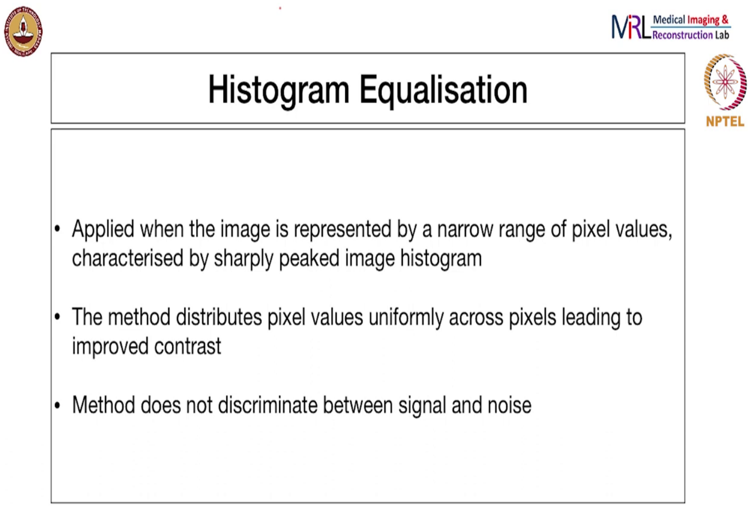
Medical Image Analysis

Professor Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology Madras

Lecture 08

Histogram Equalization

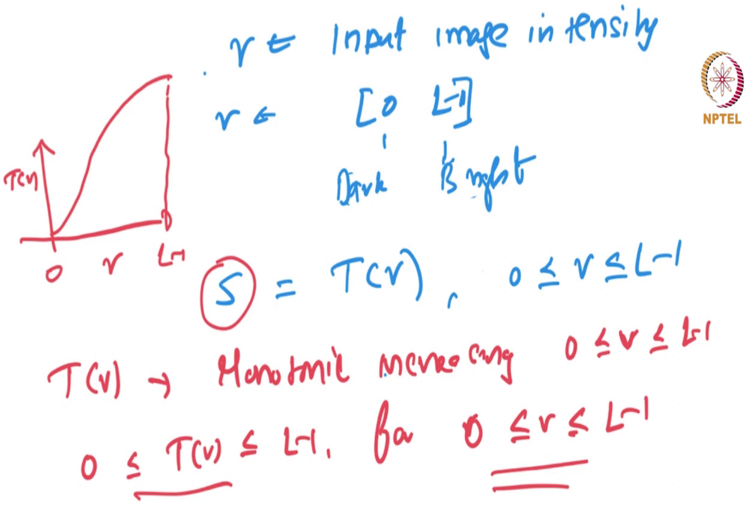
(Refer Slide Time: 0:14)



Hello and welcome back. So, in this video we will look at histogram equalization which is again a very useful preprocessing step and simply used when the image is represented by a very few or very narrow range of pixel values and it kind of shows up in the histogram when you see that histogram is sharply picked in the sense a lot of pixels have a similar range of values.

And what this method does is to distribute the pixel values uniformly across pixels leading to an improved contrast because the drawback with this method is that the weather does not discriminate between signal and noise. So, which means that you might end up corrupting the image quite a bit.

(Refer Slide Time: 1:02)



So, let us do some preliminary notation. So, we assume some continuous intensity values let us just start with a continuous domain. So, r is the let us just write this down in this case r is the intensity of the image to be processed input image intensity. So, we denote the pixel values in the input image by r. We also assume that r is in the range 0 to L. So, that is typically assumed we can always assume larger ranges if you want to typically 0 to L. Of course when you go to a real image we will have to where it had values where pixels are discrete etc we will have to change a little bit but at this point 0 to L. r takes on values 0 to L.

So, we can say that r equal to 0 is dark or black and L is bright and keeps spilling over everywhere. So, 0 is a dark pixel L is a bright pixel that matches. So, for given this r. So, we actually it is [0 (L -1)] which gives you l levels correct. So, L levels if you go to just integer values but in this case we are going to assume a continuous intensity value.

So, given this r we want to find the transformation of the form S. I want to find a transfer in the form S=T(r). This is the transformation we want to find for $0 \le r \le L-1$. So, the idea is the to produce, produces an output S this is output S and the assumption is that T(r) is a monotonic increasing function autonomic increase in the range in the interval $0 \le r \le L-1$.

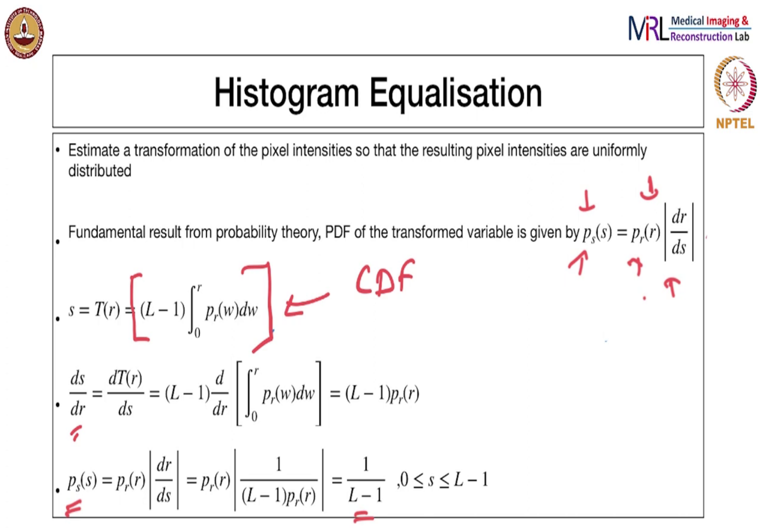
And if union after transformation $0 \le T(r) \le L-1$ for $0 \le r \le L-1$. So, this is basically just a sanity check if you can think of it. We want a 1 to 1 mapping. We want to go from r to S and we should be able to come from S to r. So, lower values of r should lead to lower values of S higher values of r should lead to higher values of S and the ordering should be preserved.

Because we want dark pixels to be dark bright pixels to be bright bright except that we want improved contrast and that we do not want we do not want also do not want to go outside the range of values that you work with. So, if the rgb for instance rgb images are between 0 to 255 when we do the so-called histogram equalization we do not want suddenly have pixel values like 500 and all that it has no meaning.

So, which is what this ensures. So, it is a monotonic increasing function and it also maps it into the same range. So, that is the idea that is the conditions for this histogram matching. So, let us draw some of these functions. So, for instance something that is allowable let us say would be something like this where this is the l levels r and this is T(r). So, that you do not have within this range within 0 to L-1 you do not. I worry about what happens to it beyond these ranges but within this range it is 1 to 1 and it is monotonically increasing or similar.

So, you cannot have multiple valued functions that go only flat etc. We cannot have functions like that. So, that much is understood.

(Refer Slide Time: 5:50)



So, what we want to do is to estimate a transformation of the pixel intensities from the source image. So, that the pixel intensities are uniformly distributed in the finally the synthesized image where the which is what I just are what we call the synthesized image or the transformed image.

So, you want to transform the pixel values in your input image according to some smooth function. So, that you get another new image and if you plot the histogram of the pixel values there that corresponds to a norm uniform distribution. So, there is a fundamental result from probability theory that the pdf of the transform variable is given by this expression right here all where r is your source the r subscript r refers to your source image that is the original input image.

And S refers to the image that you are synthesizing using this transformation function. So, why am I suddenly using the pdf probability density function. So, where did that come from? So, like I said if you plot the histogram of pixel values and then you normalize each bin in that histogram by a total number of pixels what you get can be approximately considered the pdf of the pixel distribution in that particular image pdf of the pixel intensities in that particular image not the pixel distribution I mean to say pdf of the pixel intensities in that image. So, it is actually an empirical estimate of the pdf; it is not the true pdf empirical estimate of the pdf.

So, what we are trying to do is to create from that pdf we are trying to create a uniform pdf. So, just for this reason, just to show you what uniform pdf which I already did in the last step let me just create a new one. So, what is the technique for doing that the idea is following. So, if we think of let us say pr and ps to correspond to probability distributions of intensity values r and S are intensity values they correspond to probability distribution of intensity values you can think whenever I say probability distribution of pdf think histogram it is fine just for understanding.

So, you can use r and S to correspond to the subscripts corresponding to the intensity values in two different images. At this point let us assume they are images. So, the fundamental result from probability theory is that p subscript r and t of and t of r are known. So, if you know p(r) and T(r) and also that t r is continuous and differentiable over the range of values of interest. So, then the pdf of the transformed variable s.

So, r is our original. Let us say in our case given the image we need we denote it by r. So, p r of r corresponds to its histogram from which we derive its empirical pdf and if we know some if we know the transformation function t of r we know that it is a continuous distribution then we can write ps the distribution that we obtain of the transformed variable as

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

Now, if you are wondering why it comes out this way, why this is possible it is because we will always remember the relation is that S= T(r). So, we have a function T which takes our input image pixel by pixel its intensity r and outputs an S for every pixel intensity r it outputs an S into the new image.

Now if t of r is continuous and differentiable then it is possible to estimate this the target or the sorry the synthesized pixel distribution. Now, we can do this by considering S= T(r) and we contend that the function T(r) that does this is nothing but this cumulative probability distribution; this is the cumulative cdf cumulative distribution function.

Cumulative distribution function for any probability or distribution let us say is nothing but if you have $\int_{0}^{x} p(x)dx$ That is how you define the cumulative distribution that is this right here. So, the transformation that enables. So, if you consider this transformation this is again we can clean and show that this is continuous and differentiable for all ranges of all pixel values of interest and if you put it in this form L- 1 again it corresponds to there are L levels correct and so, we can write it in this form.

So, this is the function we are claiming that this function does the job for us. So, this function is what makes your p(s).a uniform distribution. So, we can prove that but there are several things that its continuous and differentiable. You can prove that because our pdf if we know that we can from there we can show that this is actually true.

So, from here we can actually calculate ds by dr we actually calculate ds by dr from which we can calculate dr by ds it turns out if you do a lot of algebra. So, we will end up getting p S of S to be 1 over l minus 1. So, this might seem a little bit rushed. So, let me just let us take out the clutter and so, just understand what we are trying to do one second.

So, we started with so we started with this given this image given an image we know that its pdf or histogram is denoted by $p_r(r)$Now we want to find a T(r) which gives you S. So, for every pixel intensity r in the image source image we put it through this function T and it gives us an S and from the basic results of probability theory we can show that this is true that $p_s(s)$ that is the probability distribution of S is given by this for this function.

This is under the assumption that this t of r is continuous and differentiable. Now, the special case of T of r which is given here this is what we looked at $(L - 1)\int_{0}^{r} p_r(w)dw$ is just a grumpy variable p r is the input image pixel values in the distribution pixel intensity probability distribution is p of r this w is nothing but the dummy variable of integration of course this integral itself is nothing but the cumulative distribution function.

And L minus 1 where L is the number of intensity levels that are allowed in the image. So, for instance you take an rgb image all the values go from 0 to 255. So, you have to physics intensity level now if we use this particular T of r then we can show that p of S follow the algebra p of S p substitute S of S is nothing but 1/ L-1 which is the uniform distribution and this is just a bunch of algebra you can I urge you to walk through it.

But the idea is we are trying to find out a transformation T of r which will give us the uniform distribution in the synthesized image and it turns out this transformation yields that. So, again the idea here is once again to make the pixel of intensity values uniformly distributed in the synthesized image.

So, this is the algorithm or the intent behind histogram equalization of course we can put in you can try different forms of this function but typically what we want to find out is that what you want to estimate is that the synthesized image has a uniform distribution. So, the point is how do you figure out T of r? What is this T of r that will work is the idea that the intensity of an image can be viewed as a random variable.

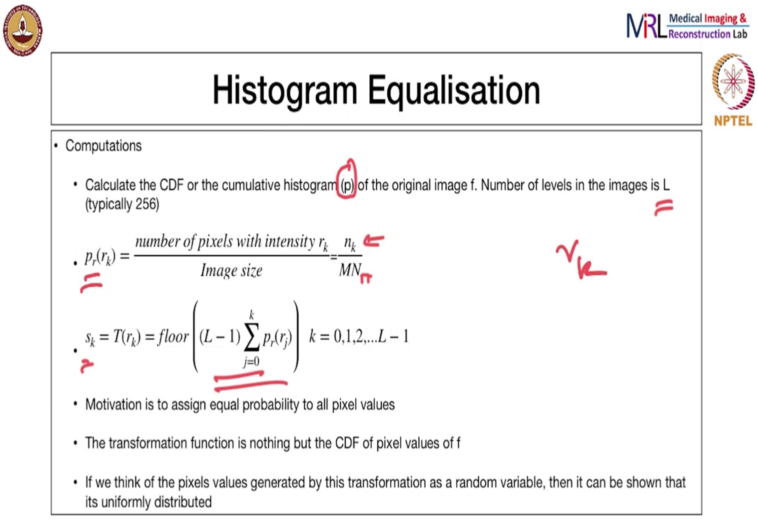
So, which means that we can say that its histogram distribution is nothing but its pdf for the probability distribution. So, then we can talk about probability distributions of the input image and the transformed image. So, $p_s(s)$ denotes the probability distribution or the histogram distribution of the synthesized image and $p_r(r)$ corresponds to the input image and there is probability theory that if r and S are related by the smooth transformation then the probability distributions are related or related to this expression.

So, we this is nothing but the multiple dimension this modulus dr /ds nothing but the jacobian of the transformation. So, what is this transformation that will work? So, generally the most common transformation is this T of r is given by this expression of course in this integral w is nothing but a dummy variable of integration. So, it is basically the input image probability distribution function which is there this integral itself is nothing but the cdf cumulative density function.

Now from here we can actually calculate ds/dr and from there dr/ds and you substitute it in this formula over here and you will be able to see that ps of S is 1/L-1 which corresponds to the uniform distribution. So, which is what we want, we want the uniform distribution for the transformed image.

So, if we let the transformation function to be L-1 a factor multiplied by the cdf of the input image pdf then we get the following output that is basically the output image does not have a uniform distribution.

(Refer Slide Time: 17:01)



So, I will solve this and we will look here. So, how do we actually do this? So, calculate the cdf of the or the cumulative histogram as we call it of the original image. So, the input image is f. So, we calculate the cdf of the cumulative histogram the number of the levels in the image is l. So, we typically have 256 levels and this is typically how you would calculate the probability of a particular level k that is to calculate the number of pixels in that level k divided by the total number of pixels which is nothing but the image size.

And the way you would calculate S k which is basically the synthesized pixel value is $T(r_k)$ which is given by this expression. So, you would do this in a numerical fashion. So, the motivation here like we thought was the acquaintance of the need to assign equal probability to all pixel values and it turns out that the transformation function is nothing but the cdf of the pixel values of f.

So, given any pixel value you can in the source image or the input image we can calculate the target's transformed pixel value. So, if we can of course we just showed that the transformations obtained this way it turns out that S is uniformly distributed and. So, that gives us the needed picture and shows the computations.

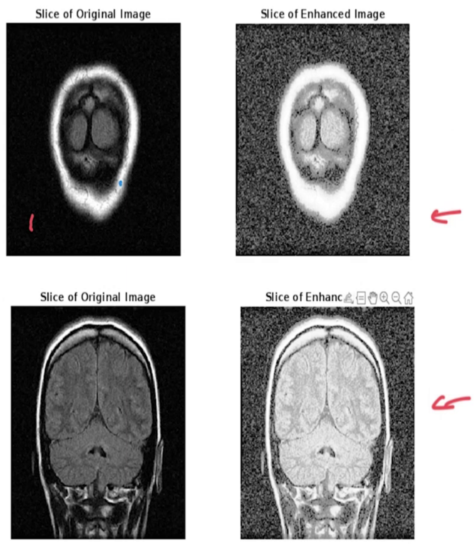
So, you just calculate the cdf or the cumulative histogram of the original image the number of levels in the image is l assuming. And then for every level in the image every r_k where k is the level in the image you can calculate what the corresponding S is and then assign it to that value. So, that is it once you are done with that then you can then you have the new image actually done for a discrete image how is it done.

So, I will wipe out the existing things here. So, that you can see clearly. So, how does it work in the case of a real image? So, for the computations we know that in a lot of cases there is a finite number of levels in the image. So, basically we say L is 256 and the cdf or the cumulative histogram of the original image f and I think I must have used the wrong terminology here. Just use cdf how do we calculate cdf of the f input image f.

So, we can calculate the probability for a pixel to be at a particular level k given by r k the intensity value corresponding to level k is given by r_k and that probability is depend nothing but the total number of pixels with that in that level k divide sorry number of pixels in the level k divided by total number of pixels M times N basically is the size of the image M rows N columns.

So, that is how we have that is a total number of pixels. So, we can always then find out S k as this finite sum. So, the integral is replaced by a summation in the case of a discretized image. So, this process is implemented for histogram equalization.

(Refer Slide Time: 20:25)



So, here are some examples: this is the original, this is an MRI of the brain. So, you can see the image on the left is uniformly dark there are some structures inside that you can see and outside of course the skull but once you do the transformation this is just the histogram equalization you can see that their structures are much more visible there is a much better contrast compared to here.

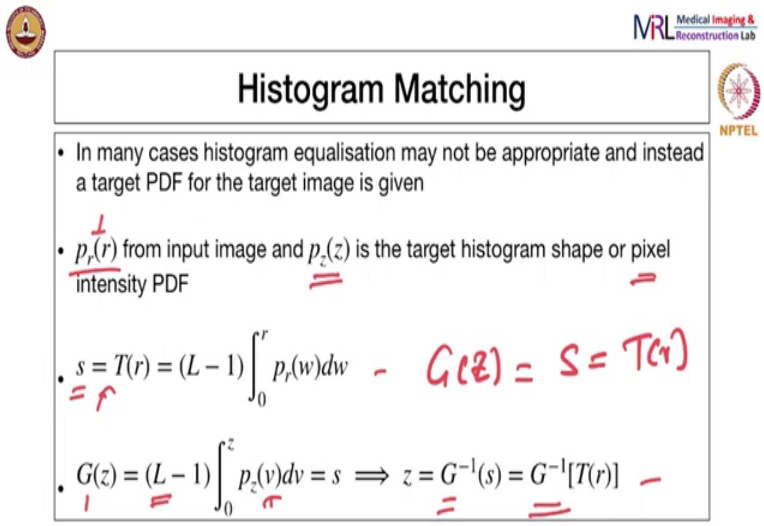
Of course it looks slightly more noisy and that is an issue but still you get better contrast here too. There is some dark see this region is slightly dark this region is slightly brighter but that is much more apparent here along with all the other artifacts which are shown artifacts or structures in the brain that are clearly shown here.

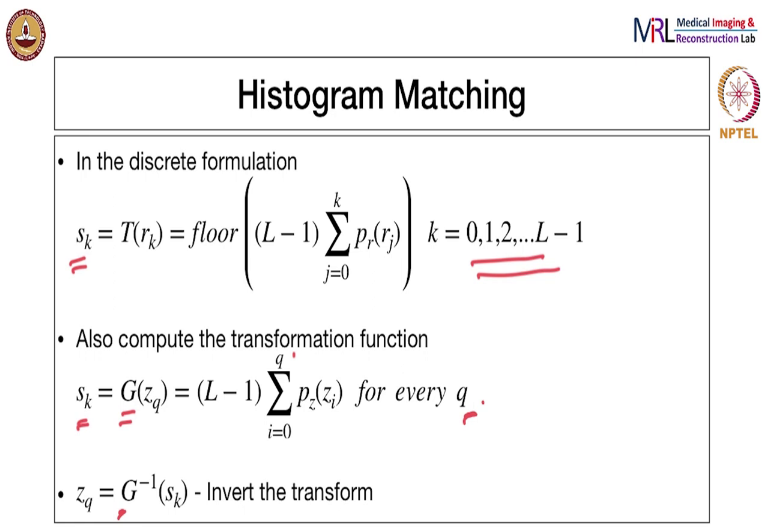
So, once again if you go and that has much better contrast than the others of it. So, for instance in this case this is a slice of an MRI of the brain as you see the top row and the bottom row the left hand picture once again I will remove the markings. So, if you look at the top row and the bottom row you see that the left column is nothing but the raw image and the right column is the histogram equalize image you can see that that some it becomes thumbnails like appear somewhat slightly noisy but you see that while the image on the left both of them are uniformly dark you can see that on the hand side there is a contrast obtained between different regions of the image.

So, you might find it noisy because the background has gotten a little bit noisy like I said because the algorithm does not distinguish between bright and dark regions sorry. So, it does sense what I mean by bright and dark because it does not distinguish between noise and real signal. So, in this case this dark region is just noise because it is just background. There is nothing there, this is just a head in the middle.

So, but then it does not know that, it just spreads pixel values everywhere. So, that is the only issue but other than that you get excellent contrast.

(Refer Slide Time: 22:29)





So, in many cases histogram equalization might not be appropriate or we will have a completely different idea of what it means to modify the histogram. So, for instance let us consider in this current scenario a lot of deep learning techniques are being used for medical image analysis. So, our machine learning techniques are used for it.

But let us consider a situation where your data is drawn from your image data is drawn from multiple scan centers et cetera. So, one preprocessing technique there in order to account for the difference in dynamic range of these pixel values in each of these centers you can do histogram matching.

So, in this case the idea is not to turn everything into a uniformly distributed pixel value image but rather you have a target histogram a target histogram is given which is kind of considered ideal you can and then what you do is you try to match all the image histograms to that target histogram. So, instead of the uniform distribution.

So, in the previous case we saw when we do this histogram equalization the uniform distribution was the target histogram of the target pdf. In this case you might have an ideal image you might actually have done a averaging of a bunch of images which had very good contrast a very good snr et cetera and derived a pdf from there or a histogram distribution from there and you want to match everything to that that is the other scenario where you know this histogram manipulation is done. So, that process is called histogram matching.

So, here we will just clarify notation once again this $p_r(r)$ is the input image distribution again once again we start from the continuous distribution and then we can we can go to the case where in the images the pixels are discrete and have discrete values et cetera and p_z is the specified target histogram which is a target histogram shape if you want to call it that I have used the term histogram distribution might not be correct or commonly used but that is for the case of just for the sake of understanding we use it or pixel intensity pdf is another terminology.

So, the $p_z(z)$ is given $p_r(r)$ can be estimated from the input image. Now what we can do is to do this transformation S which we have seen is T(r) take this and we do the cumulative pdf calculation to get s. So, this is T of r similarly with the $p_z(z)$ which is also here this is the target histogram we do a similar transformation.

So, now what we have to identify is that these two are the same. This should give you the same values provided we have this G(z) since again just to clarify this v is just a dummy variable. So, we can say that G(z)= S =T(r) this is true. So, the z that we are getting then must satisfy the following condition $z = G^{-1}(s)$ correct this is stress from this equation which is $G^{-1}[T(r)]$.

So, we can if we back one more time in the continuous formulation we can estimate p of this this is given this is from the image we can estimate this from the input image. And we can use p z of z we have this is given p is that in this case p subscript z of v is given but v is basically just a dummy variable. So, once you have $p_z(v)$ we can also calculate G of z.

So, T(r) and G(z) can be readily calculated then all you have to do is estimate $G^{-1}(s)$ and So, then we are done correctly. So, for every S value we calculate z which is z which is $G^{-1}(s)$. So, estimating the G inverse is not an easy task. It is a computationally analytically difficult task to do but when you are working with discrete quantities which you will see in the next slide this is not a problem.

So, then what we did like we did in the previous time we convert this continuous result into a discrete form. So, when you say discrete form which means that you work with the histograms of the image empirically determine histograms of the images and the target histogram that is what we have. So, how do we do. So, in the case of. So, we will go on to the next technique which is histogram matching very similar to this only that now we do not make it into a uniform distribution when I say make it into a uniform distribution I mean that we have an input image.

But our target is not to get to a uniform distribution after processing uniform distribution of the pixel intensities after processing. Here we have a target histogram given a target probability distribution of a pixel target pdf. So, go through the algorithm. This is the discrete form of formulation. In the continuous formulation of this discretization we calculate S_k from the input image for all the L - 1 levels in the input image.

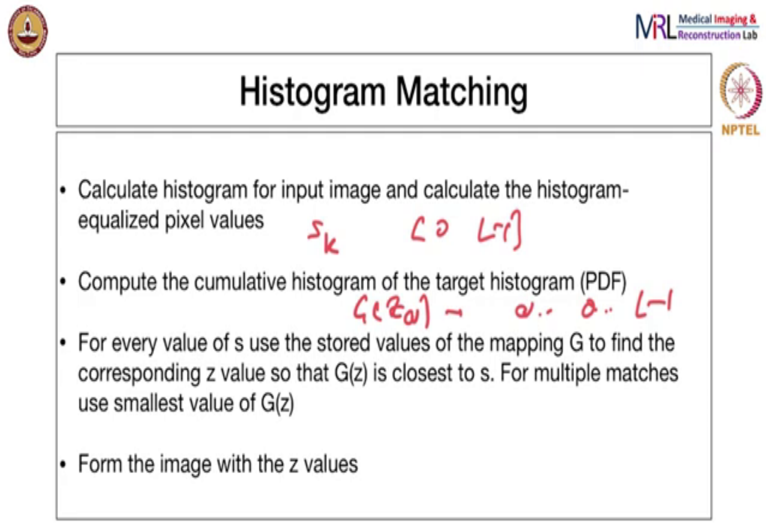
So, once again if the target histogram has q levels we calculate the same for every q and then how do we invert this G inverse how do you do that? That is the tricky part. So, from the how do we get the z of q from the inverse. So, there is no need to calculate this so-called inverse because what we do is basically we are using when we do it with these intensity levels either q L minus 1 intensity levels is that they are discrete and they are integers.

So, we can calculate all possible values of G. So, calculate all possible values of G and these are rounded to the nearest integer value in the range 0 to n minus 1 and you put them in a lookup table and then for a particular value of S_k we will look up the closest value. So, that is all that is all there is to it.

So, for every possible value of z_q you calculate G(z_q) it is not a problem to substitute q for every q level every q we calculate this. So, from z 1 to let us say 100 levels that went to that 100 you can calculate and for a given S_k find out which is the closest and you just can just do a nearest neighbor match. So, this is the typical approach. So, we do not have to invert any function but just use a lookup table.

So, for a discrete if this is the case where you are actually provided the histogram is not a continuous function et cetera but rather a discrete function.

(Refer Slide Time: 30:45)



So, what we do is we calculate the histogram of the input image and the histogram equalized pixel value. So, S we calculate S which we do S k as we want in the range 0 to L-1 we do this. So, we compute the cumulative histogram of the target cumulative histogram of the target histogram G which is basically nothing but G(z_q).

And that goes from where q goes from once again goes from 0 to L-1 for the target histogram we do the same thing. So, we always round the values of G to the nearest l minus 1 nearest 0 to l minus 1. So, the long head values in a lookup table. So, for every value of S_k that we have we find in the closest we find z_ q.

So, that G(z_q) is closest to the S. That is all we do. So, for multiple matches we use the smallest value. So, that is it then your mapping is done because we have every equalized pixel S k is now being mapped to the corresponding pixel with values and z that is all we want. So, we have now mapped from our input to S to z. So, see the intermediate step of equalizing the input image is conceptual.

And because we can we can skip it by doing a cumulative but we won't get into those details but this is the way this is the way to do it. So, given these input values given the input values given the input image you calculate its histogram and calculate its equalized histogram then you do the equalized histogram for the target histogram given and then you map them. So, now that way you are now mapped to z from your input you are now mapped to z.

So, this is what I call the histogram matching step which is used as a pre-processing step in many applications. So, now there are implementations of this available in matlab and other python etc . You are free to use them but internally this is what is happening. So, in the next videos we look at edge detection et cetera but this is all for histogram matching. Thank you.