity of neighborhood? So, when you saw that remember I told you when we do this Boxcar averaging we can for the waiting in a Boxcar averaging all the pixels in the neighborhood are weighted the same but then you can put a Gaussian weighting on the neighborhood of pixels but then that Gaussian weighting seems to be like a function of distance correct.

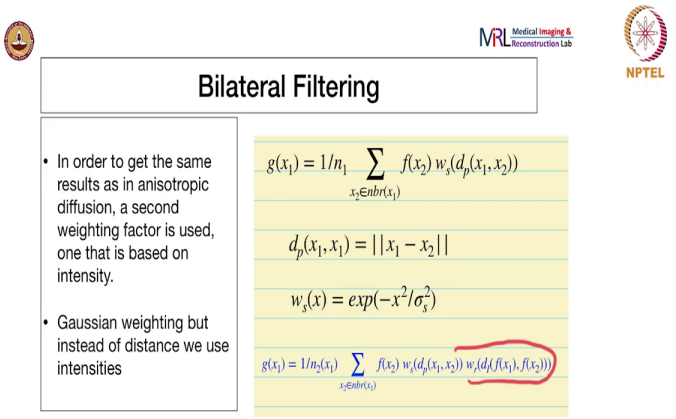
So, you have a central pixel as you move away from the pixel the neighborhood's influence reducer. So, you kind of made sure that you took that into account but here there are two things one is not only the distance from the center pixel but also the similarity in terms of the pixel intensity is themselves. So, here also it is weighted averaging only except that you average pixels that are similar rather than pixels that are similar this similar.

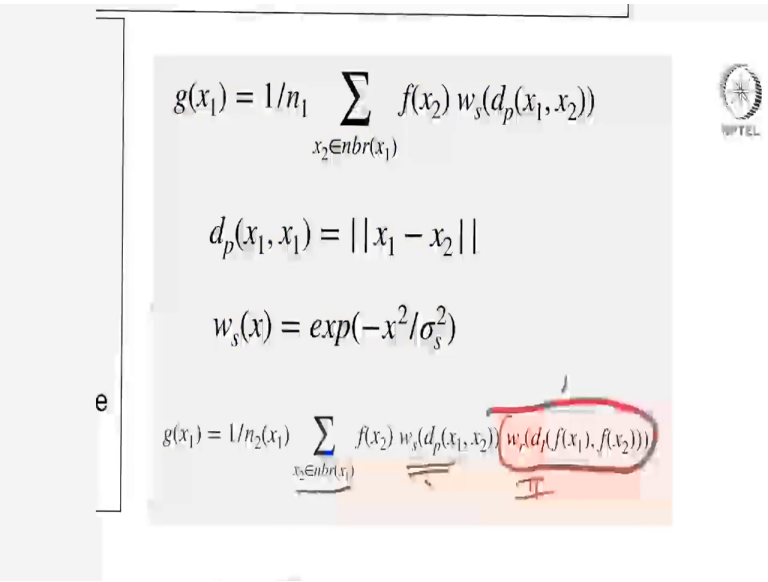
So, for instance I will give you an example let us say you take away a picture of a person a face image now if you want to average inside the eyes the iris etc you would like to consider pixels in

the other eye which we are in all the you would expect them to be similar to this in the this pixels in the two eyes to be similar.

So, given this scenario you would like to choose pixels in neighborhoods which are similar in pixel intensities not just distance. So, that is that actually improves the outcome from this filters quite a bit and bilateral filter is again a very often used filter and it is one of the and there is some problems with in terms of execution time but it actually does quite well.

(Refer Slide Time: 17:13)





Just look quickly at the formulation. So, the first level of formulation is basically this is what you have seen is you have a Gaussian weighting. So, if you want, let us say exactly 3 by 3 neighborhoods. Let us say we want to replace this pixel with a weighted average of the neighbors. So, that is basically so, if you consider a neighborhood of i j. So, x1 is the pixel you want like here and x2 basically represents locations which are neighbors of x1 which is basically the other pixel locations point here this is x2 the red ones are x2.

So, you take the function values there f(x2) and then you have a weighting function and what is that weighting function the rating function is nothing but it depends on the distance between x1 and x2 which is what I have written so which is if you subtract out the components corresponding component in this case there are two components for x1 two components for x2. You have let us say x1 is i, j and x2 is m,n. So, you have that.

So, this is just a pixel index distance and then you convert that into a weight by plugging it in here. So, this dp of x1 and x2 is ws dp. So, you calculate the distance between two pixels x or x in this case and then you plug it in this formula. So, you get exponential weighting. So, this is one level of filtering this is very similar to Boxcar averaging where in this case all the weights are 1 but in our case we just have an exponential weighting.

What is the other version of the same thing is if you further weight it with not with what you call the wr here I do not know hope you can see this zoom this in usually s. So, you can see that there is ws we saw this that we saw till now you just add on top of this you also add another filtering thing which is basically the same weighting except that now it is the distance between pixel intensity.

So, you look at neighborhoods. So, you look at average neighborhoods and see which ones are similar. So, here you predefine the length of the neighborhood that is how you would calculate it, now you can see that this neighborhood is how you define it. So, you can choose pixels from very large neighborhoods, not only are they weighted by distance. So, if you go farther away from your center pixel obviously the influence of that neighborhood goes down but you can always see that I am far away but it is very similar that is what this tells you.

So, I might be far away from my center pixel but it looks like the neighborhood there is very similar to this neighborhood. That is what you get from this other weighting function correctly. So, these two are what you call weighting function is what their play between them is what gets you a better averaging.

So, eventually what you are doing is averaging pixels initially you are averaging pixels close by on the assumption that in a very small neighborhood of the center pixels, pixels around it should be similar. So, let us say I was giving you an example you are looking at a blue sky and if you take a small neighborhood in that picture of a blue sky and it is all be blue cannot be anything some random a purple color cannot come up maybe there are some exotic times when that happens but mostly that is true.

So, the locally constant assumption is true there, here not only do you not assume that is true but you also say that there is redundancy in the picture. So, you can for instance take a picture of some rolling fields that might have some greenery, some water bodies etc and then some more gradients.

So, there are all these green patches everywhere in a picture of nature and they are all very similar objects. Let us say grasses are green so, then it is okay to average pixels from those areas also and that is what this intensity weighting does for you. So, there are various modifications. So, as you can see this is a very computationally intensive process.

So, for every you can take this into the extreme by considering for every pixel you can consider the entire picture or you can consider like a sampled areas in the image all kinds of things you can do and that will increase the computation time. So, one of the initial criticisms is that it takes too long and gives you exceptional results.

But there have been quite a few innovations in this field and people have figured out how to do this fast. We will not look at those algorithms but just give you information about that. So, bilateral filters once again open source implementations are available. I will ask for you to try them out . I will show them in a later lecture a demonstration of bilateral filtering on images. You can add noise and then do the filtering.

So, in case you are wondering how I know whether it is working or not, you take a perfectly nice picture, add noise to it and then do a bilateral filter to see if it removes it. So, that is the end to our filtering in terms of all these kinds of methods for noise removal. So, in the next few lectures next lectures not few lecture next lecture we will look at the Bayesian estimation just again a brief introduction there because the problem itself is very complicated but just to give you an understanding of how to how that problem evolved. Thank you.