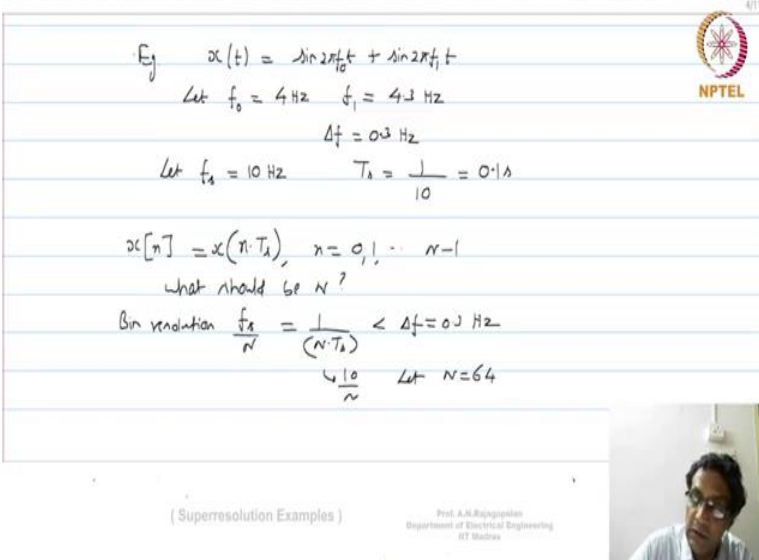


**Image Signal Processing**  
**Professor A. N. Rajagopalan**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Madras**  
**Lecture 83**  
**Super-resolution Examples**

(Refer Slide Time: 0:20)



Handwritten notes on a slide titled "Superresolution Examples". The notes show the following calculations:

$$x(t) = \sin(2\pi f_0 t) + \sin(2\pi f_1 t)$$

Let  $f_0 = 4 \text{ Hz}$ ,  $f_1 = 4.3 \text{ Hz}$

$$\Delta f = 0.3 \text{ Hz}$$

Let  $f_s = 10 \text{ Hz}$ ,  $T_s = \frac{1}{10} = 0.1 \text{ s}$

$$x[n] = x(n \cdot T_s), \quad n = 0, 1, \dots, N-1$$

What should be  $N$ ?

Bin resolution  $\frac{f_s}{N} = \frac{1}{N \cdot T_s} < \Delta f = 0.3 \text{ Hz}$

$\hookrightarrow \frac{10}{N} \quad \text{Let } N = 64$

( Superresolution Examples )

Prof. A.N. Rajagopalan  
Department of Electrical Engineering  
IIT Madras

Okay, so we were at this 1D super-resolution example, where we had said that we have a signal  $X$  of  $t$  that is given as  $\sin 2\pi f_0 t + \sin 2\pi f_1 t$ . Suppose let us just assume that  $f_0$  is, I am just taking some values arbitrarily 4 hertz and let me say  $f_1$  is equal to 4.3 hertz that means a difference between the two is  $\Delta f$  which is equal to 0.3 hertz. And the whole idea is we want to be able to draw samples from  $x$  of  $t$  and then use those samples in order to be able to super resolve these two in the sense that we should be able to resolve both frequencies when you get your  $x_n$ .

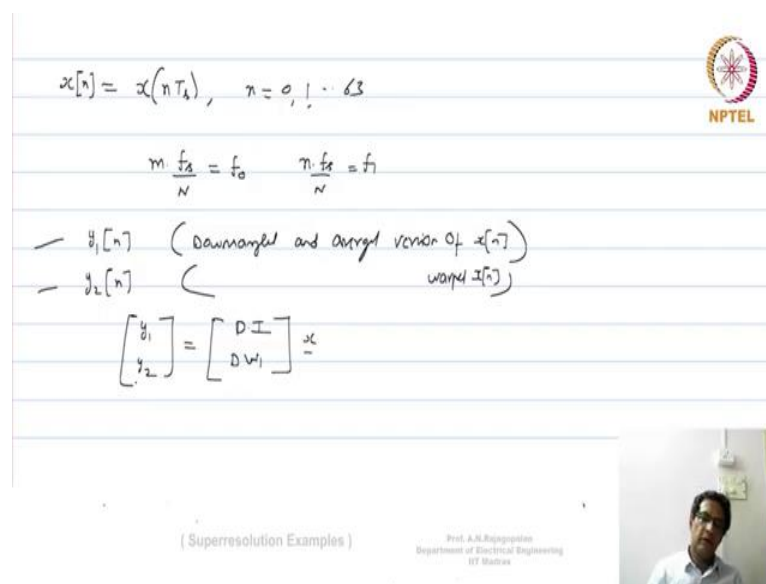
Now, first of all we had to make sure that  $x_n$  has both the frequencies and then we have to look at down sample versions of  $x_n$  which will run into trouble because they would not be able to release these two frequencies and then we have to use this super resolution idea by taking multiple observations and combining them under a, within a super resolution framework, alright, that is the idea of this example.

Now, our sampling rate in order to avoid aliasing and  $x_n$ , let  $f_s$  be equal to 10 hertz. Now, in this case the maximum which is of course, which is clearly greater than twice the maximum of the frequency which is 4.3, two times 4.3 is 8.6, therefore we have safely taken  $f_s$  to be 10 hertz, therefore our sampling interval is  $1/10$  which is equal to 0.1 second. This is our sampling interval. Now, we have to get our  $x_n$ , therefore  $x_n$  will be simply  $x$  of  $n$  in time  $t_s$ , where your  $n$  should go from whatever.

And  $n$  going from 0, 1, all the way up to 1 minus 1, now we have to see what should be  $n$ . It should be the capital  $N$ . That means how long should I really watch this signal  $x$  of  $t$  in order to be able to resolve, in order to be able to not lose this resolution ability. Now, we know that when a computer computes the 2D, when you compute a DFT, when you compute a DFT of  $x$  of  $n$ , the bin resolution in the DFT is going to be  $f_s$  by  $n$  which is nothing but  $1$  by  $n$  times  $t_s$ .

Which is why we say that  $n$  times  $t_s$  is the amount of time for which you are watching  $x$  of  $t$ , therefore the more you watch the signal, their ability to produce, ability to kind of, to be able to resolve frequencies will be higher. Now, in our case we want this  $n$ ,  $1$  by  $n$  such that this  $f_s$  by  $n$  should be less than  $\Delta f$  because that is when we can see these two frequencies falling into different bins. So,  $\Delta f$  is of course 0.3 hertz and our  $t_s$  of course is actually 0.1, so you got 10 by  $n$  here and therefore in order to just play it safe, let  $n$  be equal to 64 because that will ensure that 10 by 64 is definitely less than 0.3.

(Refer Slide Time: 3:26)



Handwritten equations on the slide:

$$x[n] = x(nT_s), \quad n = 0, 1, \dots, 63$$

$$m \frac{f_s}{N} = f_0, \quad n \frac{f_s}{N} = f_1$$

Below the equations, two signals are defined:

- $y_1[n]$  (Downsampled and averaged version of  $x[n]$ )
- $y_2[n]$  (warped  $x[n]$ )

The DFT equations are written as:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 W_1 \end{bmatrix} x$$

NPTEL logo is visible in the top right corner of the slide.

Video inset shows Prof. A.R. Nagarkar, Department of Electrical Engineering, IIT Madras.

So, suppose you take  $n$  equal to 64, and now you sample  $x$  of  $t$  as  $x$  of  $n$  times  $t_s$  and then you got like  $n$  equal to 0, 1 all the way up to 64, because you got 64 samples. So, now this is your  $x$  of  $n$ . This is the sequence  $x$  of  $n$ . Now, in this you will be able to see those two bins occurring at inappropriate places because you have to see as to some  $n$  times  $f_s$  by  $n$  will give you  $f_s$  by  $n$  where  $n$  is 64, will give you your first frequency  $f_0$  and some other  $n$  times  $f_s$  by  $n$  will give you your second frequency which is  $f_1$ .

So, you will be able to, if you compute DFT of  $x_n$ , you will see two peaks at those two frequencies. Now, the problem is you are going to not get from  $x$  of  $n$ , now we will assume that all the  $(())(4:14)$   $Y_1$  of  $n$  which will down sample and averaged version of  $x$ , averaged version of  $x_n$ . Like if we do this, similarly,

just as we showed the 1D example, use  $D_x$  but now  $D$  will be 1, 1, 0, 0, 0, 0 and so on, 0, 0, 1, 1, 0, 0 and so on.

Again, then  $Y_2$  of  $n$ , so for this let us assume some warp of  $x$  of  $n$  and again create of warped  $x_n$ , now we have to assume some value for  $\Delta x$  for this, so that we can apply the warp matrix and get  $Y_2$  of  $n$ . Now, if we interpolate  $Y_1$  or  $Y_2$  to the size of  $x_n$  and try to compute the DFT of  $Y_1$  and  $Y_2$ , you will not be able to see those two peaks because you have lost because of down sampling and averaging.


Now, instead what you could do is in order to retrieve  $x$  you could take  $Y_1$ ,  $Y_2$ , start them up together and then you know the matrices, so in one case you have to like  $D_i$ , the other case it is like  $DW_1$ . Now, assuming you know the exact warp, now what you can do is you can compute  $x$  as the inverse of this matrix times  $Y_1$  and  $Y_2$  and that will give back here  $x$ .

So, in the sense whatever you have lost because of down sampling and averaging, you are able to see it, retrieve the  $x$ . Now, what are the things that you will wonder is, what about the warping matrix, who will give me, so in a real image super resolution problem, you will ask who will give me the motion, well the motion will have to be determined.

(Refer Slide Time: 5:50)

Handwritten notes and diagram on a slide:

- Diagram showing a sequence of frames:  $L_{R_1}$ ,  $L_{R_2}$ , ...,  $L_{R_m}$ . A point  $(t_x, t_y)$  is marked above the first frame.
- Assumptions:
  - Motion: In-plane translation
  - Flat Scene: Fronto-parallel scene
- Vector notation:  $(t_x, t_y, 0)$  labeled "in-plane".
- Equation:  $q_x t_x, q_y t_y$
- NPTEL logo in the top right corner.
- Video feed of a professor in the bottom right corner.
- Footer text: "( Superresolution Examples )" and "Prof. A.R. Rajagopalan, Department of Electrical Engineering, IIT Madras".



$$x[n] = x(nT_s), \quad n = 0, 1, \dots, 63$$


$$m \frac{f_s}{N} = f_0 \quad n \frac{f_s}{N} = f_1$$

$y_1[n]$  (Downsampled and averaged version of  $x[n]$ )  
 $y_2[n]$  (warped  $x[n]$ )

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} D I \\ D W_1 \end{bmatrix} x$$

( Superresolution Examples )

Prof. A.R. Kishoregoudar  
 Department of Electrical Engineering  
 IIT Madras



In the earlier days how it was done was, you had your LR1, which is lower resolution image, another lower resolution image LR2, all the way up to maybe LRm, and what you would typically do is, we would actually use Sift or any such algorithm that you have in order to be able to estimate the motion. Now, the motion itself was normally assumed to be in-plane translation of the camera. The camera is just assume to move, seem to follow an in-plane translation.

Now, why is this so? Because if you have an in-plane translation and there was an assumption on the scene. Now, what were all these assumptions? When you are doing this we are making some assumptions, let us look at what assumptions we have implicitly made, we have made an assumption that it is in-plane translation and this seem to be fronto-parallel. That means the camera, and the camera sensor plane and the scene are exactly parallel to each other and the scene is flat, fronto-parallel scene.

So, flat scene which is actually fronto-parallel that means there is no kind of 3D in the scene to introduce the parallax. We have seen what is the advantage of having a fronto-parallel scene, it means when the camera moves all the pixels move by the same amount because every pixel will encounter the same motion and therefore, we have only  $T_x$  comma  $T_y$  to calculate, you have a global  $T_x$ ,  $T_y$  to calculate, so basically that is the only motion.

Otherwise if you had use rotation or something else then, rotation is still okay but then if you had assumed a 3D scene then each point will move independently depending up on to at what depth it is from the camera and therefore that will lead to a more complicated situation. These days people are kind of working on super resolution for 3D scene but that is all very complicated.

Let us look at the simplest of cases where you assume a global motion and  $T_x$  and  $T_y$  is applicable for let us say for the entire image and therefore this leaves with you now two parameters or you can even afford a more general motion so you can go ahead and use Sift, so typically it is okay to assume  $T_x$ ,  $T_y$  and some angle  $\theta$  which is again in plane rotation. Because if you are not able to do a perfect in-plane translation that mean some rotation but then your warping matrix will then have to be changed here.

Then this  $W$  should then account for, just as you know how to account for any kind of homography. This will be a simple case of that. You will have to account for in-plane translation, and rotation and warp  $x$ . Now these have to be found from LR. Now what happens is, but then you know there is this high resolution, the motion is of the high-resolution grid, it is not of the low-resolution grid.

You are applying it on  $x$  and not on, this is a high-resolution grid. Therefore, how do you go from low resolution to high resolution,  $\theta$ , for example, for the time being let us assume that we are doing in-plane translation. Then in the earlier days what they used to do was, if your super resolution factor is  $q$ , then they will take the high-resolution grid motion to be  $q$  times  $T_x$  and  $q$  times  $T_y$ .

It is like saying that a 1 pixel motion in LR is equivalent to a 2 pixel motion in the high resolution grid, if the factor of super resolution is 2, which seems reasonable and that is where people used to compute motion from low resolution observations first, sample them up by a factor of  $q$  and then use that for the high resolution warping. Now, this is the way it was done in the olden days. But then in the last 10 – 15 years things have changed. Now, people right now they say generalization of the problem.


(Refer Slide Time: 9:18)

Recent trends:

- Allow for general camera motion
- Scene is flat and fronto parallel: Some work assume legend 3D
- Joint motion and image estimation


$$\min_x \sum_i \|y - D_w x\|^2 + \gamma \text{ prior}$$

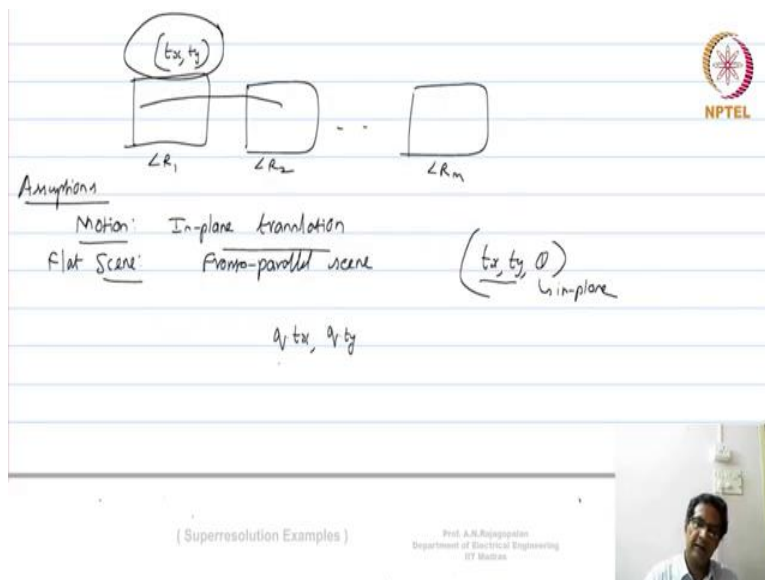
$$\min_y \sum_i \|y - D_w x\|^2 + \beta \text{ prior on camera motion}$$



( Superresolution Examples )

Prof. A.R. Rajagopalan  
Department of Electrical Engineering  
IIT Madras





Now, right now what are the recent trends? The recent trend is to allow for general camera motion, so do not really say restrict yourself to simply an in-plane translation or something and the scene is still pretty much flat and fronto-parallel. Now, this has not gone, the maximum that has been done is 3D, well some works assume layered 3D. What that means?

That means you have a 3D scene but they are all various fronto-parallel layers and something is the front, something is back, something is even further back and so on, so the advantage is that if you have a situation like that then the translation factor simply scales, so if you find for one layer then you can basically find for other layers by simply translating, by simply taking the depth to be, simply translate to a scale factor.

And the other thing, so all these are complicated works, we are not going to go into details of this but I am just trying to throw some hint upon what is going on. Other thing is joint motion and image estimation. Now, this is important because we really look at it. People do not want to do motion first and then whatever scale by factor of  $q$  and then use that on the image because if something went wrong during a sift or whatever that you are using to compute motion, then that will remain fixed.

There is no way to change it, you just blindly believe that the motion estimation is correct and therefore when you try to invert whatever it, when you have solved for  $x$ , your  $x$  will kind of, all the errors that you have made, the motion estimation will kind of creep into  $x$  and therefore  $x^2$  will no longer look as nice as you might want it to look like.

Now, that is why people do a joint motion estimation, motion remains, that means in one iteration you solve for motion and the other iteration you solve for the image and then you keep on updating both, which is kind of alternating framework. As far as the image is concerned, people are using priors, like the L1 prior and so that I already mentioned. The observation model will still be normal. The observation model will become  $Y$  minus  $DWx$  norm square and summed over all  $i$ .

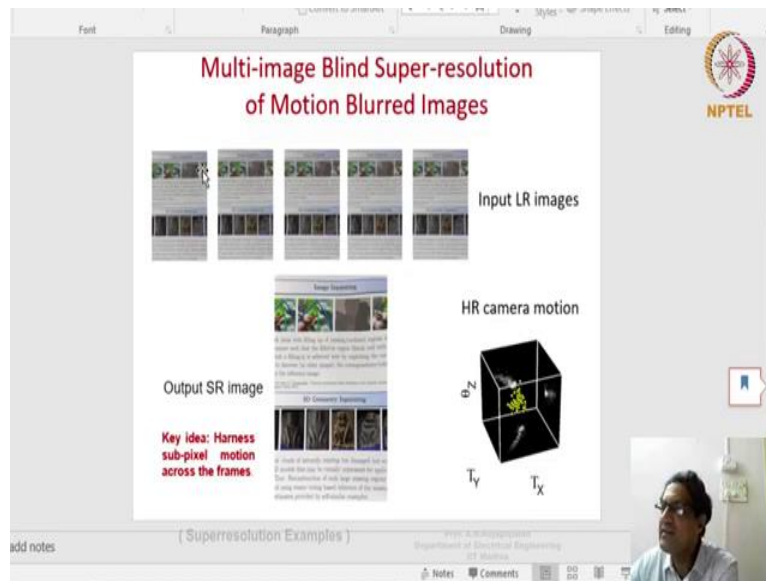
Now, minimize over  $x$  and the camera pose, so what you have is plus some  $\gamma$  times, then you will have a prior for the image, prior for  $f$  and then you have plus, a prior let us say  $\beta$ , a prior for... Well, I mean, what you typically do is of course, when you alternate, and not the sense that you do not have a prior here, now instead what you have this, this is when you solve for  $x$ . And then you alternate.

Next time you solve the camera motion  $p$  because like  $i$  and then norm of  $y$  minus  $DWx$  square plus let us say  $\beta$  times a prior on camera motion. This will be typically sparsity and so on. So, sparsity prior and this is  $p$  is 1, that will sit in this warping matrix. So, what are those sparse camera poses that you have in order to be able to get these warping matrices for each of these  $X$ s. The other reason why this is done is because if you simply up sample by  $q$ , this  $t_x$  and  $t_y$ , take another  $q$ , this is actually incorrect.

This is not correct because this is used as LR2 can be exactly identified from LR1 but then the whole point is, if that is the case then you actually cannot do super resolution because LR2 does not carry anything new. Now, between LR1, LR2 there is nothing new, that was a pathological case that I mentioned. For super resolution LR2 cannot be identical to LR1, therefore the  $t_x$  and  $t_y$  that are actually calculated may not be correct because these two images are not identical.

They are not just shifted versions of each other and therefore multiplying by  $q$   $t_x$  and  $t_y$ , you are already making an error in terms what estimates for the motion you are taking and therefore the recent trend is to actually do what is called joint motion and image estimation and this is the way it is all done and it is a very nice problem, very elegant and as you can see it brings in all the prior, brings in all the observation model that you have and tries to undo these effects.

(Refer Slide Time: 13:49)

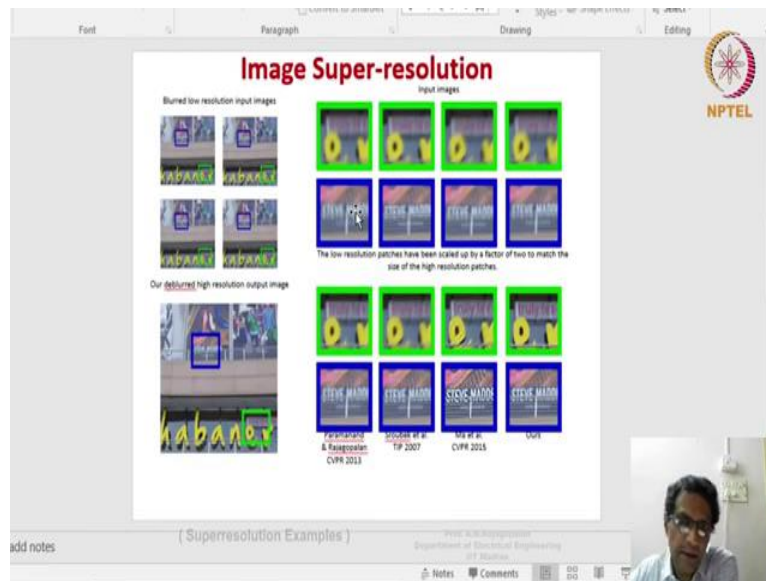


Here is an example that I am going to show. Here is a super resolution example that we have done in our lab. As you can see these are your low resolution observation and as you can see we are not able to read them very well and if you take these... Typically what can also happen is, one more thing that it should add is, when you move you could also incur some kind of motion blur, and therefore the recent trend is to not only solve for LR images, the HR image but they also assumed that the lower resolution observation is motion blurr.

And therefore, if you see that these images are slightly motion blurred and you are not able to read them very well, read the text very well. They are multiple observations, all of them low resolution, and then the high resolution looks like this and here of course, you are able to start reading 3D, geometry painting and then you are able to read what else is there in that text, filling up ( ) (14:39) and there is missing information and so one, those things are very difficult to read here.



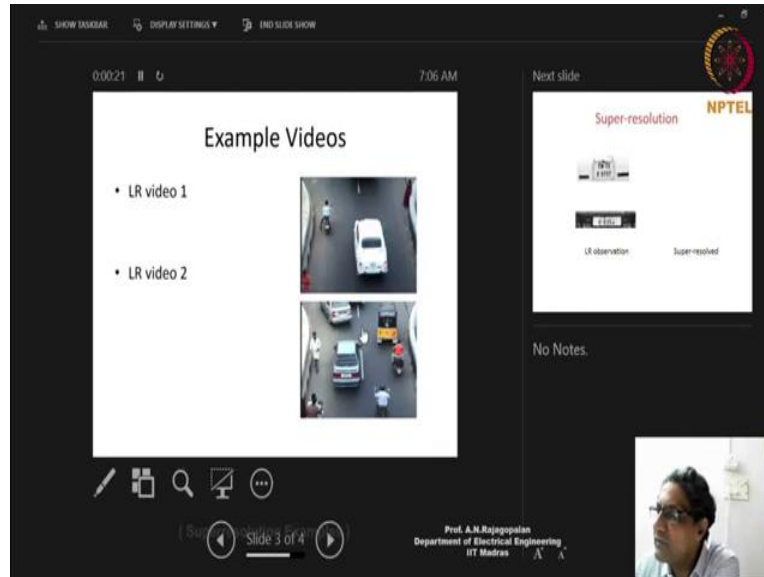
(Refer Slide Time: 14:48)



Now, if you go further in the next example here, this is one of the malls where my student went and captured some pictures, so these are 4 low resolution images and here is high resolution image constructed over there and clearly you can see a lot more details here as compared to this and here we have shown the low resolution patch is zoomed up and here is the high resolution patch for each of those low resolution patches and we also compared with some other methods.

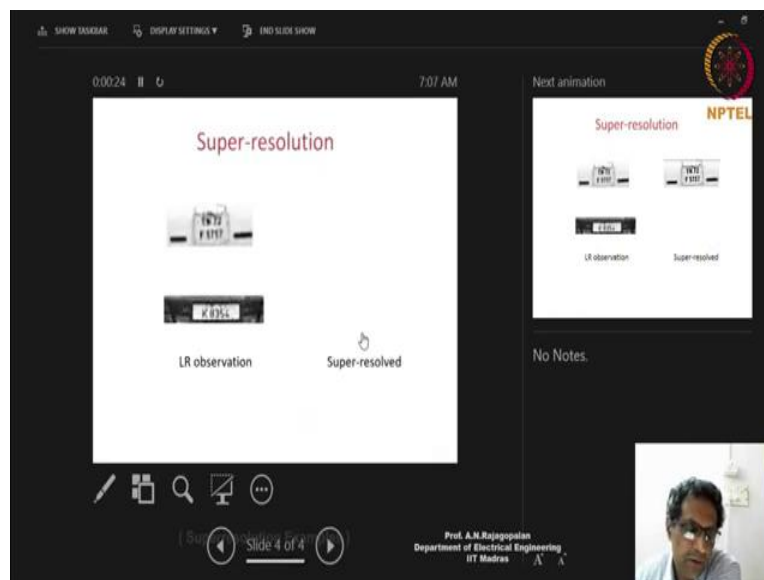
Ours is sitting here and you can see that the Steve Madd and then Truly Tech and so one which is very difficult to read in the LR, you are able to see here it is hardly able to see Truly Tech whereas here you are able to see that. And then here with some difficulty we can see Steve Madd whereas here it is much more clear. So, basically such is the ability of the super resolution algorithm

(Refer Slide Time: 15:34)



Now, here is one more, here is a video that I am going to play and let me just play this video for you. And here what you see is really a video where you see this white Ambassador car, you want to be able to pick up the number plate with this car and if you try to use low resolution observation, you are not able to make much out of them therefore we try to apply the super resolution idea on this video. Similarly, the other one, here is this Maruti car, that is the blue colour car or whatever, we want to get its number plate.

(Refer Slide Time: 16:06)



Now, if you see we directly use the low resolution frame and then crop the number plate, this is what you see, very difficult to make up what it is and even here, the last digit, even it looks like K835 and last it

could be 2 or 4, it is not very clear, but if we do super resolution then the first becomes TN72, some F5757, now we do not whether that is the number but then it is much more clear than this and TN makes sense because it was actually a Tamil Nadu vehicle, so therefore to carry TN it makes a lot more sense.

It has done something good. One you super resolve there is no doubt, this is no longer 2, this is clearly 4. So, you can see the ability of the super resolution algorithm and now it is also very important in several areas especially if you are doing some kind of medical imaging, where you want to see finer details, even if you have high resolution picture you do not want to blow it up further super resolve in order to be able to see finer details and so on.

Therefore, this is an exciting area, lot of research going on, even in this. Of course, it is impossible to cover all of that in a course like this but I hope that you have an idea, you have really a good idea about how this super resolution idea works.