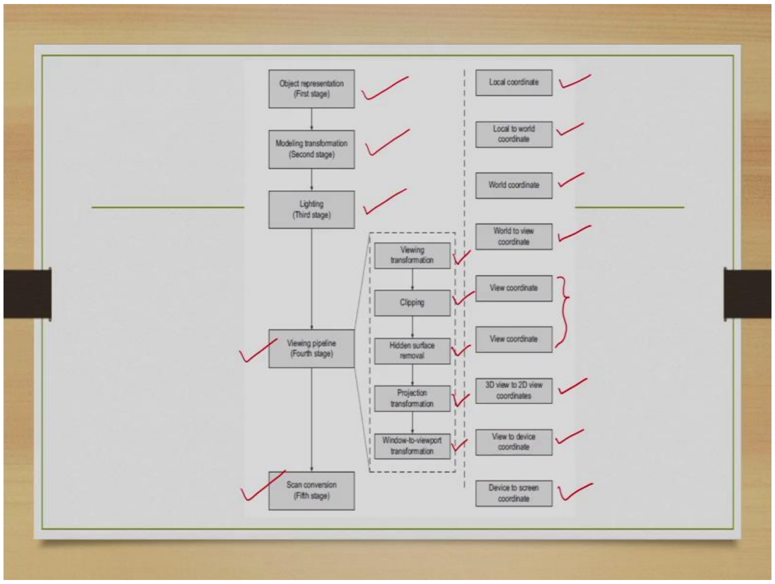
Computer Graphics
Professor Doctor Samit Bhattacharya
Computer Science and Engineering
Indian Institute of Technology Guwahati
Lecture 28
Anti-Aliasing Techniques

Hello and welcome to lecture number 28 in the course Computer Graphics. We are currently in our last leg of discussion on the 3D graphics pipeline. Let us quickly recap the pipeline and then we will continue our discussion today.

(Refer Slide Time: 00:47)



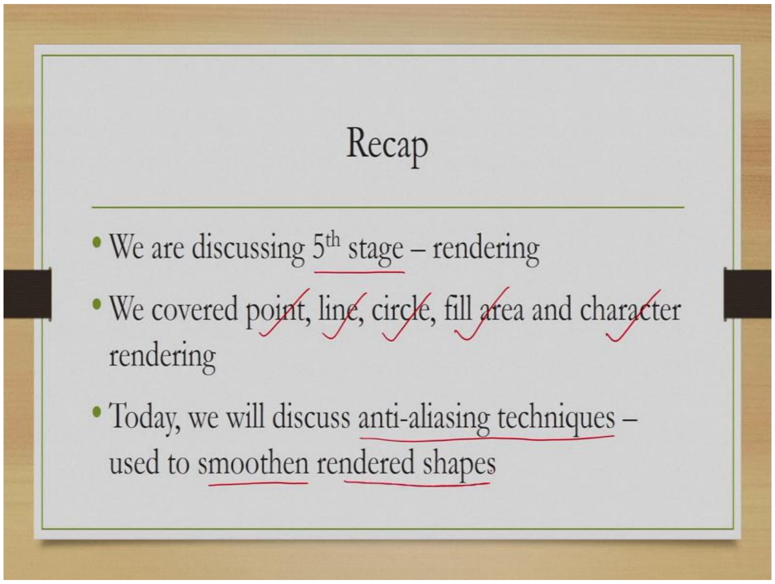
So, as we have learned there are five stages, let us quickly go through the stages once again. First stage is object representation, second stage is modelling transformation, third stage is lighting, four stage is viewing pipeline and there is a fifth stage which is scan conversion and just to recap among these stages object representation deals with representation of objects that constitute a scene and there the objects are defined in a local coordinate system.

Second stage is modelling transformation where we combine different objects to construct a scene and there we perform a transformation that is local to world coordinate transformation and then the scene is defined in world coordinate system. Now, in this world coordinate system we assign colours to the object that is the lighting stage, then in the fourth stage that is viewing pipeline we perform a series of transformations namely view transformation where we transform from world to a view coordinate system then projection transformation where we transform from

3D view coordinate system to 2D view coordinate system and then thirdly window to viewport transformation where we transform from 2D view coordinate system to a device coordinate system.

In this fourth stage we also performed two operations namely clipping and hidden surface removal both of these operations are done in this view coordinate system. The fifth stage that is scan conversion is also related to transformation where we transform this device coordinate description of a scene to a screen coordinate or pixel grid. Currently we are discussing on this fifth stage that is scan conversion.

(Refer Slide Time: 03:01)

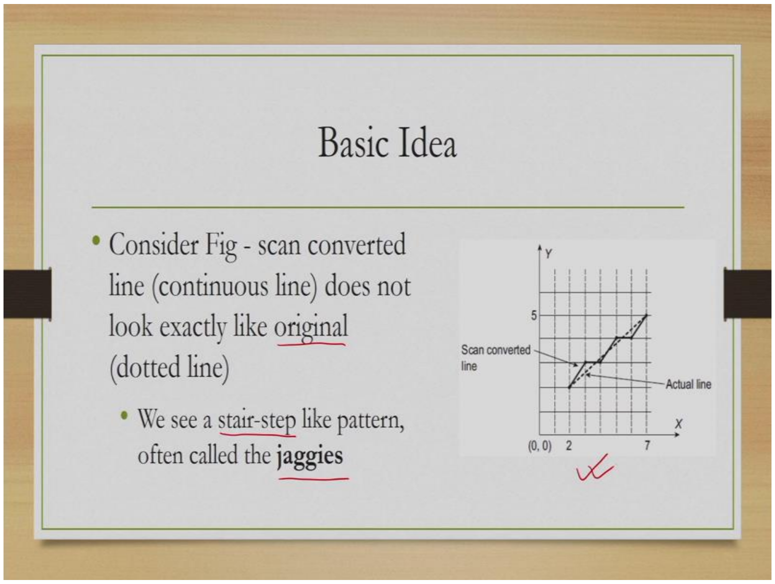


Now, here in this stage we have already covered few topics namely how to convert point, line, circle, fill area, and characters. Today we will discuss an important concept related to scan conversion which is called anti-aliasing techniques. This is required to smoothen the scan converted or rendered shapes.

(Refer Slide Time: 03:39)

Let us, try to understand what is anti-aliasing then we will discuss few anti-aliasing techniques with examples.

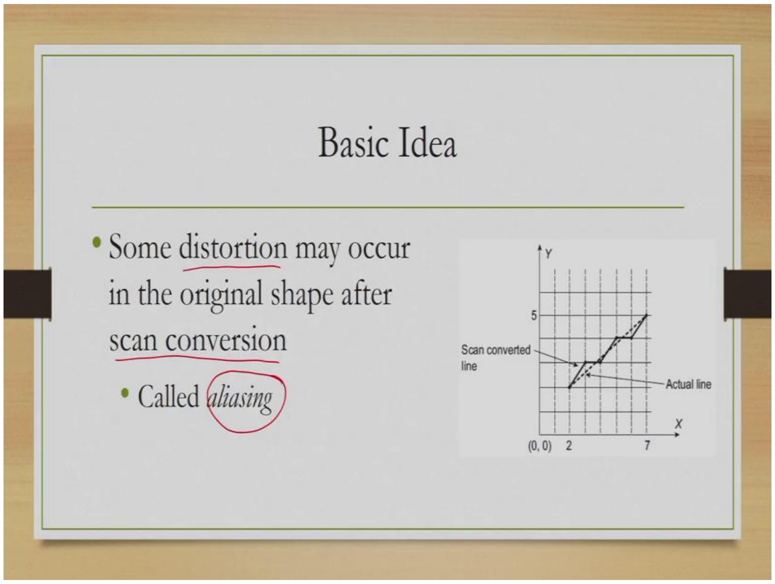
(Refer Slide Time: 03:48)



Let us, start with the basic idea. What is anti-aliasing? Now consider this figure here as you can see the dotted line indicates the original line that we wanted to scan convert and this is the pixel grid shown here and in this pixel grid, we want to scan convert this dotted line. However, as you can see not all the points on the line passes through the pixels so we need to map those points to the nearest pixels as we have seen in our earlier discussion.

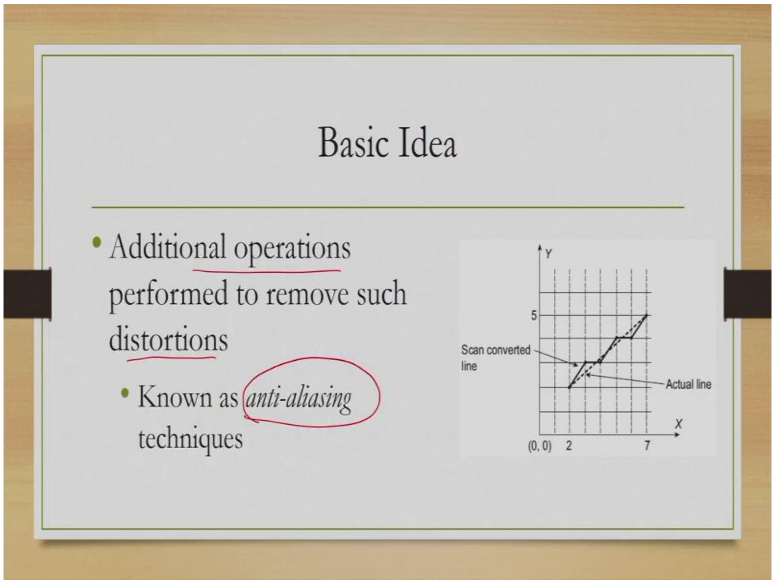
As a result, what we get, we get this scan converted line which looks like a stair step pattern represented with this thick black line. Now, this is definitely not the original line exactly it is an approximation as we have already said however due to this approximation some distortion happens. In case of line these are called jaggies or stair-step like patterns.

(Refer Slide Time: 05:13)



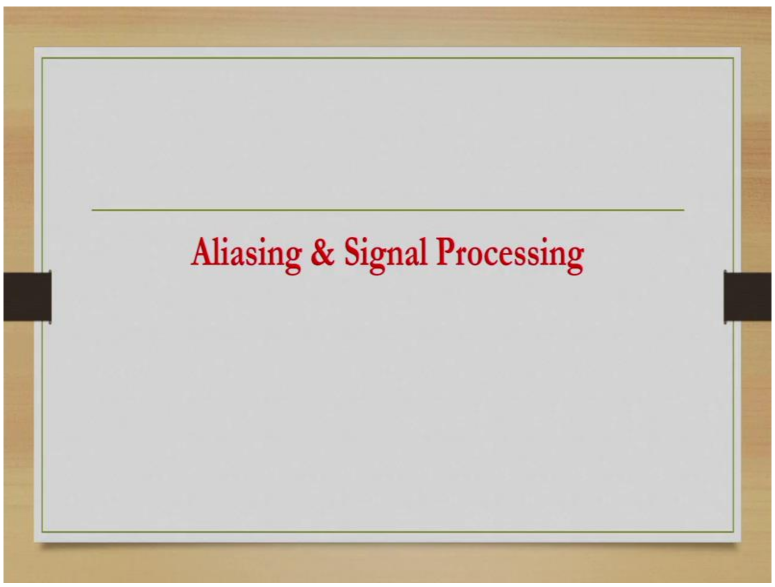
In general, there will be distortion to the original shape after scan conversion for any shapes not only lines. Now, these distortions are called or the distorted shapes that is there after scan conversion is called aliasing this phenomenon where we get distortion due to the algorithms that we follow for scan conversion. Now, why it is called aliasing? What is the significance of this term?

(Refer Slide Time: 05:52)



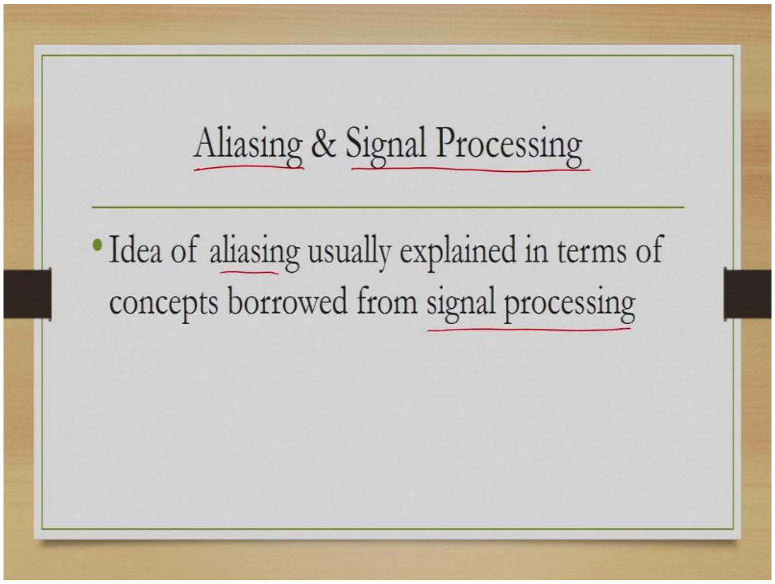
Before going into that we should note that aliasing is an undesirable side effect of scan conversion we want to avoid that to the extent possible. So, additional operations are performed to remove such distortions these techniques together are known as anti-aliasing techniques. So, when we perform some techniques to remove aliasing effects then we call those techniques as anti-aliasing techniques.

(Refer Slide Time: 06:33)



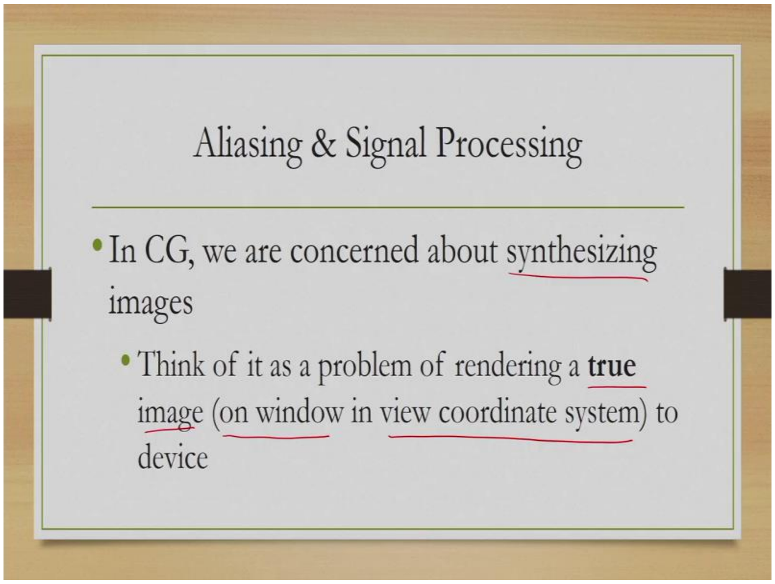
So, why we call it aliasing? What is the significance of this term?

(Refer Slide Time: 06:41)



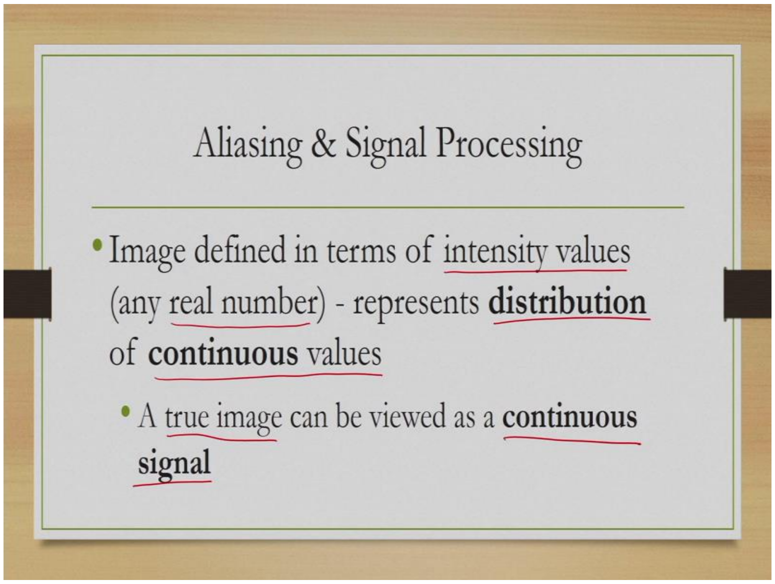
In fact, the term aliasing is related to signal processing we can explain the idea in terms of concepts that are borrowed from signal processing domain.

(Refer Slide Time: 07:05)



In computer graphics what we want we want to synthesize images. In other words, we want to render true image and here we have to think of it as rendering the image on the window in the view coordinate system or the 2D view coordinate system.

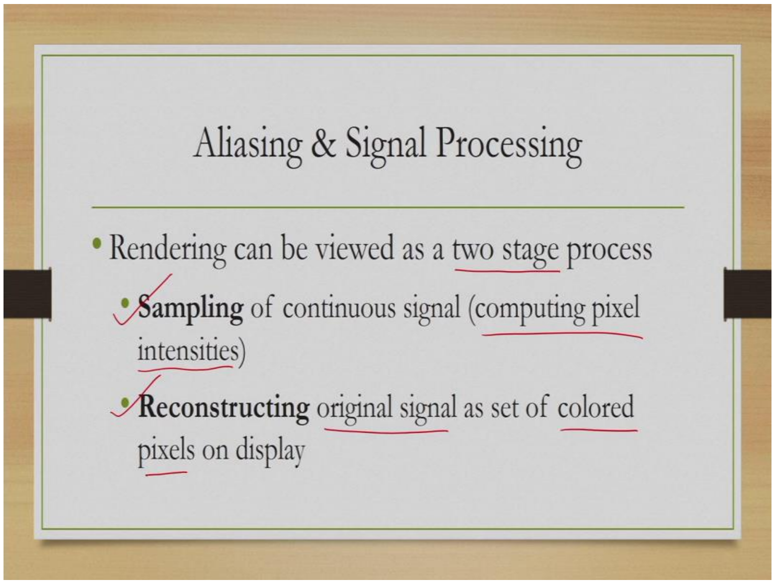
(Refer Slide Time: 07:41)



Now, how we define this image in terms of intensity values, which can be any real number. Remember that we are still in the view coordinate system where we are free to use any real number to represent intensity. Now, those intensity values if we think in a different way, they represent some distribution of continuous values.

So, if we plot that plot those values in a graph then we will get some curve which represents the distribution and here the values are continuous, it can take any real value real number. So, essentially we can think of a true image as a continuous signal that is mapping the idea of image rendering to signal representation.

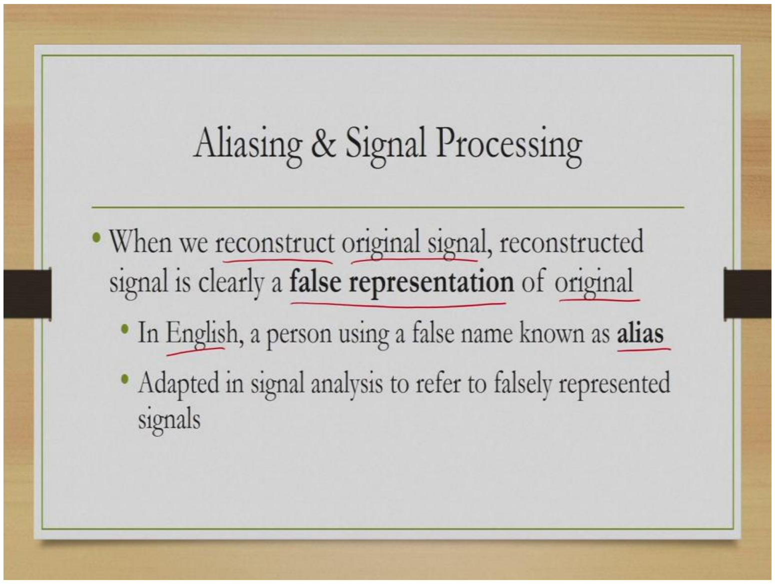
(Refer Slide Time: 08:44)



Now, when we perform rendering what we do? The process of rendering can be viewed as it two-stage process in a very broad sense in the first stage what we do we sample the signal or the intensity values. In other words, we sample those values for pixel locations. So, we try to get the pixel intensities, this is what we have discussed so far how to obtain the pixel intensities from the actual intensities.

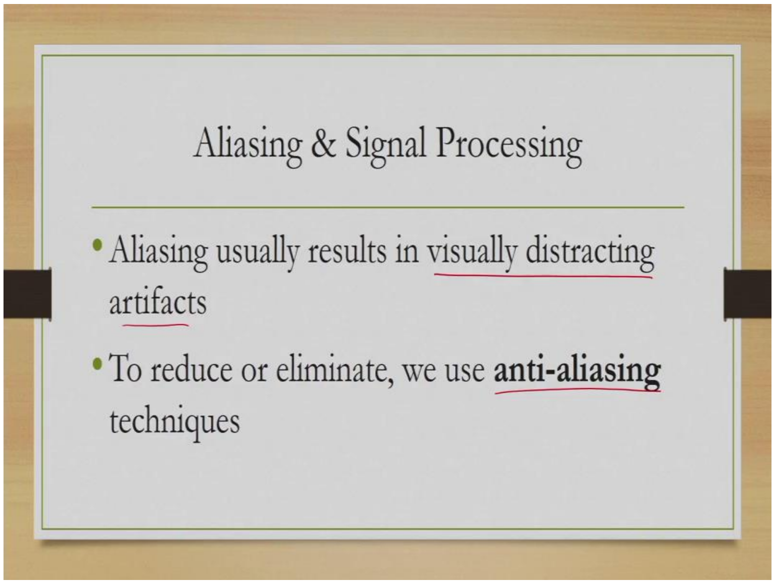
But there is also a second stage that is from the sampled intensities we want to reconstruct the original signal as a set of coloured pixels on the display. So, essentially we are given an image which is a continuous signal of intensity values. We can think of it in that way, then we want to render this image on the pixel grid that is the purpose of scan conversion, how we do that? We follow two stages broadly. In the first stage, we sample the pixel values and in the second stage we reconstruct from those samples to get the rendered image.

(Refer Slide Time: 10:20)



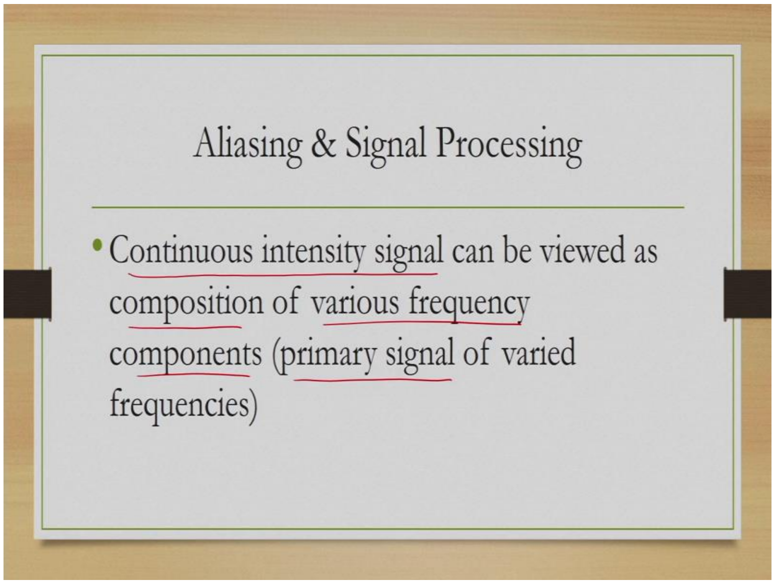
Since we have reconstructing the original signal from the sampled value, clearly it is not the exact signal and we are dealing with a false representation of the original signal. Now, in English a person using a false name is known as alias and the same idea is adapted here where we are trying to represent an original signal in terms of false reconstructed signal hence this particular signal is called alias and whatever we get is known as aliasing.

(Refer Slide Time: 11:19)



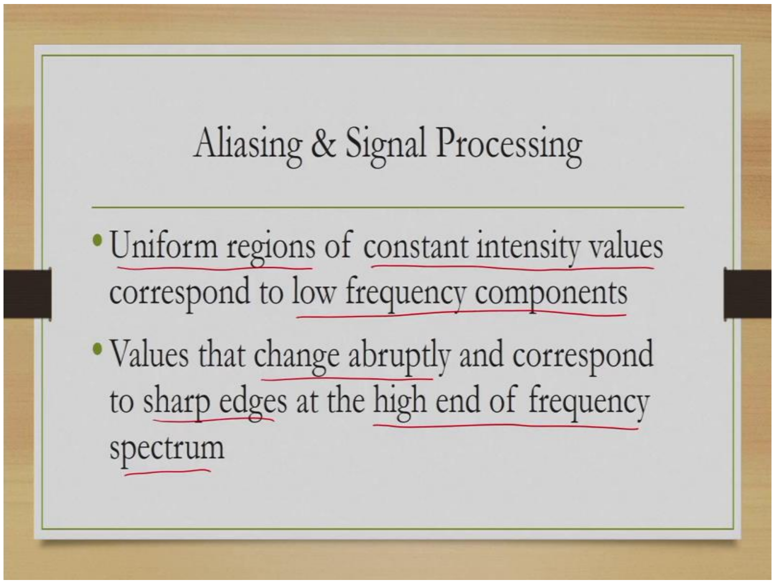
Now, since we are reconstructing there will be some change from the original. Usually it results in visually distracting images or artefacts and to reduce or eliminate this effect we use anti-aliasing techniques. We use techniques that reduce aliasing effects, they are called anti-aliasing techniques.

(Refer Slide Time: 11:48)



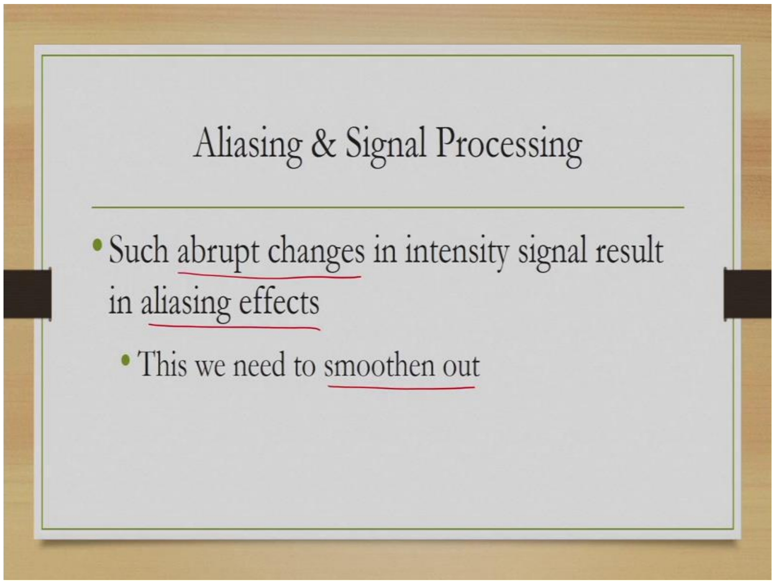
Now, how we do that? Again we can borrow terms from signal processing. So, continuous intensity signal that is the true image can be viewed as a composition of various frequency components or in other words primary signals of varied frequencies. That is one way of giving the intensity values.

(Refer Slide Time: 12:21)



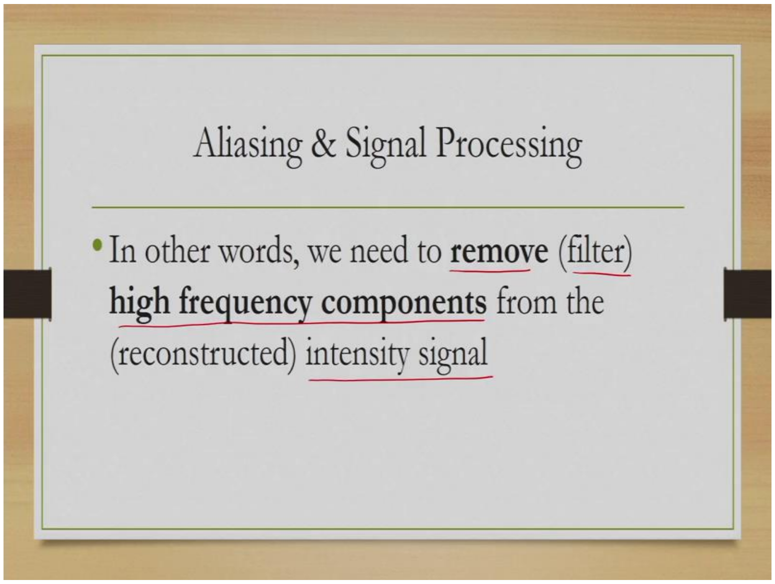
Now, there are two components in those signals. Uniform regions with constant intensity values may be viewed as corresponding to low frequency components whereas there are values that change abruptly and these values correspond to sharp edges at the high end of frequency spectrum. In other words, wherever there are abrupt changes in values we can think of those regions as representing the high frequency components.

(Refer Slide Time: 13:10)



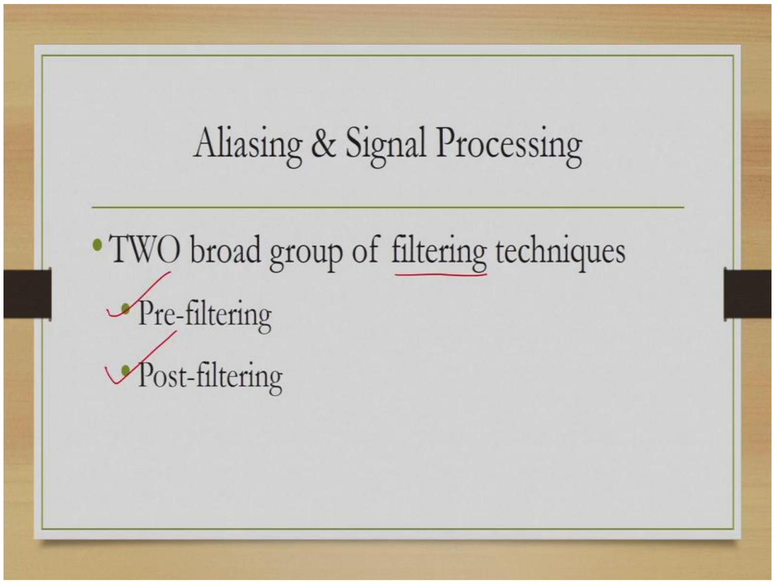
Now, because of those high frequency components such abrupt changes we get aliasing effects. So, we need to smoothen out those aliasing effects, and how do we do that?

(Refer Slide Time: 13:30)



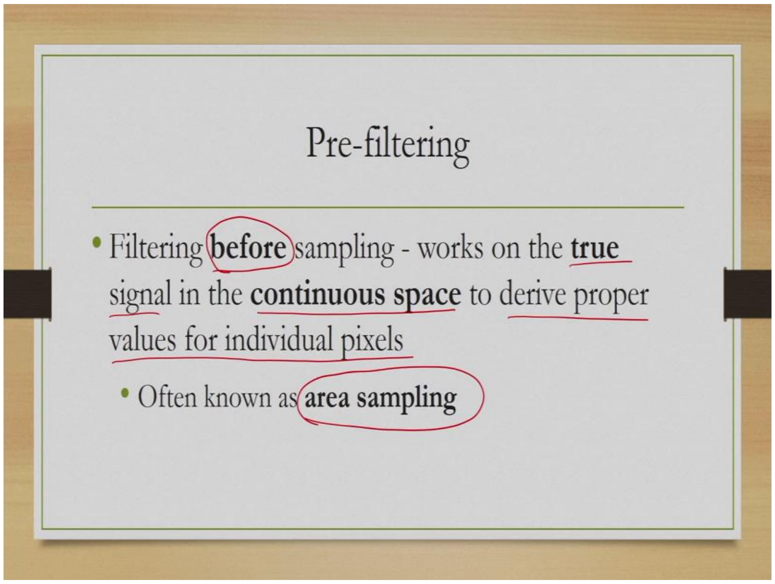
By removing or filtering out the high frequency components from the reconstructed intensity signals. So, if we do not do any additional operations, we will have the reconstructed signal having both high and low intensity which will result in distortion. So, our objective would be to eliminate or remove, filter out the high frequency components in the reconstructed signal.

(Refer Slide Time: 14:12)



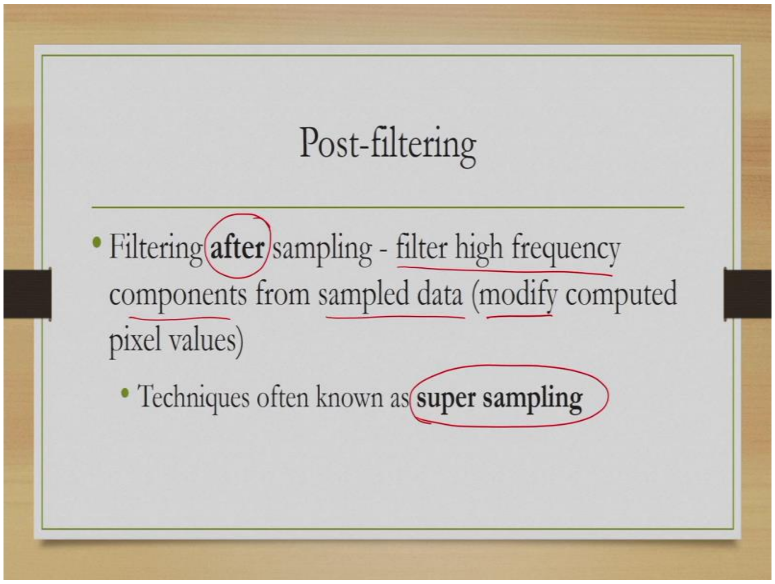
Now, there are broadly two ways to do that. We apply broadly to group of techniques filtering techniques, one set of techniques comes under pre-filtering techniques and the other set comes under post-filtering techniques.

(Refer Slide Time: 14:34)



In pre-filtering, what we do is we perform filtering before sampling that means we work on the true signal which is in the continuous space and what we do we try to derive proper values for individual pixels on the true signal and these group of techniques are also called area sampling techniques.

(Refer Slide Time: 15:07)

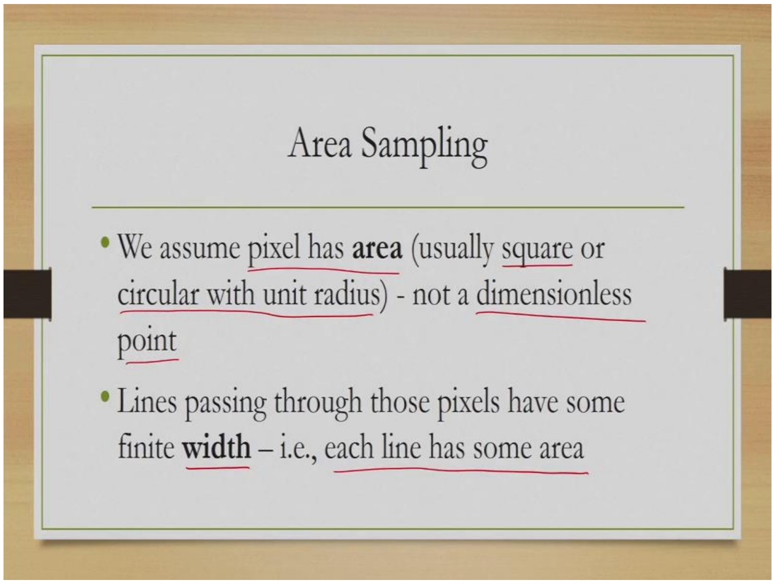


In contrast, in post filtering as the name suggests, we perform filtering after sampling. So, we filter high frequency components from the sampled data. In other words, we compute pixel values and then using post filtering techniques we modify those values. Now, this group of techniques are also known as super sampling techniques. So, we have pre-filtering or area sampling techniques and post filtering or super sampling techniques. Now, let us try to learn about few of these techniques.

(Refer Slide Time: 15:53)

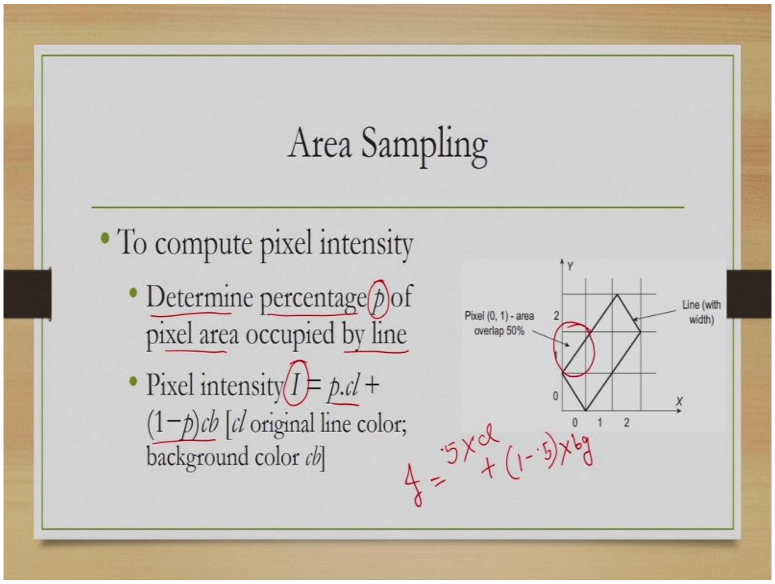
We will start with area sampling. What are the techniques and how they work?

(Refer Slide Time: 16:01)



Now, in case of area sampling, we assume that pixel has area. So, pixel is not a dimensionless point instead it has got its area usually considered to be square or circular with unit radius and lines passing through those pixels have some finite width. So, each line also has got some area.

(Refer Slide Time: 16:35)



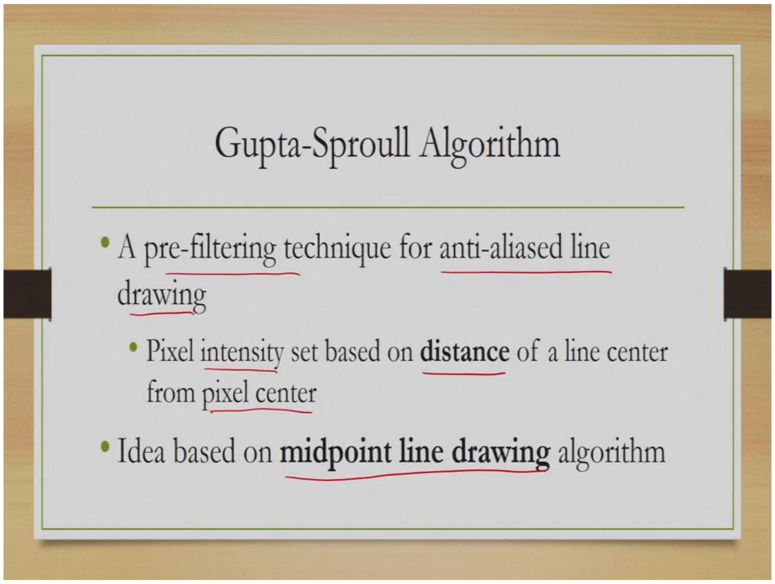
Now, to compute pixel intensity what we do we determine the percentage of pixel area occupied by line. Let us denote it by p then pixel intensity I is computed as a weighted sum of the line colour and background colour as shown in this expression. So, to compute intensity of a pixel say for example here this is pixel 01.

Now, here you can see that the line covers 50 percent of the pixel area or 0.5 into the line colour whatever that is, that cl then 1 minus 0.5 into whatever is the background colour that will be this pixel colour of this particular pixel 01 in the figure. Note that earlier what we did earlier we simply assigned the line colour to this particular pixel, but here we are considering area, how much of that area is occupied by the line and then accordingly we are changing the colour. This is one approach.

(Refer Slide Time: 18:16)

Let us, have a look at a little bit more sophisticated approach involving more computations, which is called Gupta-Sproull algorithm.

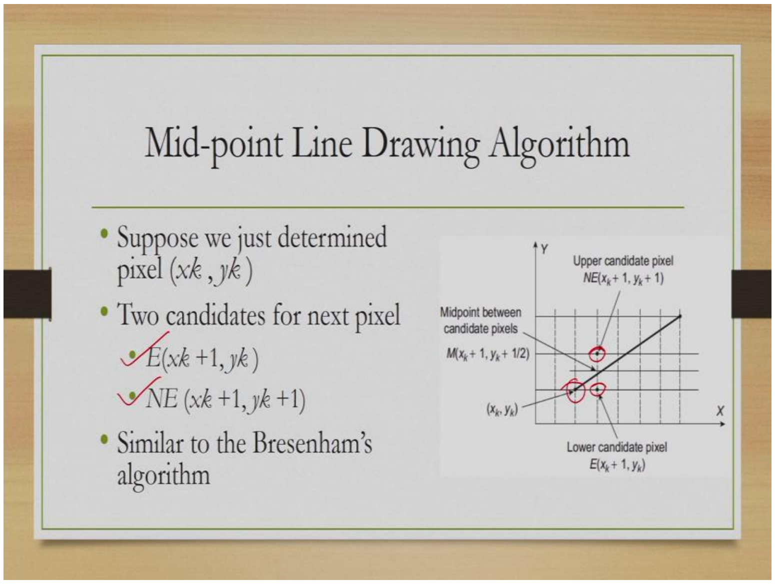
(Refer Slide Time: 18:27)



Now, this is a pre filtering technique used for anti-aliased line drawing and here the intensity, pixel intensity is set based on distance of a line or line centre. Since here we are assuming line as finite width from the pixel centre. Now, this idea of this particular algorithm is based on midpoint line drawing algorithm.

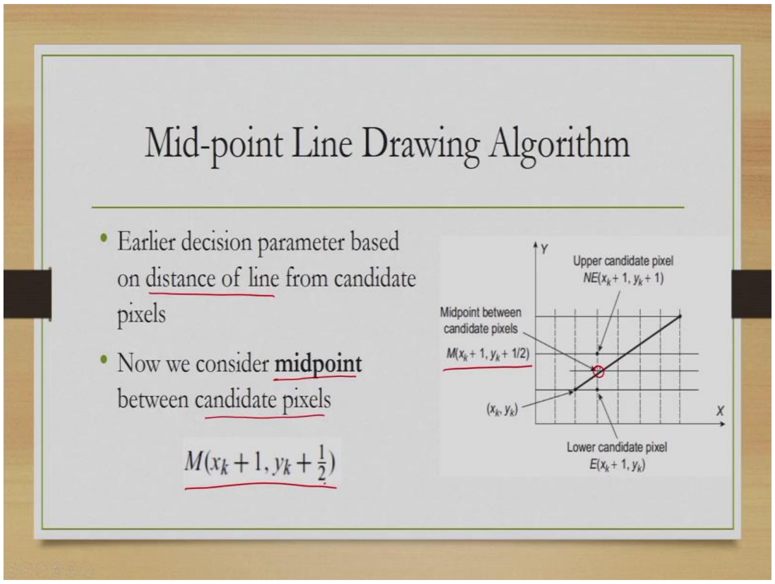
So, earlier we have talked about DDA algorithm, Bresenham's line drawing algorithm. Now, there is another algorithm that is midpoint line drawing algorithm, this algorithm is used in the Gupta-Sproull pre-filtering technique. So, we will first try to understand this algorithm then we will talk about the actual pre filtering technique.

(Refer Slide Time: 19:30)



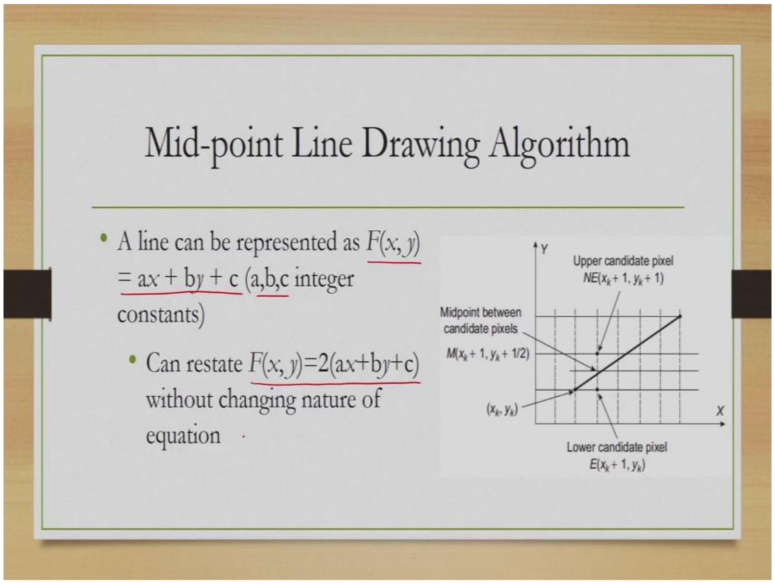
So in the midpoint algorithm, what we do? Let us assume that we just determined this current pixel here and for the next pixel we have two candidate pixels. One is upper candidate pixel, one is lower candidate pixel given by these two E and NE. Up to this, it is similar to the Bresenham's line drawing algorithm that we have seen earlier. Now, what changes is that the way we decide on these candidate pixels, which one to choose.

(Refer Slide Time: 20:15)



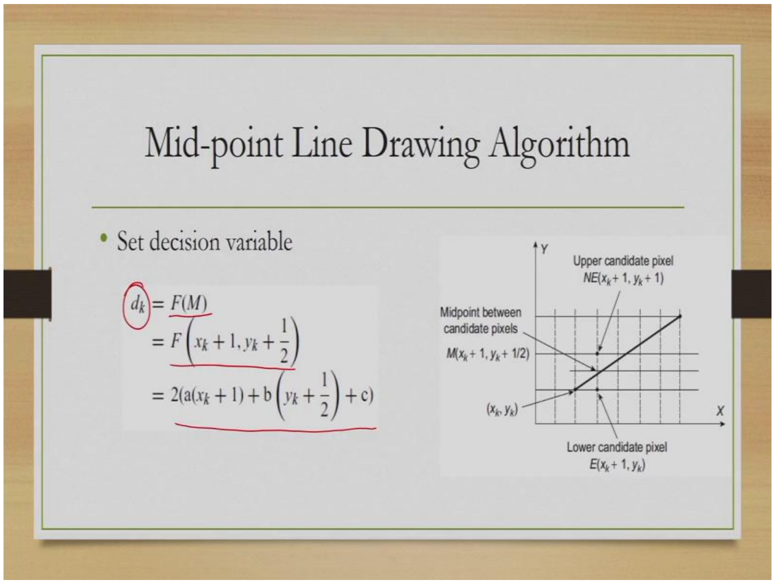
Earlier what we did we made the decision based on distance of line from candidate pixels. Now, in this midpoint algorithm, what we will do is we will consider midpoint between the candidate pixel rather than distance from line. So, midpoint is shown here between these two candidate pixels, which can be represented by this expression as you can see.

(Refer Slide Time: 20:53)



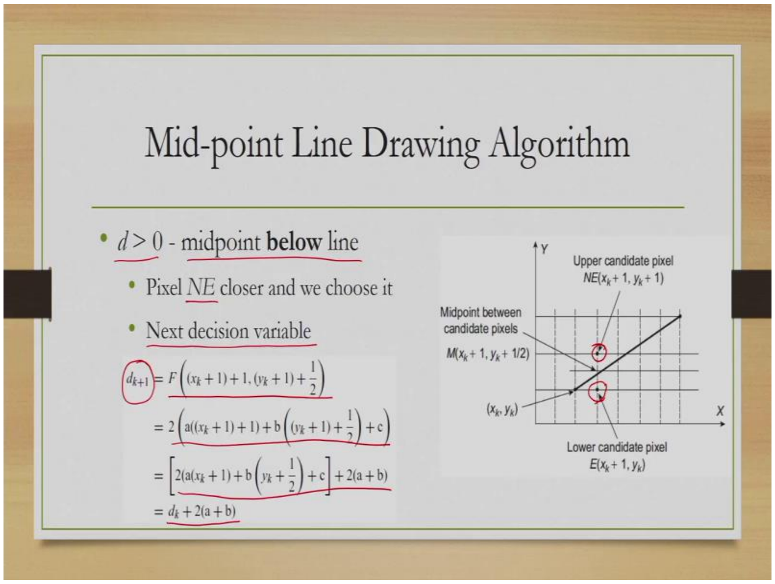
Now, we can represent a line as shown here with this expression where a, b, c are integer constants. Now we can restate this equation by multiplying 2 and get this expression without affecting anything in the equation. So that is just a trick.

(Refer Slide Time: 21:28)



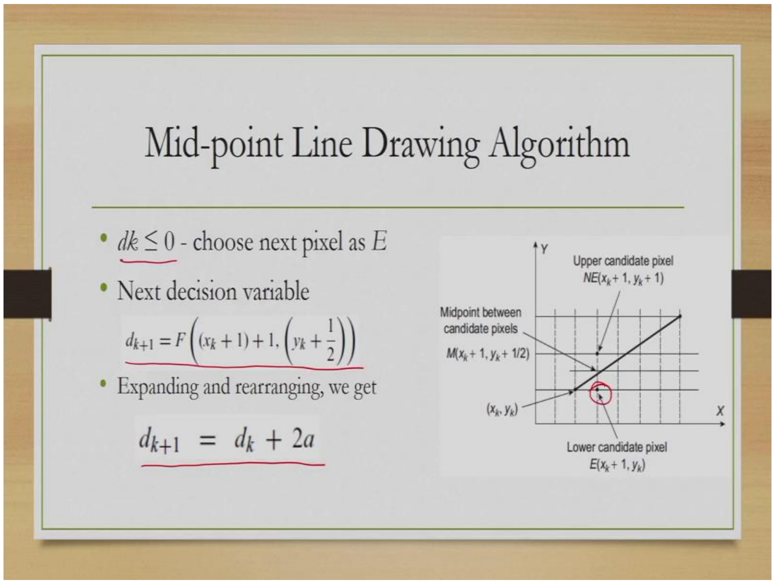
Then we set the decision variable d_k which is the function evaluated at the midpoint M or at this point (x_k+1, y_k + ½). Now if we use the modified equation after multiplication by 2 then we can see that if we expand it will look something like this that is the decision variable at k.

(Refer Slide Time: 22:00)



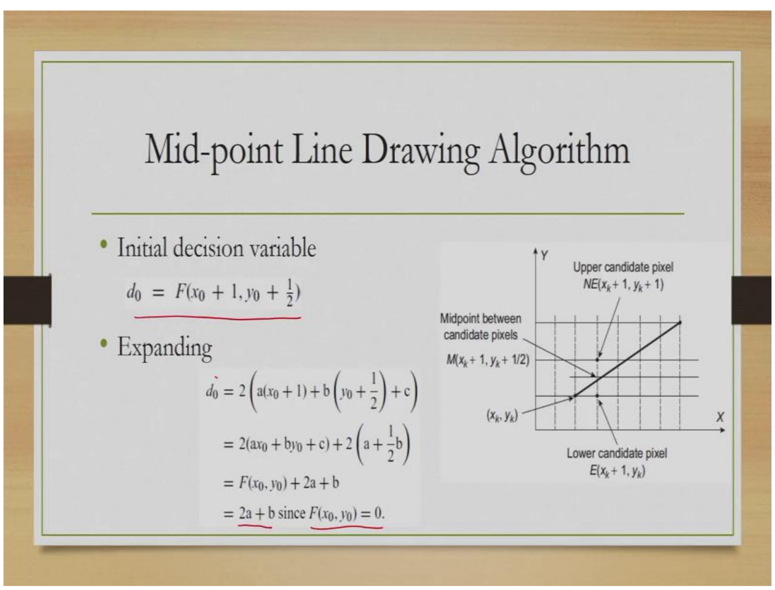
Now, if d is greater than 0 then midpoint is below the line. In other words, if midpoint is below the line then pixel NE is closer and we should choose NE here. Otherwise, we should choose E. Now when NE closer that means d is greater than 0. The next decision variable will be d_k+1 which is the next midpoint and if we expand and rearrange then we get in terms of the previous decision variable d_k. So, d_{k+1} we can represent in terms of d_k and a constant twice a plus b.

(Refer Slide Time: 23:02)



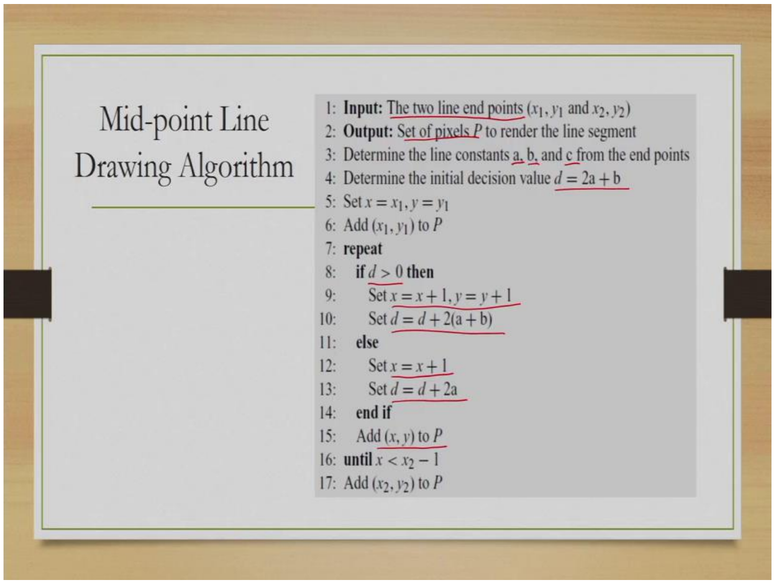
Now, when $d_k \leq 0$, then we choose this one because midpoint is closer to NE and in that case the next decision parameter or decision variable would be given by this expression and which if we rearrange and reorganize then we will get $d_{k+1} = d_k + 2a$.

(Refer Slide Time: 23:41)



Now, what is the initial decision variable, that is given by this expression. And after expanding we get this initial variable to be 2a+b, knowing that this value is 0 as you can see here in this derivation.

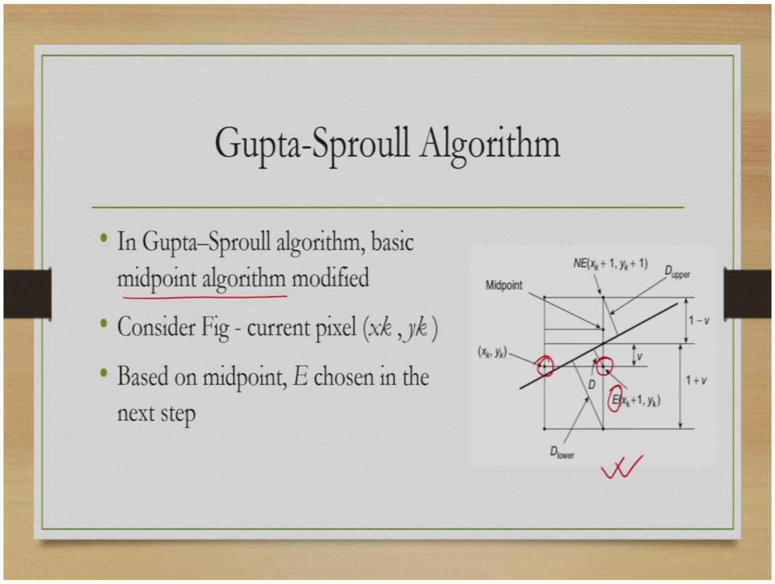
(Refer Slide Time: 24:04)



So, here we have summarized the steps of the algorithm. So, input is two line endpoints and the output is set of pixels to render the line. Now, first task is to find out the value of these constants say a, b and c from endpoints and then initial decision value d then we start with one end point and continue till the other endpoint as before. If d>0 then we update x, y in this way and update the decision parameter in this way.

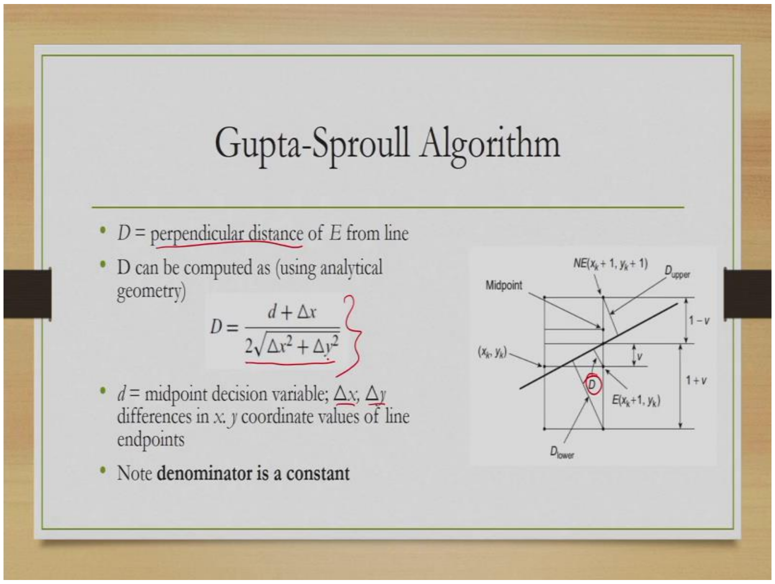
Otherwise, we update x in this way update the decision parameter in this way and eventually in each iteration we add this pixel to P we continue till we reach the other end point that is the midpoint algorithm. Now, let us see how this midpoint algorithm is used in Gupta Sproull algorithm.

(Refer Slide Time: 25:15)



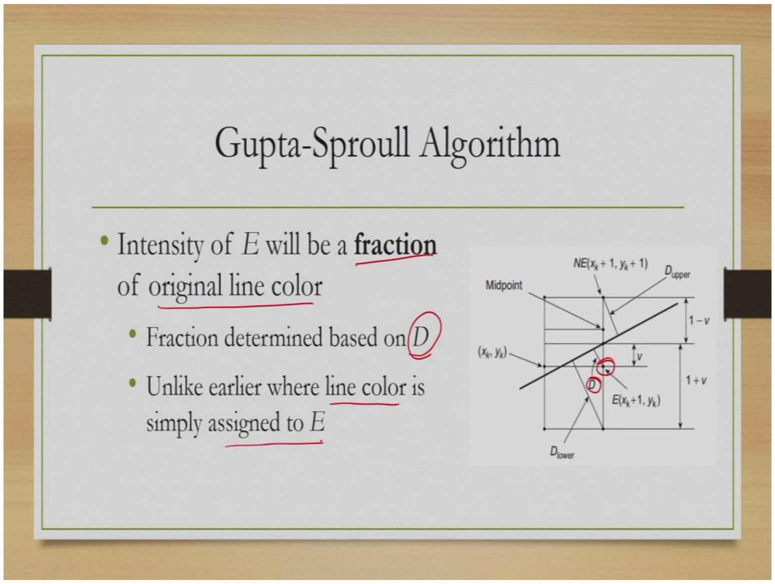
Now in Gupta Sproull algorithm there is some modification to the basic midpoint algorithm. Consider this figure here, here suppose we have chosen this pixel at present x_k, y_k and based on midpoint let us assume that we have chosen E this pixel in the next step. Later on, we will see what will happen if we choose NE instead of E.

(Refer Slide Time: 25:53)



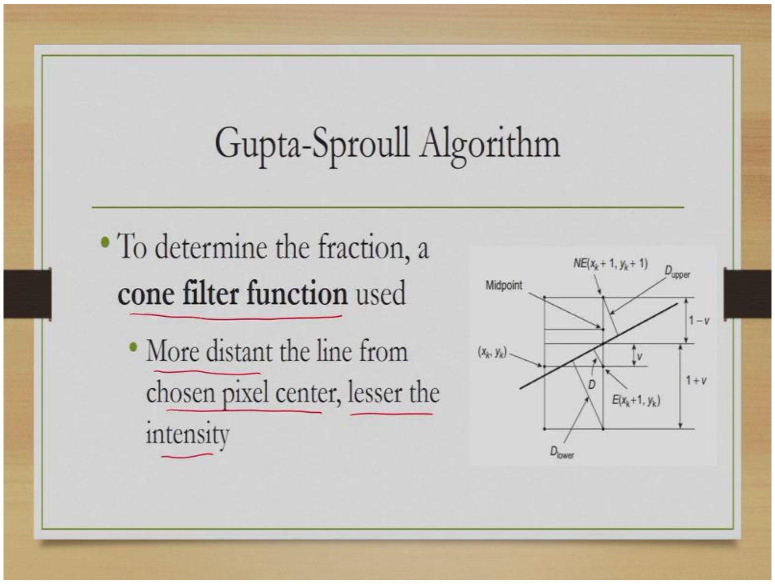
Now, D is perpendicular distance from the point to the line and we can compute D using geometry as shown here where Δ_x, Δ_y are the differences in x and y coordinates of the line endpoints so they are constants essentially. So, the denominator here is constant.

(Refer Slide Time: 26:35)



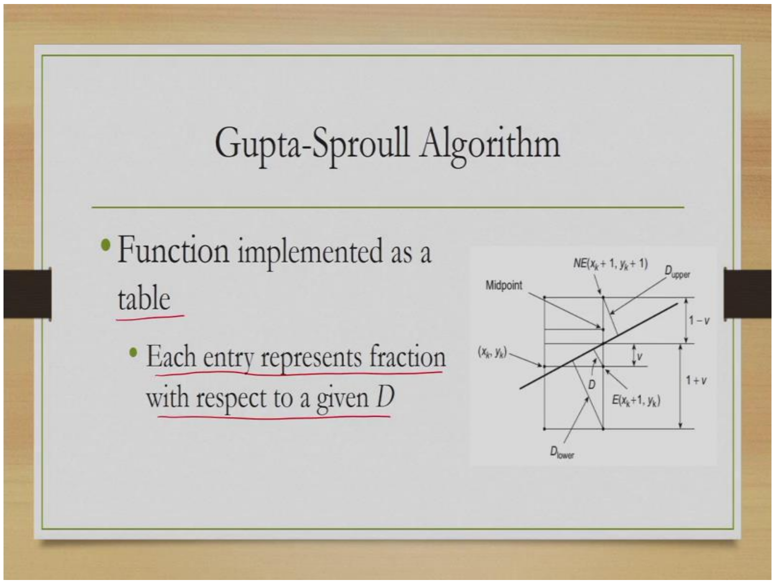
Now, what should be the value of the intensity here, it will be a fraction of the original line colour. So, will not assign the original line colour here to avoid aliasing effect instead we will assign a fraction of the original line colour. So, how to choose this fraction it is based on that distance D. Now see that this is in contrast to the earlier approaches like Bresenham's algorithm or DDA algorithm where line colour is simply assigned to the chosen pixel.

(Refer Slide Time: 27:14)



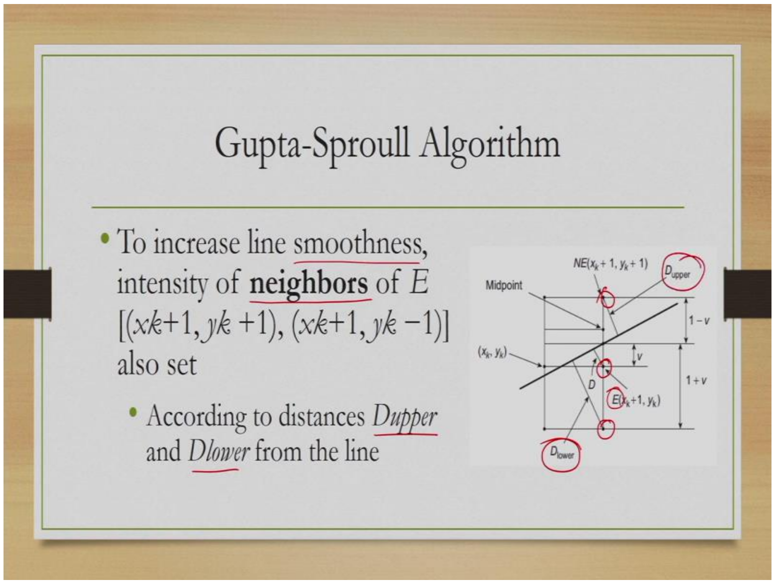
Now, how this distance determine the colour? Typically, a cone filter function is used that means more distant the line from the chosen pixel centre is the laser is the intensity. So, more distant the line is from the chosen pixel centre the less will be the intensity it will be dimmer.

(Refer Slide Time: 27:39)



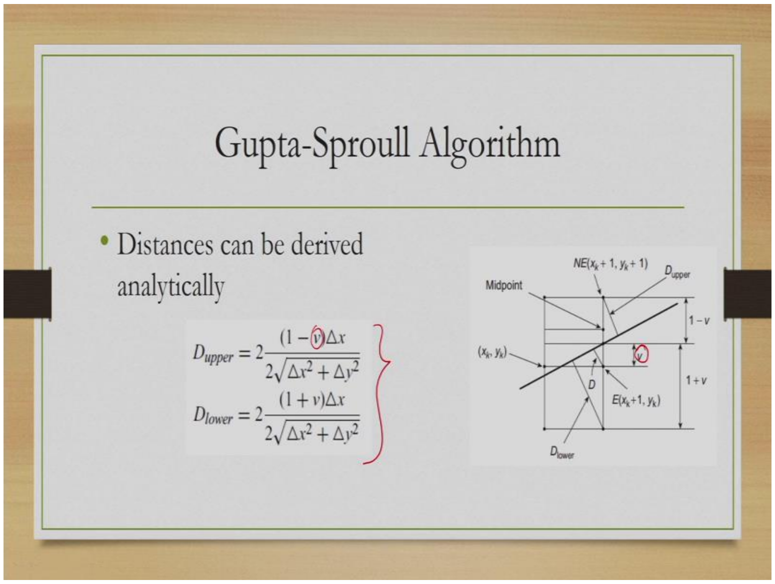
And this distance to intensity value determination is implemented as a table. So, we maintain a table where based on distance a particular intensity value is there and depending on the computer distance we simply take the intensity value from the table and apply it to the chosen pixel. So, each entry in the table represents fraction with respect to a given D. So, some precomputed D values and their corresponding intensity values are there in that table.

(Refer Slide Time: 28:22)



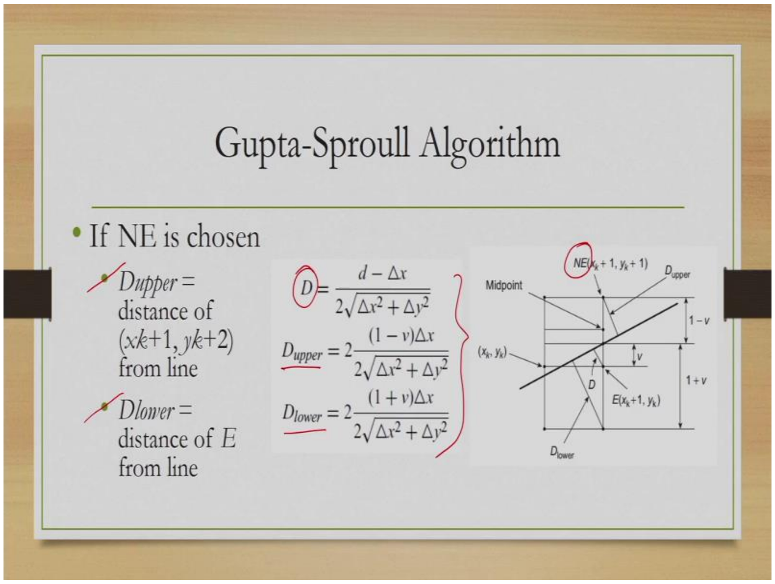
That is not all, along with that to increase the smoothness of the line, intensity of neighbours are also changed. So, here E is the chosen pixel here its neighbours are this pixel and this pixel. Now, their intensity is also modified. Again, according to the distances D_{upper} and D_{lower} from the line. So, here as you can see, this is D_{lower} and this is D_{upper} and depending on these values this neighbour pixel values are set.

(Refer Slide Time: 29:02)



This D_{upper} and D_{lower} can be again obtained using geometry as shown in these expressions where v is this distance and Δ_x Δ_y are as before difference between the x and y coordinates of the endpoints.

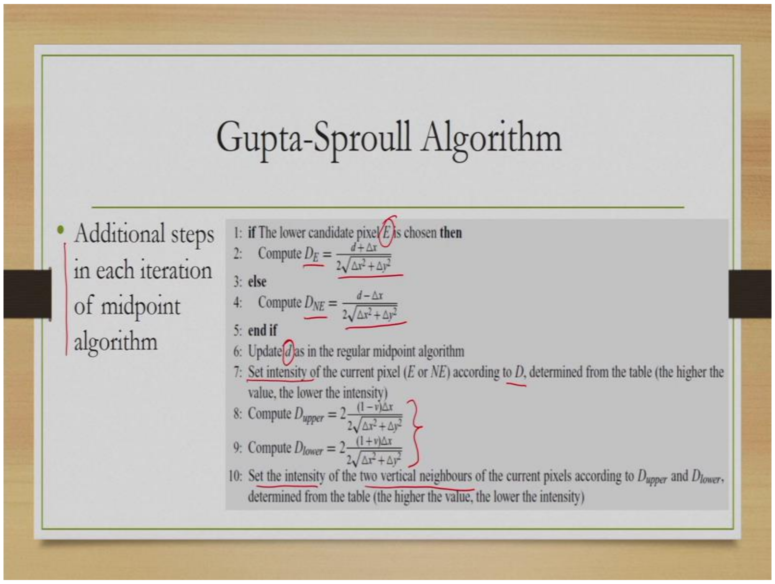
(Refer Slide Time: 29:33)



And depending on those distances again tables are maintained to set the values of the neighbouring pixels. Now, in case of E suppose we have chosen NE, then of course these distances will be computed differently. Again, we can use geometry to check that the distances will be represented in this way. Of course here D_{upper} is different and D_{lower} is different as compared to the previous case and their distances we can represent using these expressions.

So, this is the distance of the chosen pixel from the line, perpendicular distance then this is D_{upper} this is D_{lower} for each we maintain tables to implement the cone filter functions. In the table just to recap we maintain distances and the fraction of line colour to be applied based on that we choose the colour.

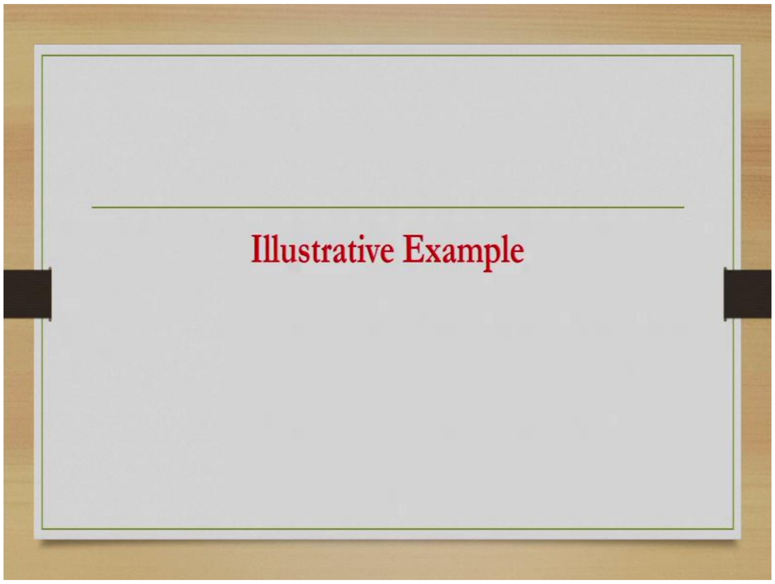
(Refer Slide Time: 30:43)



So, there are a few additional steps performed in each iteration of the midpoint algorithm. In the regular algorithm we do not modify the pixel colours, whatever line colour is there we choose that to be the chosen pixel colour but here in Gupta-Sproull algorithm those steps are modified. So, after choosing a candidate pixel, we determine the line colour by first computing the distance if E is chosen then distance is given in this expression otherwise if NE is chosen then separate expression.

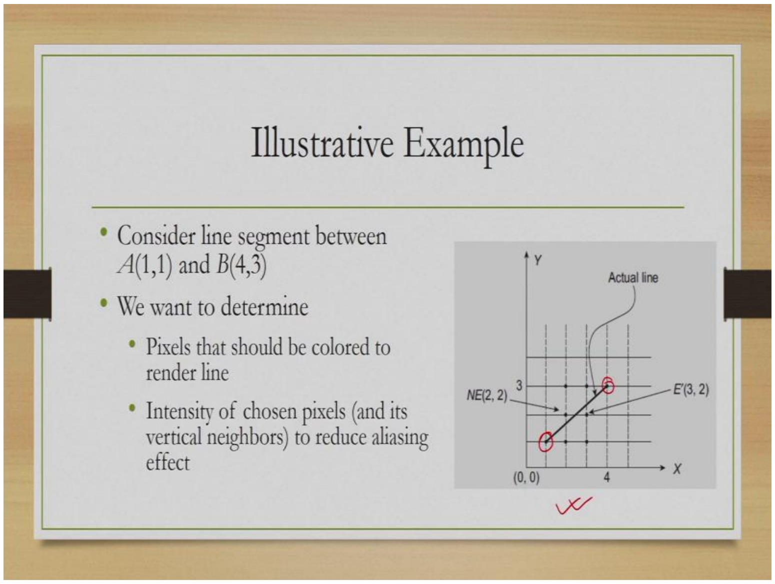
Then we update the decision value as in the regular algorithm, then we set the intensity values according to D, then we compute D_{upper} and D_{lower} and then set intensity of the two vertical neighbours, these are the additional steps that are performed in the Gupta-Sproull algorithm as compared to the original midpoint line drawing algorithm.

(Refer Slide Time: 32:00)



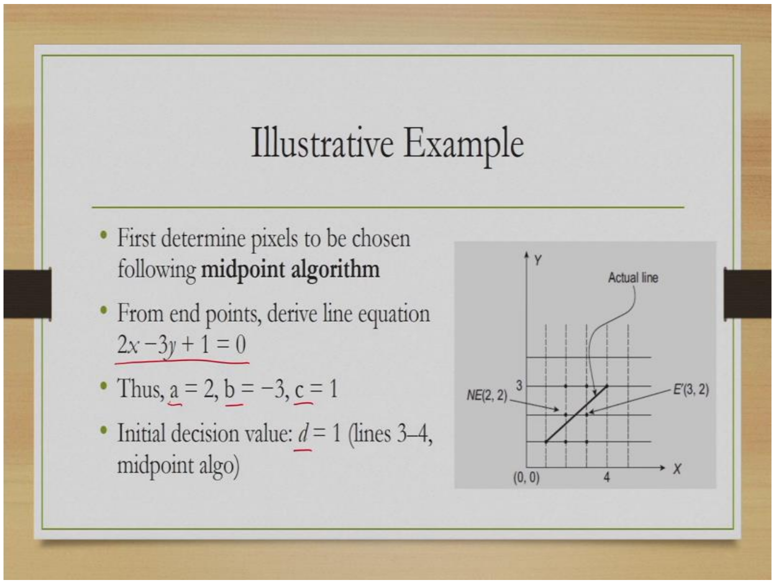
Let us, try to understand the algorithm with one example.

(Refer Slide Time: 32:06)



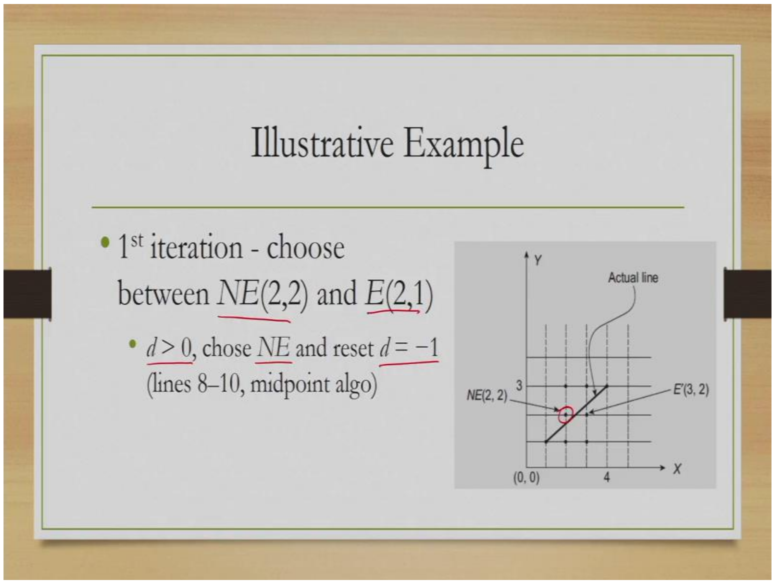
Suppose, this is our line shown here, these are the endpoints and we want to choose the pixels and colour them. So, both the things we need to do, choose pixels and also to choose appropriate intensity values for those pixels.

(Refer Slide Time: 32:35)



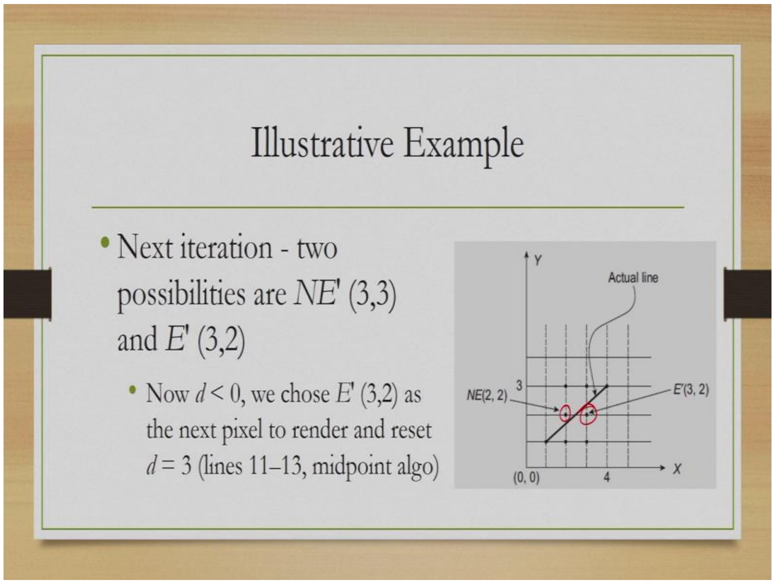
So, first we would go for choosing the pixel and then we will decide on its colour. So, the line equation can be given in this way or we get a equal to this, b equal to this, and c equal to this, and initial decision value d is given as 1.

(Refer Slide Time: 33:02)



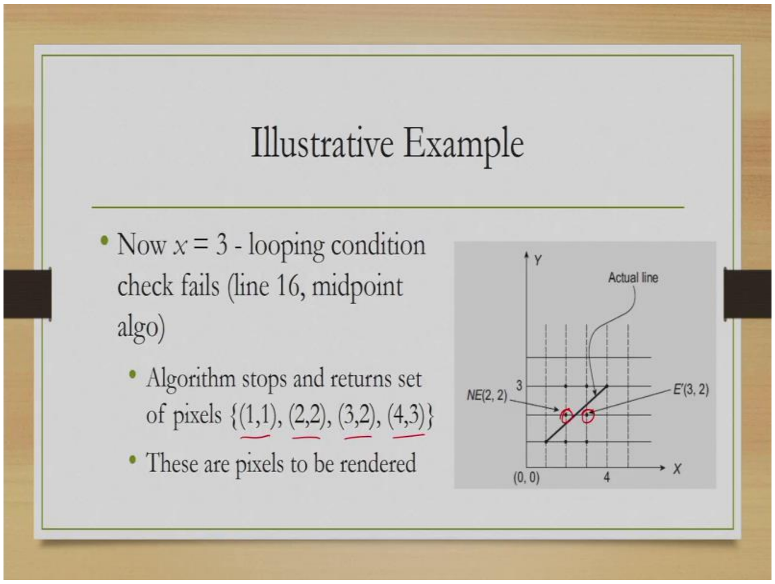
So, in the first iteration, we need to choose between NE and E, d is greater than 0 if we can see. So, then we need to choose NE that is this pixel and then reset d.

(Refer Slide Time: 33:23)



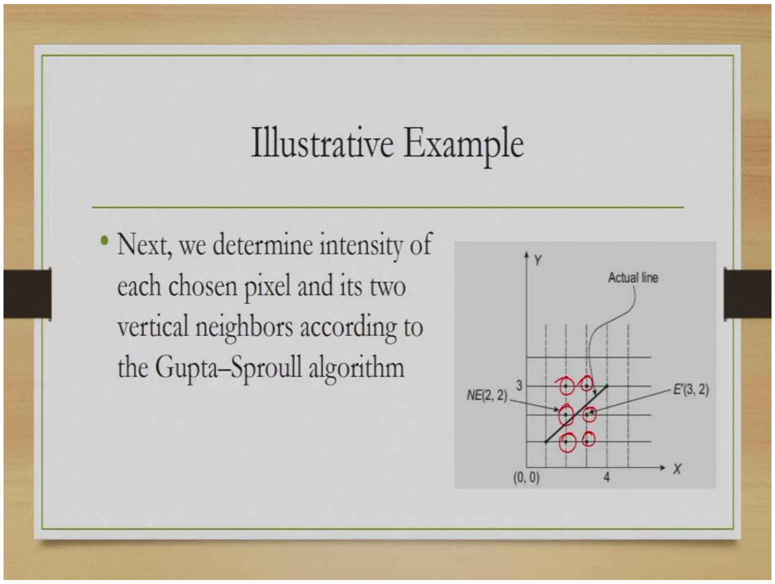
Now, after choosing NE the next iteration we choose this pixel E'(3, 2) depending on the value of d and then again we reset d.

(Refer Slide Time: 33:51)



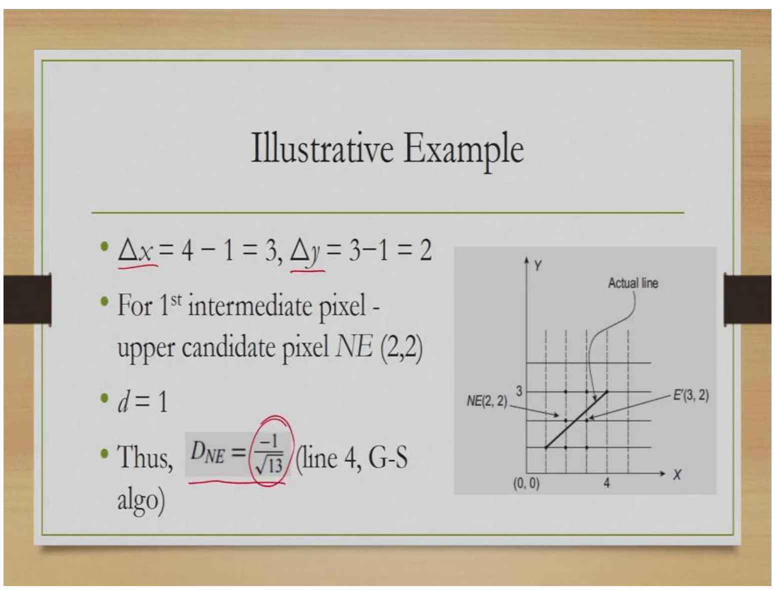
Now, after choosing these two pixels, we check that where is the other endpoint so stop. So, at the end the algorithm returns these four pixels, these are to be rendered. Now, while choosing a pixel we also use the modifications proposed in the Gupta-Sproull algorithm to assign colours.

(Refer Slide Time: 34:21)



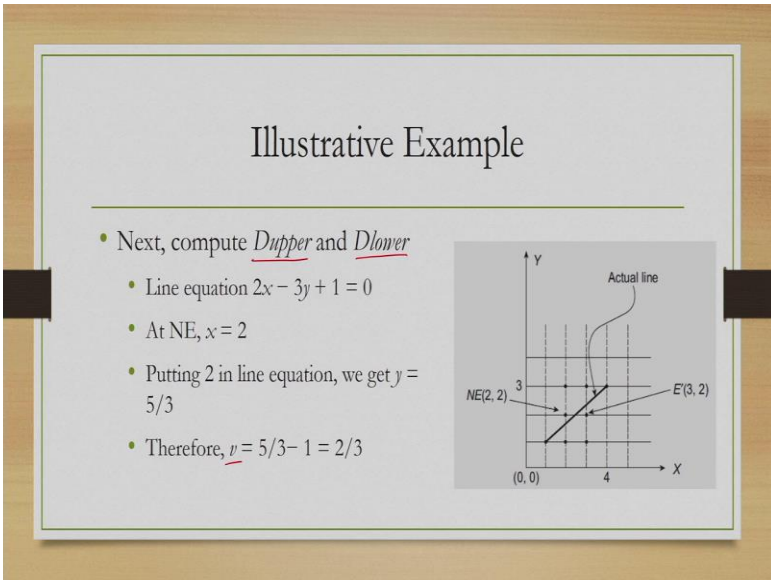
So, for example when we are talking about NE this is one chosen pixel, so you have to choose its colour as well as the colour of its two neighbours. Similarly, when we have chosen E we have to choose its colour as well as the colour of its two neighbours.

(Refer Slide Time: 34:46)



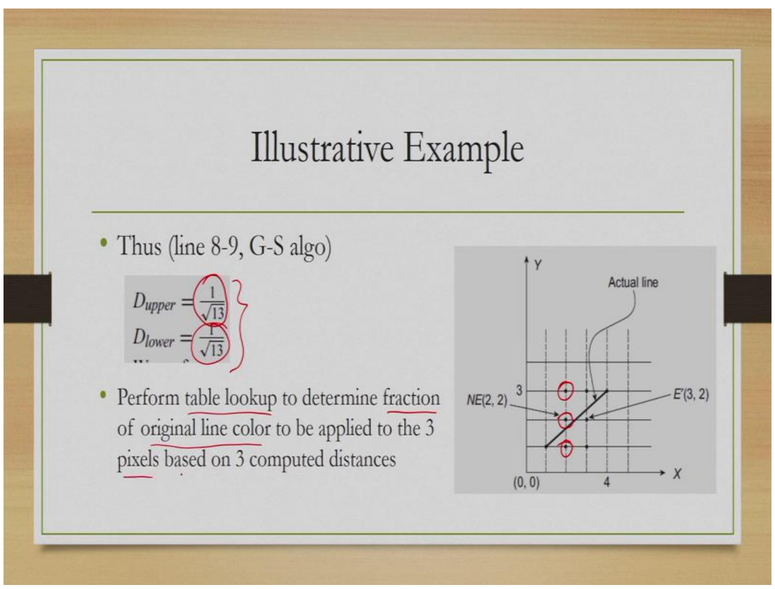
For that we determine Δx, Δy. Then we compute the distance from NE D which is given here and based on this distance.

(Refer Slide Time: 35:06)



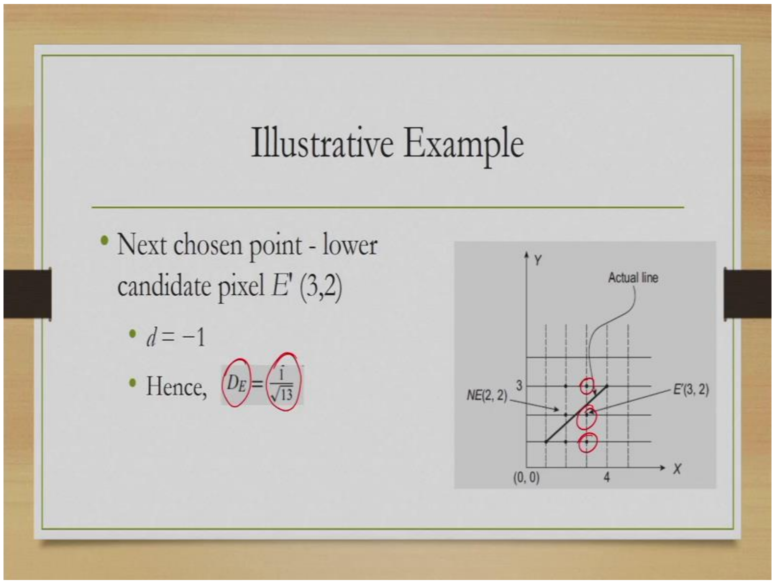
Also, we compute D_{upper} and D_{lower} by computing the first.

(Refer Slide Time: 35:25)



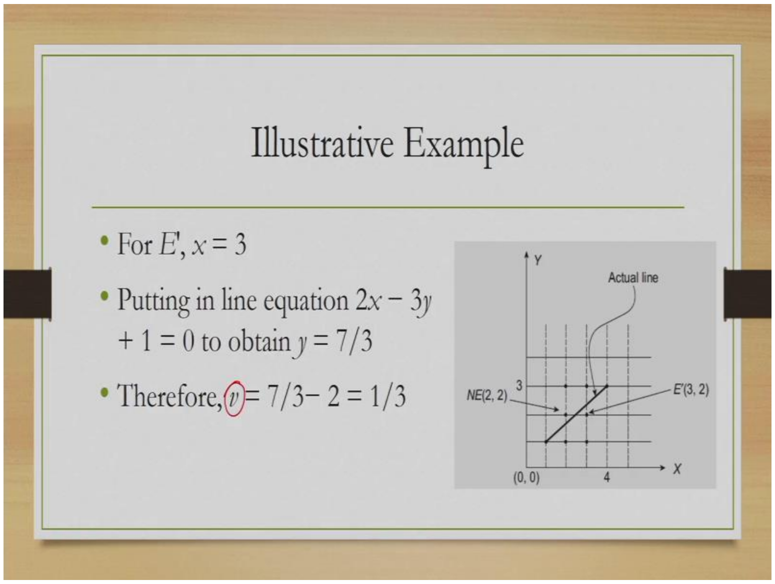
So, D_{upper} and D_{lower} are these two values for this particular pixel NE. So, you have D_{upper}, D_{lower}. Now, we use the table to determine the fraction of the original line colour to be applied to the three pixels based on the three computed distances.

(Refer Slide Time: 35:56)



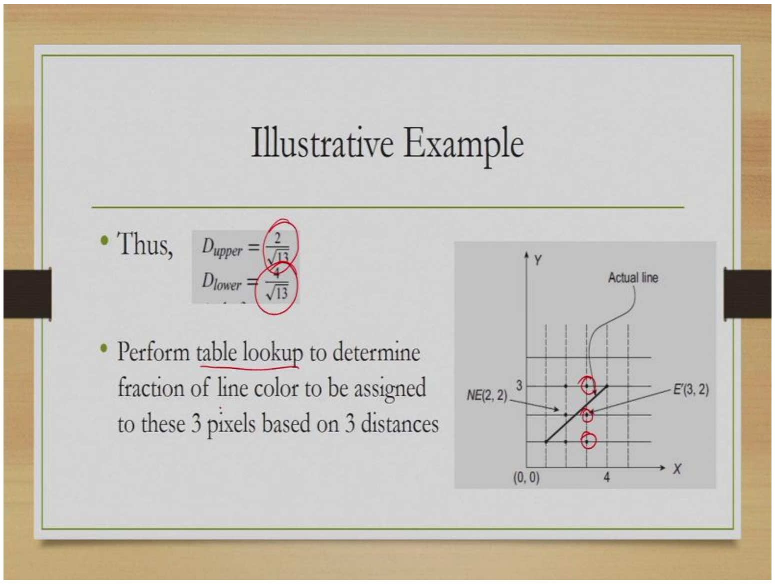
That is for any now for E' we have these two pixels, neighbouring pixels and the chosen pixel E'. So, here similarly we can compute D to be this value 1/√13.

(Refer Slide Time: 36:17)



And we compute v here to get the D_{upper} and D_{lower} values.

(Refer Slide Time: 36:33)



So, using v we get D_{upper} to be this value, D_{lower} to be this value and now we know the distance of this pixel from line as well as D_{upper} and D_{lower} again we go back to the table perform the table lookup to determine the fraction of the line colour to be assigned to these 3 pixels.

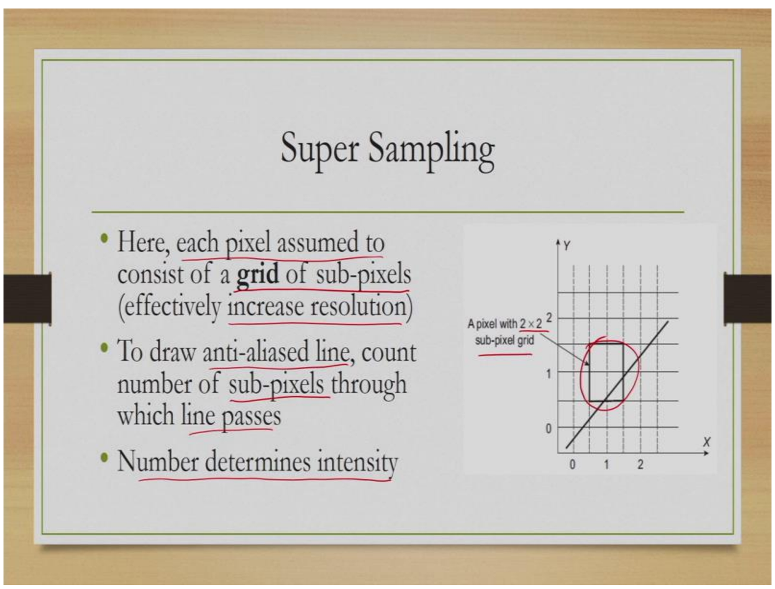
So that is how we choose the pixel colours. So in summary, what we do is in this Gupta-Sproull algorithm is we do not simply assign line colour to the pixels that are chosen to represent the line, instead we choose pixels following a particular line drawing algorithm and then compute three distances, distance of the pixel from the line and distance of the neighbouring pixels from the line. Here we are talking only in terms of vertical neighbours.

And based on these distances we find out the fraction of the line colour to be applied to the chosen pixel as well as the neighbouring pixels. Here we apply a cone filter function in the form of a table. In the table corresponding to the distances some fractions are mentioned. So, those were area sampling techniques

(Refer Slide Time: 38:15)

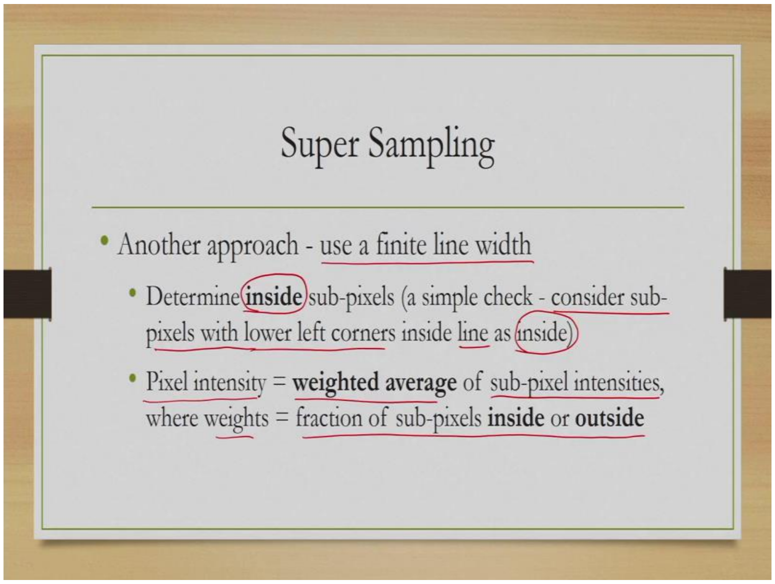
Now, let us try to understand the other broad class of anti-aliasing techniques known as super sampling.

(Refer Slide Time: 38:23)



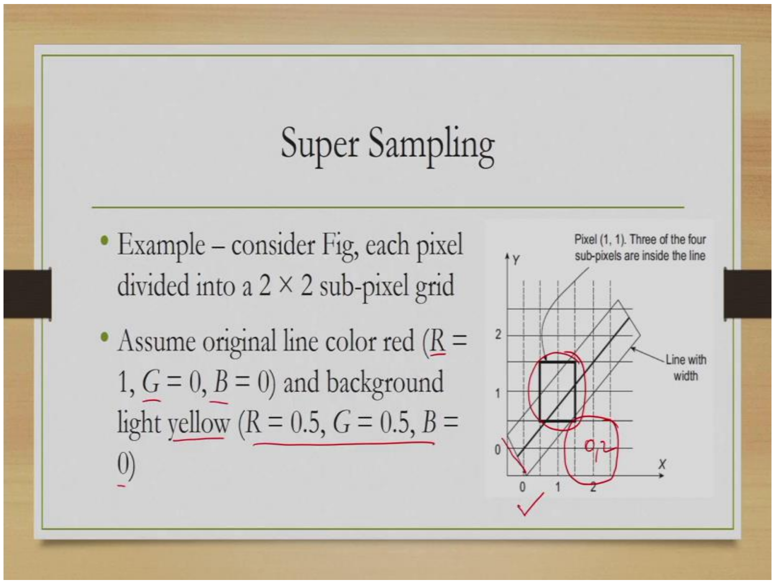
Now, in super sampling what we do? Here each pixel is assumed to consist of a grid of subpixels. In other words, we are effectively increasing the resolution of the display. Now to draw anti-aliased lines, we count number of sub pixels through which the line passes and this number determines the intensity to be applied. For example, here this whole square is a pixel, which is represented with two by two subpixel grid.

(Refer Slide Time: 39:12)



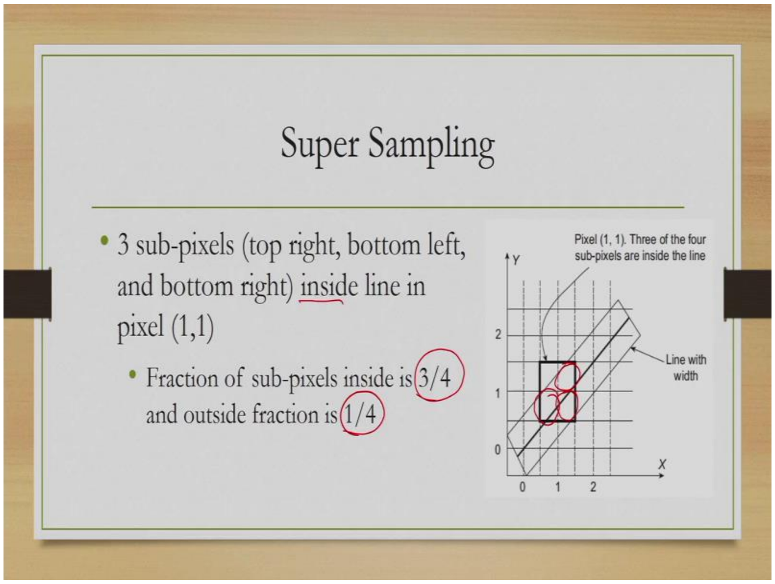
That is one approach. There are other approaches also, for example, we can use a finite line with determine inside subpixels that means the subpixels that are inside the finite width of the line. There can be a simple check for that. We may only consider those subpixels which has its lower left corners inside the line as inside. So if a subpixel has its lower left corner inside the line we can consider that subpixel to be inside subpixel and then pixel intensity is weighted average of subpixel intensities where weights are the fraction of subpixels inside or outside, that can be another simple approach.

(Refer Slide Time: 40:25)



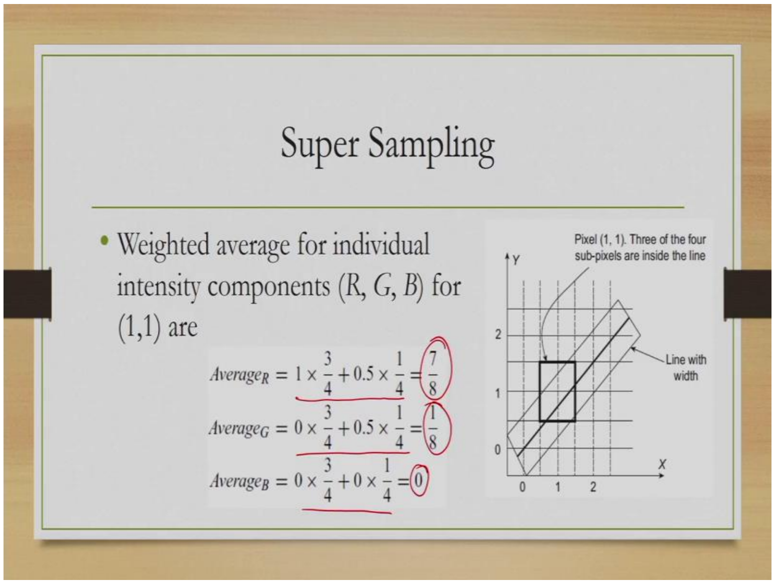
For example, in this figure as you can see the line has some width and each pixel has got 2×2 subpixel grid. So for example, this is a pixel which has got 2×2 subpixel grid, this pixel (0, 2). Similarly, this is a pixel (1, 1) which has got 2×2 subpixel grid. Now let us assume that the original line colour is given by this R, G, B values and the background light is yellow with again these R, G, B values. So, then how to choose the actual colour after rendering?

(Refer Slide Time: 41:30)



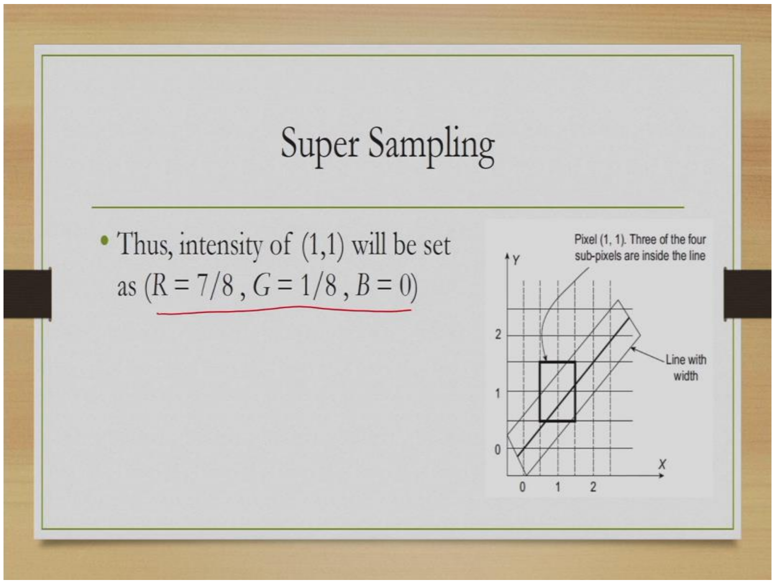
Now in the figure as you can see these three subpixels maybe considered inside the line. Thus the fraction of subpixels that is inside is 3/4 and outside fraction is 1/4.

(Refer Slide Time: 42:07)



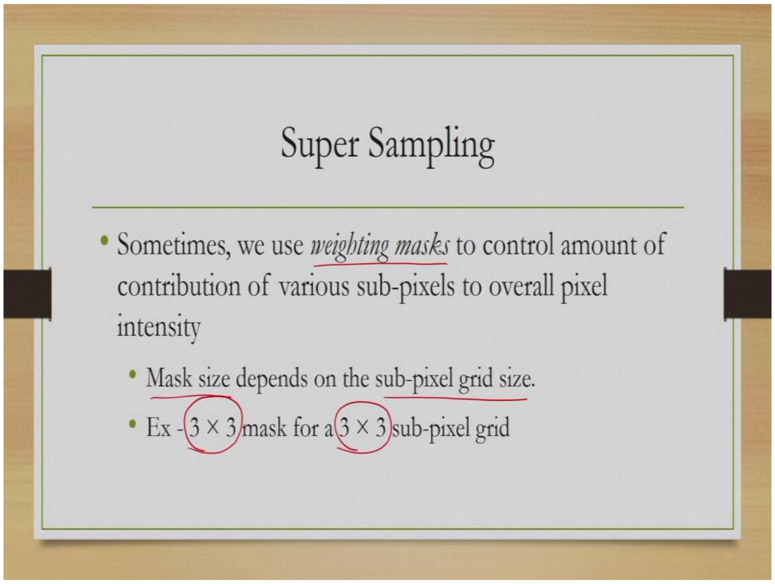
Now if we take a weighted average for individual intensity components, then we get the R value for this particular pixel as this or this value, G value will be this value and B value will be 0. And this R value or this G value and this B value together will give us the colour of that particular pixel.

(Refer Slide Time: 42:45)



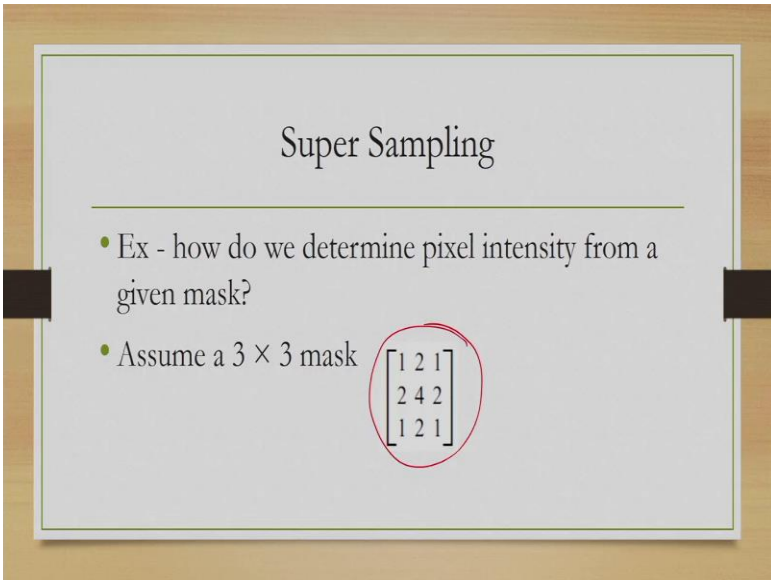
So, the intensity will be set as R equal this, G equal to this and B equal to this value and we got this by considering the fraction of subpixels that are inside the line and fractions that are outside.

(Refer Slide Time: 43:07)



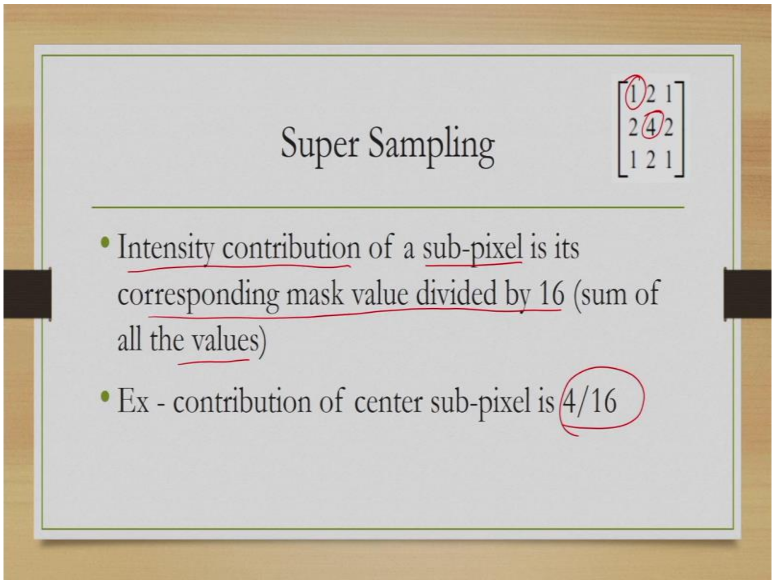
Sometimes we also use weighting masks to control the amount of contribution of various subpixels to the overall intensity. Now this mask size depends on the subpixel grid size, that is obvious because we want to control the contribution of each and every subpixel. So if we have a 3×3 subpixel grid then the mask size will be 3×3, there should be same.

(Refer Slide Time: 43:48)



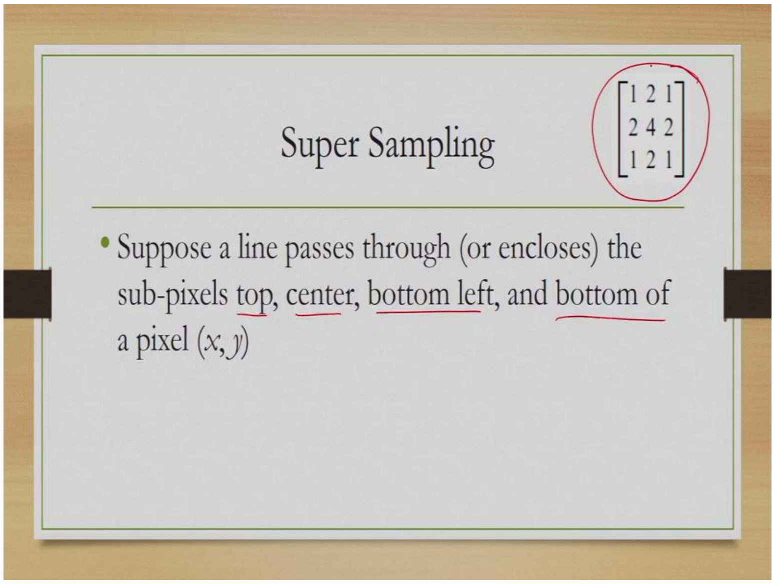
For example, consider a 3×3 subpixel grid and we are given these masks. So, given this mask and these three subpixel grid, how to choose the colour or the intensity of a pixel?

(Refer Slide Time: 44:09)



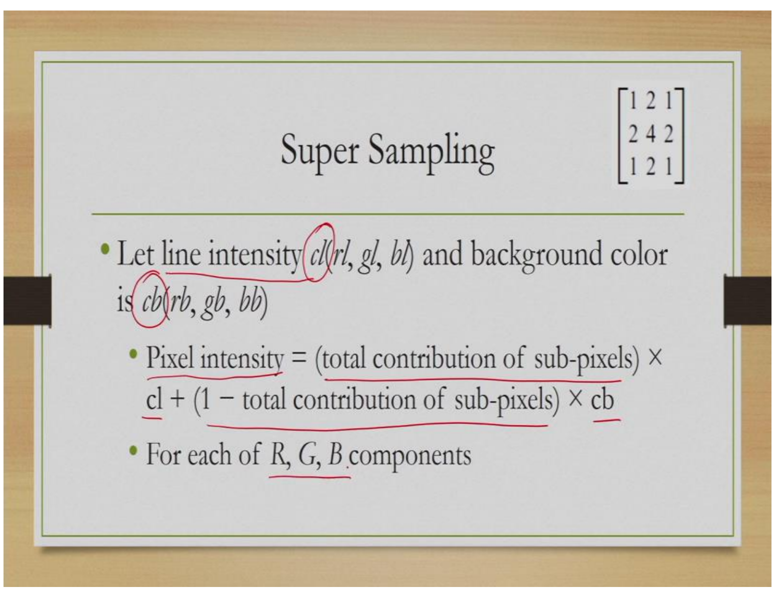
So, we can use this rule that the intensity contribution of a subpixel is its corresponding mask value divided by 16, which is the sum of all the values. So for the subpixel (0, 0) its contribution will be 1/16, for subpixel (1, 1) contribution will be 4/16 because the corresponding value is 4 and so on. So whatever subpixel intensity value is we will multiply with these fractions to get the fractional contribution and then we will add those up to get the overall pixel intensity.

(Refer Slide Time: 45:07)



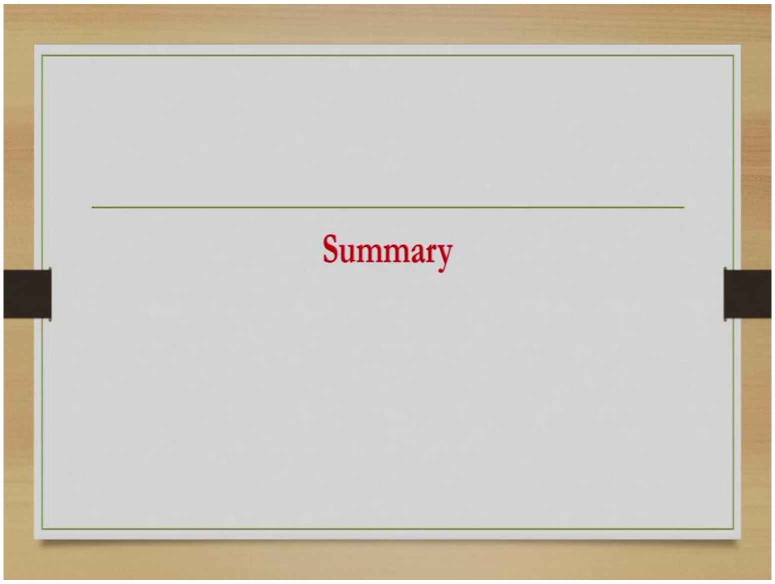
Now suppose a line passes through or encloses the sub pixels top, center, bottom left and bottom of a pixel. Assuming the same masks as shown here and 3×3 subpixel grid.

(Refer Slide Time: 45:32)



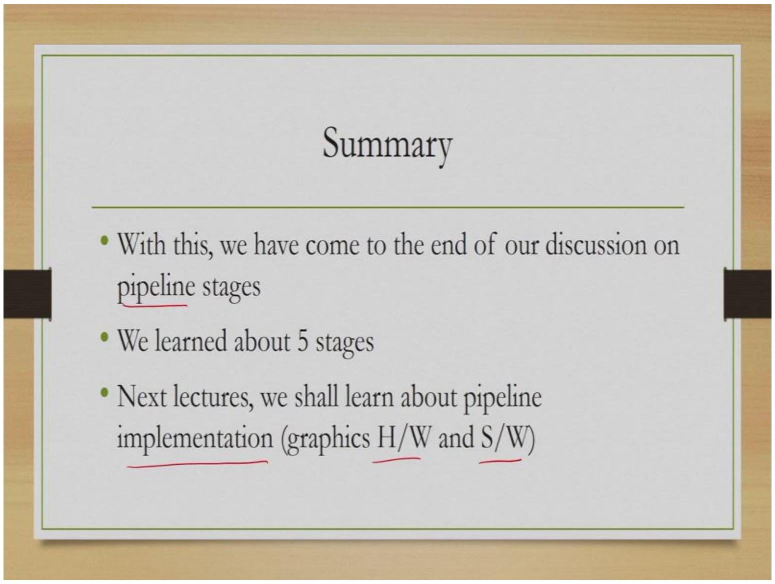
Then to get the line intensity at any particular pixel location or that pixel intensity for the line where line colour is given by cl original colour and background colour is given by cb, what we can do, we can use this simple formulation; pixel intensity will be given by total contribution of the subpixels into cl plus 1 minus total contribution into cb, this is for each of these R, G, B. components.

(Refer Slide Time: 46:11)



So, that is in effect what we can do it super sampling. So, let us summarize what we have learned so far.

(Refer Slide Time: 46:25)



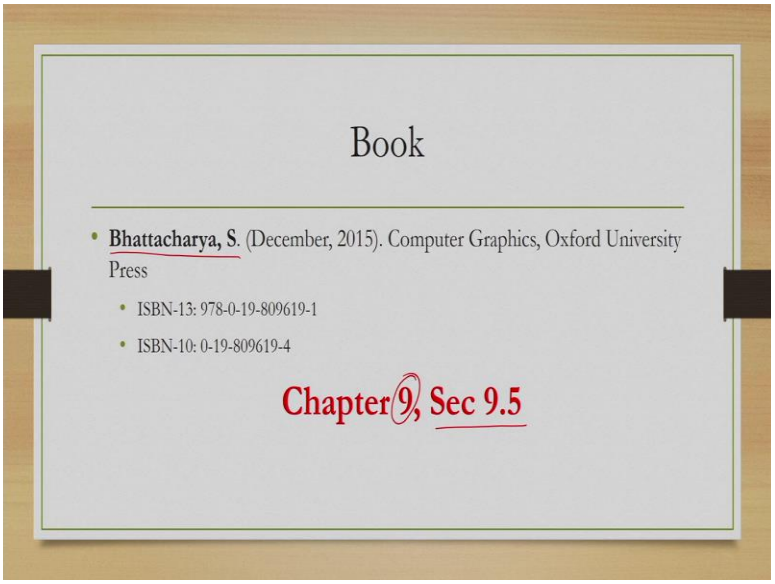
So with this discussion we have come to the end of our discussion on 3D pipeline and here we have covered in this series of lectures the five stages as I mentioned at the beginning. Today we have learned about one important technique that is aliasing and how to address it. It happens because in the fifth stage when we convert a shape for an image to a pixel grid then we get distortions and anti-aliasing techniques are useful for avoiding those distortions.

To do that, we follow broadly two groups of techniques; either area sampling or pre-filtering and super sampling or post filtering. In pre-filtering, we learned about Gupta-Sproull algorithm along with some other simple approach. Similarly, in post filtering we learned about three approaches there are many others, of course.

So these are meant to give you some idea of what are the issues and how they are addressed but clearly as you can see if we are going for anti-aliasing techniques, it entails additional computations and additional hardware support which of course has its own cost. With that we end our discussion today and that is also the end of our discussion on the 3D graphics pipeline.

In the next lecture, we shall learn about pipeline implementation, how the pipeline stages that we have learned so far are implemented, in particular we will learn about the graphics hardware and software. In the introductory lectures we learnt in brief the very small way graphics hardware and software, here we will go into more details.

(Refer Slide Time: 48:53)



Whatever I have discussed today can be found in this book. You are advised to go through chapter 9 section 9.5 for more details on the topics. So see you in the next lecture, till then, thank you and goodbye.