Deep Learning for Computer Vision

Professor Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture 15
Feature Matching

Over the lecture so far, we have talked about basic methods to process images, we talked about operations such as convolution, correlation, and then we talked about how we can use such operations to detect edges in images, corners in images, different kinds of corners, different methods to extract those corners, as well as how do you describe these corners in ways in which they could be used for further tasks. We also talked about how this process could be similar to how the human visual system also perceives the world around us.

One of the aspects that we mentioned is, if you have two different images, and let say, you want to stitch a panorama consisting of these two images or more than two we ideally detect interest points in both of these images, get descriptors of each of these points in both of these images and then we match points across these images.

How you match is what we are going to get into next. So, over the next few lectures we will talk about a few different methods to match key points between images, not just key points between images, we will try to use these methods to do other kinds of tasks like finding different kinds of shapes and images such as circles, lines, or whatever shape you like, as well as even more descriptors from what we have seen so far.

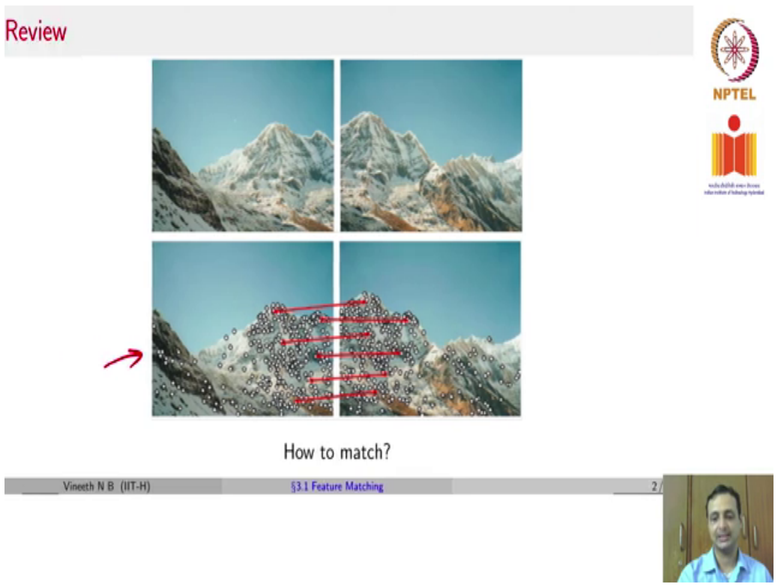
(Refer Slide Time: 01:52)

Acknowledgements

- Most of this lecture's slides are based on lectures of **Deep Learning for Vision** course taught by Prof Yannis Avrithis at Inria Rennes-Bretagne Atlantique

Most of this week's lectures are based on the excellent lectures of Professor Yannis, at the University of Rennes, Inria in France.

(Refer Slide Time: 05:07)



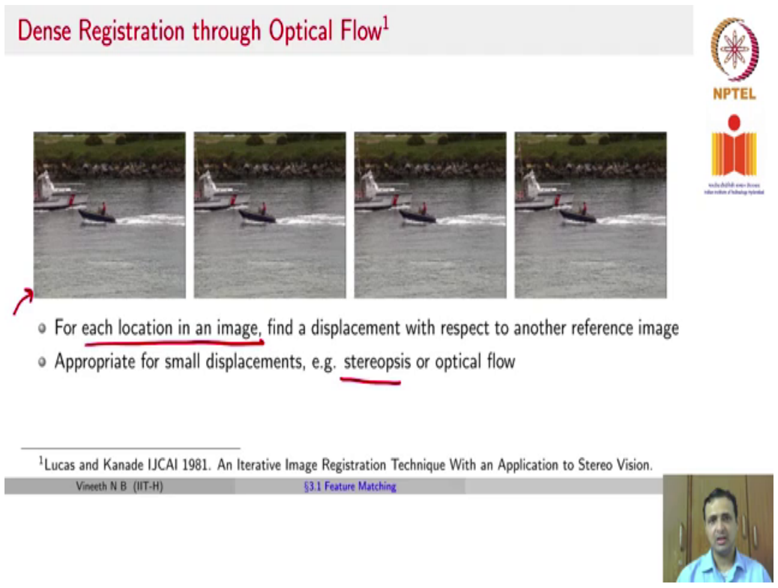
Review

How to match?

If you recall, we gave this example earlier of two images taken of the same scene, perhaps from different viewpoints, perhaps at different parts of the day, or perhaps with just different illuminations or different camera parameters. And if you want to stitch a panorama of these two images the standard process is to find key points and match them.

So, we know how to find key points in both these images individually. We also know how to describe each of those key points as a vector. We have seen SIFT, we have seen HOG, we have seen LBP, we have seen a few different methods to do this. The question that's left is if you now have the key points and descriptors in two different images, how do you really match them and be able to align them? That's what we will do next.

(Refer Slide Time: 3:05)



Dense Registration through Optical Flow[1]

- For each location in an image, find a displacement with respect to another reference image
- Appropriate for small displacements, e.g. stereopsis or optical flow

[1]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching

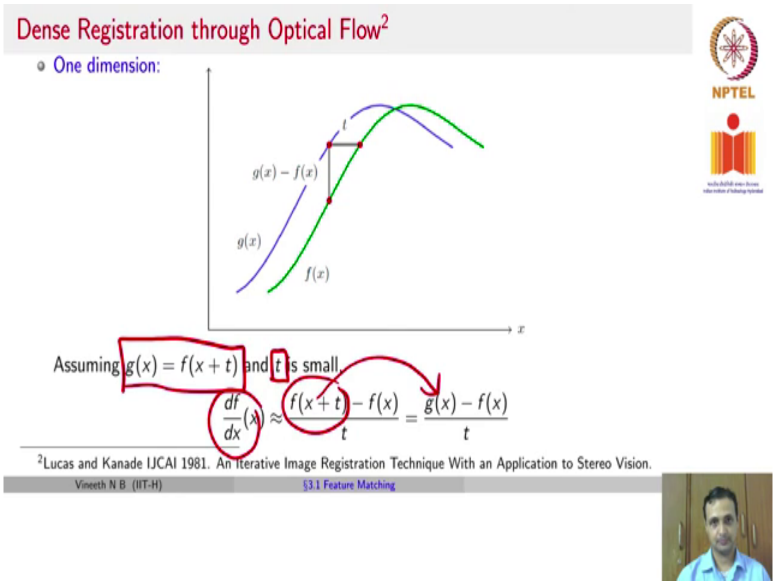
We will start with a very simple method called dense registration to optical flow, a fairly old method, which pertains to a setting where you have very small change between different images. So, if you again take the example of the cell phone, if you are going to gradually move your cell phone over a scene, and then you want to stitch a panorama the differences between successive images is going to be very little.

So, if you have tried this yourself you will notice that in certain cases if you move your hand very fast you will get an error message to repeat and move your hand very slowly to get a panorama from the app on your cell phone.

So, in these kinds of cases the displacement of the scene between successive images is very little. In these settings you can use this kind of a method called dense registration through optical flow. Here is a visual example of a scene where a boat is going across water. You can see that the scene is more or less the same, but a few changes in the positions of the boats.

Our goal here is for each location in the image, say a key point in the image we want to find a displacement with respect to another reference image. Once you have a displacement, we can simply place one image on top of the other image and be able to align them. So, this kind of a method of using dense registration is generally useful for small displacements such as stereopsis or optical flow.

(Refer Slide Time: 4:50)



Dense Registration through Optical Flow[2]

- One dimension:

Assuming $\boxed{g(x)=f(x+t)}$ and $\boxed{t}$ is small,

$$\frac{df}{dx}(x) \approx \frac{f(x+t)-f(x)}{t} = \frac{g(x)-f(x)}{t}$$

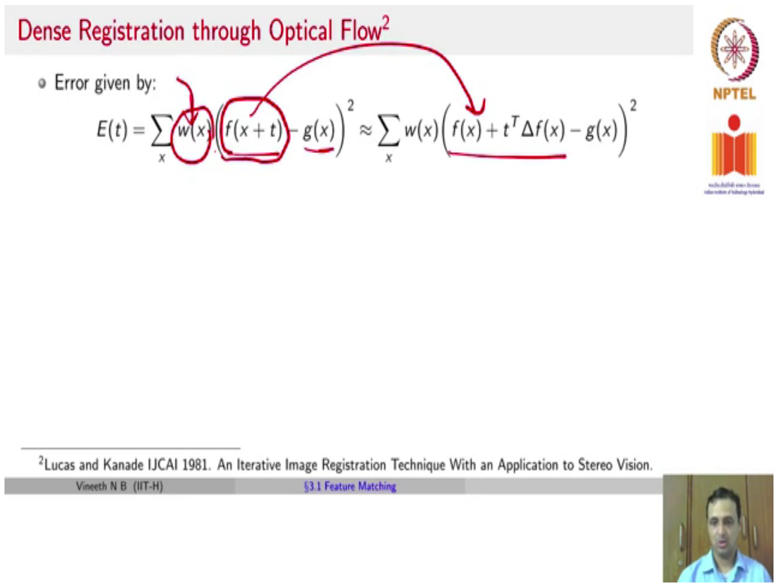
[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching

To understand how to do this, let's first take a one dimensional case. Let's work out the math and then we will go into a two dimensional case. So let's consider a one dimensional case, let's consider a function $f(x)$, which is given by this green curve and let's consider this function $g(x)$, which is simply a displaced version of the same $f(x)$, or mathematically speaking I can say that $g(x) = f(x + t)$ is just a displaced version of $f(x)$ and we also assume that t is small, we are only looking at small changes between successive images.

We know by first principles definition of derivative you can say that $\partial f/\partial x$ is given by $\lim_{t->0} (f(x + t) - f(x - t)) / t$ which would be the formal definition. But we know now that $f(x + t)$ is $g(x)$. So, which means we can write $\partial f/\partial x$ to be, $(g(x) - f(x)) / t$. Where do we go from here?

(Refer Slide Time: 6:09)



Dense Registration through Optical Flow[2]

- Error given by:

$$E(t) = \sum_x w(x)\left(f(x+t) - g(x)\right)^2 \approx \sum_x w(x)\left(f(x) + t^T \Delta f(x) - g(x)\right)^2$$

[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

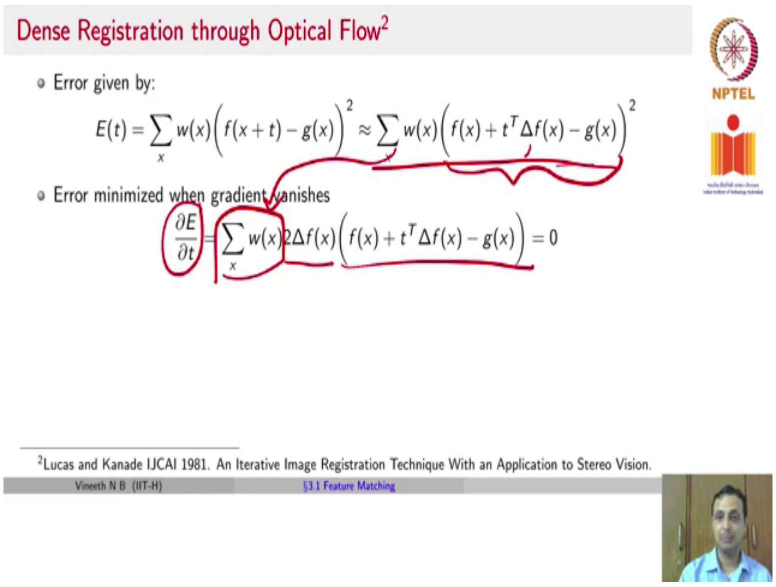
Vineeth N B (IIT-H) §3.1 Feature Matching

Now, we define the error between these two signals in this particular case, because we are considering a one dimensional equivalent right now that is going to be some weighted combination. Let's assume that this is very similar to the weighted autocorrelation that we talked about for the Harris corner detector, just that in that case we talked about autocorrelation. Here, we are looking at differences between two signals f and g.

So, you have $f(x + t) - g(x)$, that's going to be the difference, and you do a weighted combination of these two to be able to find the actual displacement. So, you have $w(x)\,(f(x + t) - g(x))^2$. Now, this second term, this first term here using a first order Taylor series expansion can be written as $f(x) + t^T f(x)$.

The remaining terms are the same across these two equations. The first term is simply expanded as a first order Taylor series expansion and you get the right hand side of this equation. Where do we go from here?

(Refer Slide Time: 7:21)



Dense Registration through Optical Flow[2]

- Error given by:

$$E(t) = \sum_x w(x)\Big(f(x+t) - g(x)\Big)^2 \approx \sum_x w(x)\Big(f(x) + t^T \Delta f(x) - g(x)\Big)^2$$

- Error minimized when gradient vanishes

$$\frac{\partial E}{\partial t} = \sum_x w(x) 2\Delta f(x)\Big(f(x) + t^T \Delta f(x) - g(x)\Big) = 0$$

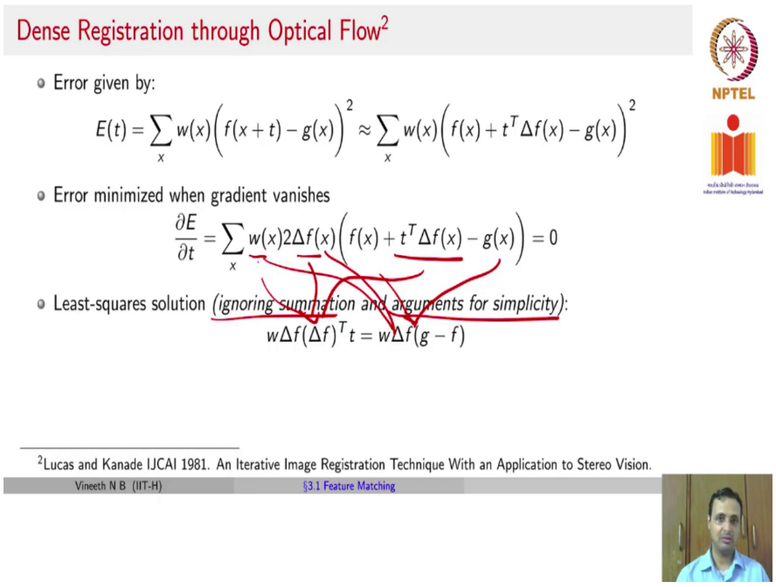
[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching

We know that the error is minimized when the gradient vanishes, so we take $\partial E/\partial t$, which is just going to take a simple derivative of this right hand most term which is going to be $w(x)$ summation of x, that part stays the same as here and the term that depends on t is this particular term $(f(x) + t^T f(x) - g(x))^2$.

So, if you take the gradient of that, you are going to have 2 into the entire term inside the brackets into the derivative of the term that's affected by t, which $\Delta f(x)$. So, you are going to have $2\Delta f(x)$ into the entire term inside the brackets. We want to set this gradient to 0, and then solve for what we're looking for.

(Refer Slide Time: 8:13)



Dense Registration through Optical Flow[2]

- Error given by:

$$E(t) = \sum_x w(x)\Big(f(x+t) - g(x)\Big)^2 \approx \sum_x w(x)\Big(f(x) + t^T \Delta f(x) - g(x)\Big)^2$$

- Error minimized when gradient vanishes

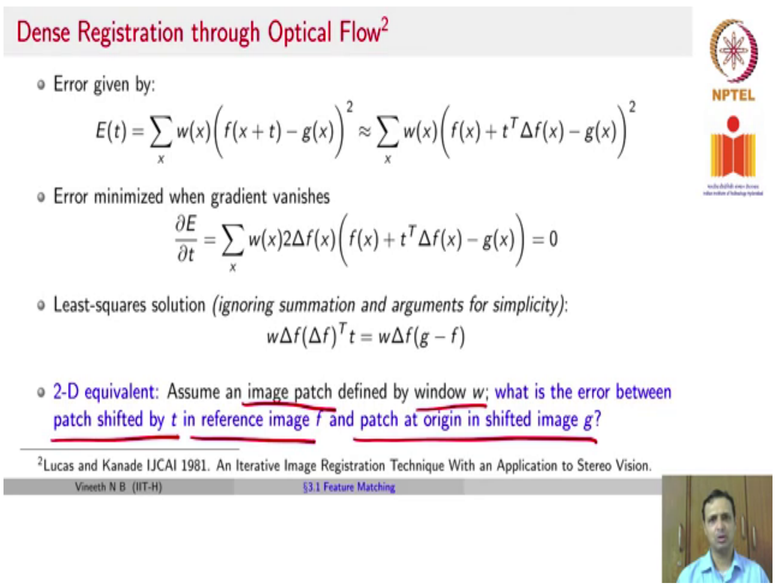
$$\frac{\partial E}{\partial t} = \sum_x w(x) 2\Delta f(x)\Big(f(x) + t^T \Delta f(x) - g(x)\Big) = 0$$

- Least-squares solution *(ignoring summation and arguments for simplicity)*:

$$w\Delta f(\Delta f)^T t = w\Delta f(g - f)$$

[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching



Dense Registration through Optical Flow[2]

- Error given by:

$$E(t) = \sum_x w(x)\Big(f(x+t) - g(x)\Big)^2 \approx \sum_x w(x)\Big(f(x) + t^T \Delta f(x) - g(x)\Big)^2$$

- Error minimized when gradient vanishes

$$\frac{\partial E}{\partial t} = \sum_x w(x) 2 \Delta f(x)\Big(f(x) + t^T \Delta f(x) - g(x)\Big) = 0$$

- Least-squares solution *(ignoring summation and arguments for simplicity)*:

$$w\Delta f(\Delta f)^T t = w\Delta f(g - f)$$

- 2-D equivalent: Assume an image patch defined by window w; what is the error between patch shifted by t in reference image f and patch at origin in shifted image g?

[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching

So, now simply expanding this equation, you can simply take terms out on both sides and write this out as we are just going to ignore the summation and the arguments just for simplicity of explaining this. If we ignore those you would have $w\Delta f(\Delta f)^T$, these terms are branched here. Similarly, $w\Delta f(g - f)$ if you take it the other side. 2 does not matter here because we are anyway equating into 0. So, now by doing this you can solve for the Δf and be able to figure out the displacement between these two signals.

What is the two dimensional equivalent it's exactly the same set of equations just that instead of a 1D signal you will now have an image patch that's defined by a window w and we then try to find what is the error between the patch shifted by t in reference image f, and the patch at origin and shifted image g.

So, if you moved f by a certain t in the original image do you get g is the question that we want to ask. We want to find that t that minimizes this change, because that would give you the displacement between f and g. Now, by solving for this you can get the value of t, find the displacement and now be able to match or align these two images. A very simple solution.

(Refer Slide Time: 9:51)

Dense Registration through Optical Flow[2]

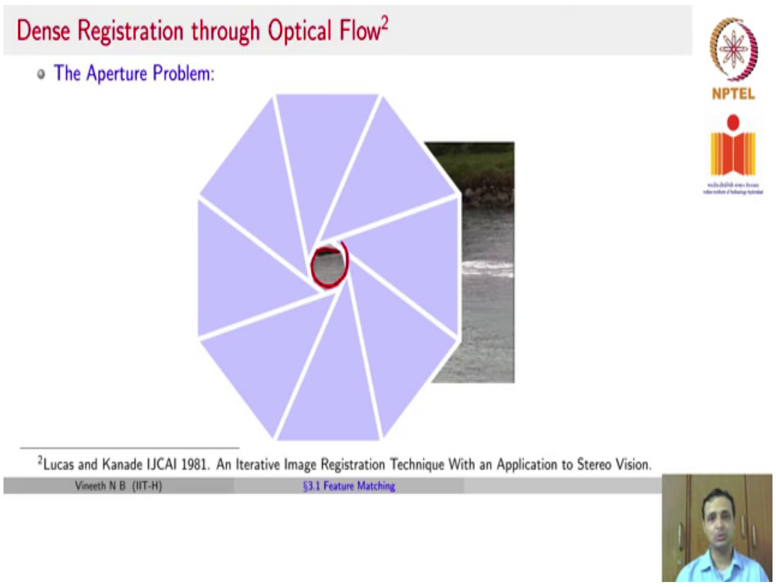
- The Aperture Problem:

[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching

One of the problems of this approach is the same aperture problem that we have dealt with when we moved from images to the Harris corner detector.

(Refer Slide Time: 10:02)



Dense Registration through Optical Flow[2]

- The Aperture Problem:

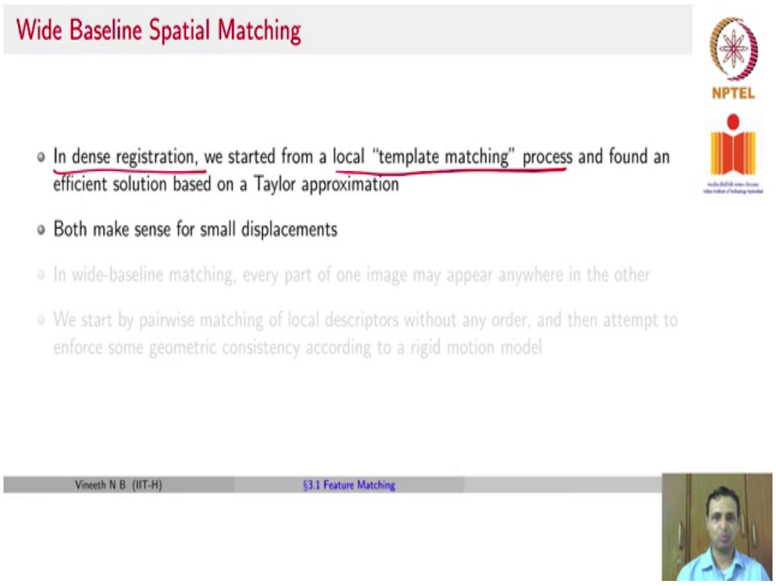
[2]Lucas and Kanade IJCAI 1981. An Iterative Image Registration Technique With an Application to Stereo Vision.

Vineeth N B (IIT-H) §3.1 Feature Matching

Remember, the aperture problem simply means that you can only solve this problem for a very local neighborhood. Why so? Because the entire definition or the way we solve the problem assumes a local neighborhood. If you looked at the first order Taylor series expansion that approximation holds only for a local neighborhood, which means this entire formulation holds only if the displacement is inside a very small neighborhood and that's the reason why we said that this method works when there is only very small changes between successive images of the frame.

So, what do we do if there is more than a minute difference between these two images? For example, a few slides ago, we saw those images of those mountain ranges. It did not look like those two images were displaced by a very small amount; it looked like there was a significant rotation or a significant perspective difference in how those pictures were taken. How do you solve those kinds of problems?

(Refer Slide Time: 11:08)



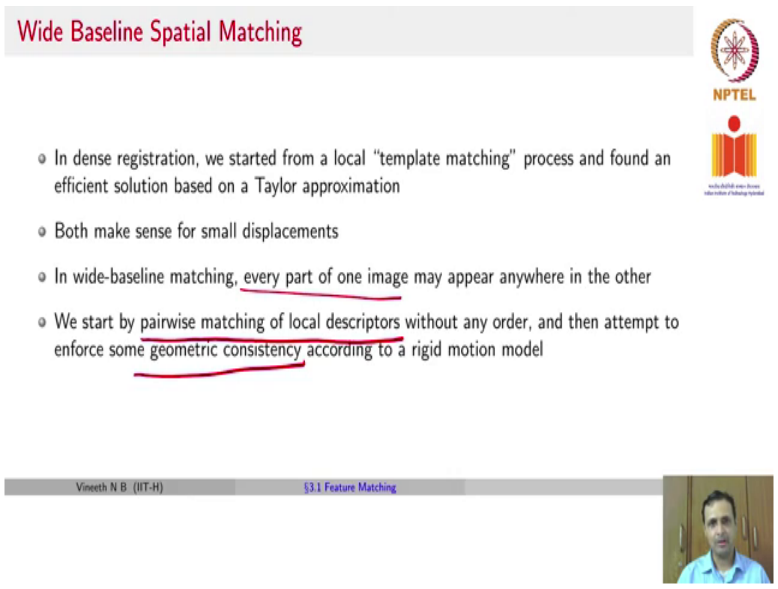
Wide Baseline Spatial Matching

- In dense registration, we started from a local "template matching" process and found an efficient solution based on a Taylor approximation
- Both make sense for small displacements
- In wide-baseline matching, every part of one image may appear anywhere in the other
- We start by pairwise matching of local descriptors without any order, and then attempt to enforce some geometric consistency according to a rigid motion model

Vineeth N B (IIT-H) §3.1 Feature Matching

And for that we move into what is known as white baseline spatial matching. In white baseline spatial matching there is a difference from the dense registration just to repeat again, in dense registration we started from a very local template matching process and we found an efficient solution based on a Taylor approximation. Both of these make sense only when you have small displacements.

(Refer Slide Time: 11:37)



Wide Baseline Spatial Matching

- In dense registration, we started from a local "template matching" process and found an efficient solution based on a Taylor approximation
- Both make sense for small displacements
- In wide-baseline matching, every part of one image may appear anywhere in the other
- We start by pairwise matching of local descriptors without any order, and then attempt to enforce some geometric consistency according to a rigid motion model

Vineeth N B (IIT-H) §3.1 Feature Matching

But in wide baseline spatial matching, we are going to assume now that every part of one image may appear in any part of the second image. It's no longer a small displacement; you could have a corner point that was lying in the top left of one image and the bottom right of the other image and we still want to be able to match these points across these images.

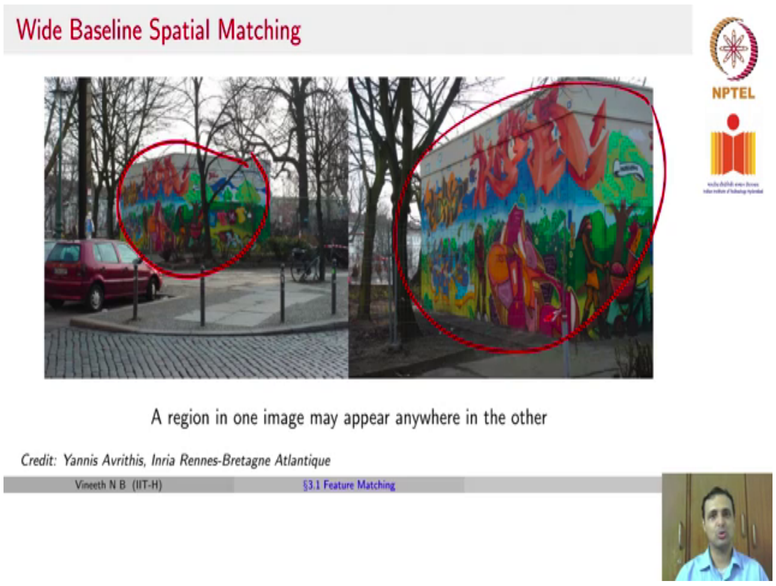
How do we go about this? The key intuition is going to be that we start by pairwise matching of local descriptors. So, you have a bunch of key points in image one, and a bunch of key points in image two. For each of these key points you have a descriptor, you now match those descriptors with the descriptors of all key points in the second image.

Wherever you have the best match of descriptors you are going to say that this point in image one is likely to match with this particular point a certain point in image two, and these points could be at completely different coordinate positions in the first image and the second image.

So, we start by pairwise matching of local descriptors with no other order imposed and then we try to enforce some kind of geometry consistency according to a rigid motion model. So, we know that in the real world you can perhaps, rotate an image, translate or move your camera on pan your camera, you can probably zoom in and zoom out, there are a few different transformations that is generally possible all of them is what we mean as a rigid motion model or geometric consistency.

So, we are going to assume a particular model that could have taken place and using these pairwise matching of local descriptors we are going to try to solve what would be the parameters of the transformation between the two images. This is going to be the key idea, but we will now talk about how we actually go about doing this.

(Refer Slide Time: 13:29)



Wide Baseline Spatial Matching

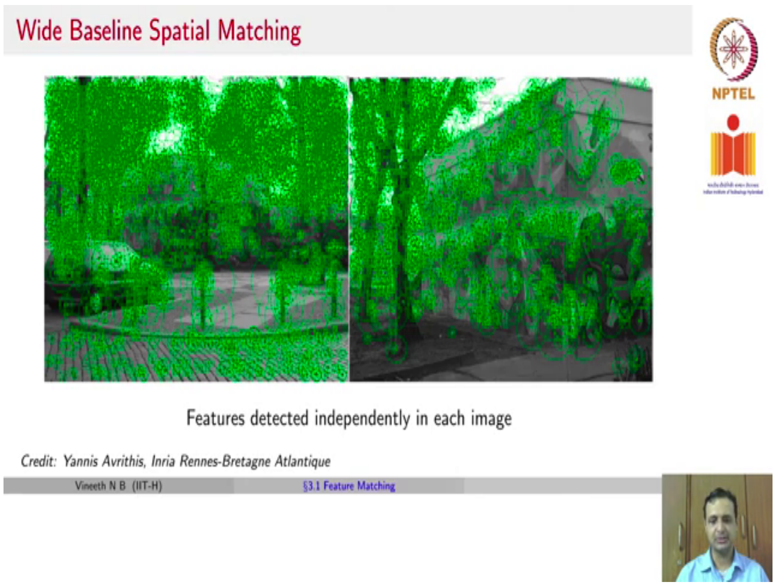
A region in one image may appear anywhere in the other

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

So, once again, in wide baseline spatial matching you could have two images such as this where a region in one image may appear anywhere in the other. There could be a zoom in zoom out, it could be a different angle or it could be translated by some bit any of those could happen when we try to do this kind of magic.

(Refer Slide Time: 13:52)



Wide Baseline Spatial Matching

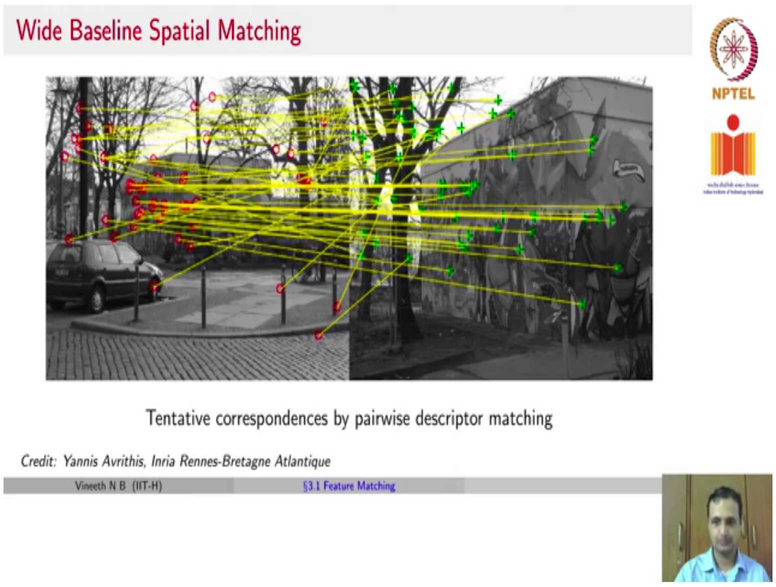
Features detected independently in each image

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

So, as we already said, we first independently detect features in both these images, so each of them are different features that you see across these images.

(Refer Slide Time: 14:04)



Wide Baseline Spatial Matching

Tentative correspondences by pairwise descriptor matching

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

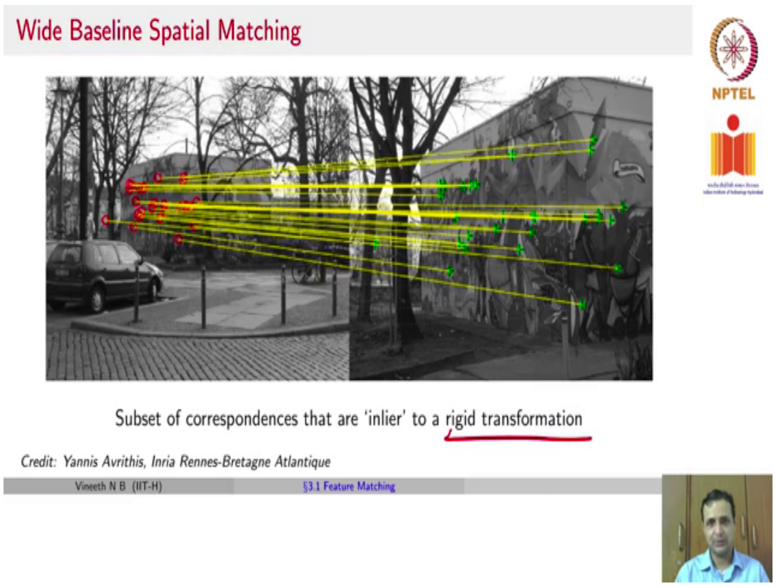
Vineeth N B (IIT-H) §3.1 Feature Matching

Then we try to do a pairwise descriptor matching for each detected feature, we can come up with a descriptor such as histogram of oriented gradients, or local binary patterns, or the variant of histogram of oriented gradients that SIFT uses, so on and so forth. You try to do a pairwise matching of the descriptors between the key points on these two images.

Clearly, when there is a lot of change between two images it is not necessary that every key point will match with some key points on the other. In this particular case you can see that the car does not even exist in the second image. So, any key points on the car would not have an equivalent match on the second image, which is perfectly fine with us. So, only a subset of features that were detected in the first step would actually lead to matches in both cases.

In both these cases, even in the first image, only a subset of features will match with the second image. Even among all the features detected in the second image, only a subset of features from the second image would match with features in the first image. How do you match? Once you get the descriptors in terms of vectors you can simply take the Euclidean distance to match, you can use other kinds of distances too, but you can simply use the Euclidean distance between the descriptors of the features in both these images to be able to match.

(Refer Slide Time: 15:32)



Wide Baseline Spatial Matching

Subset of correspondences that are 'inlier' to a rigid transformation

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

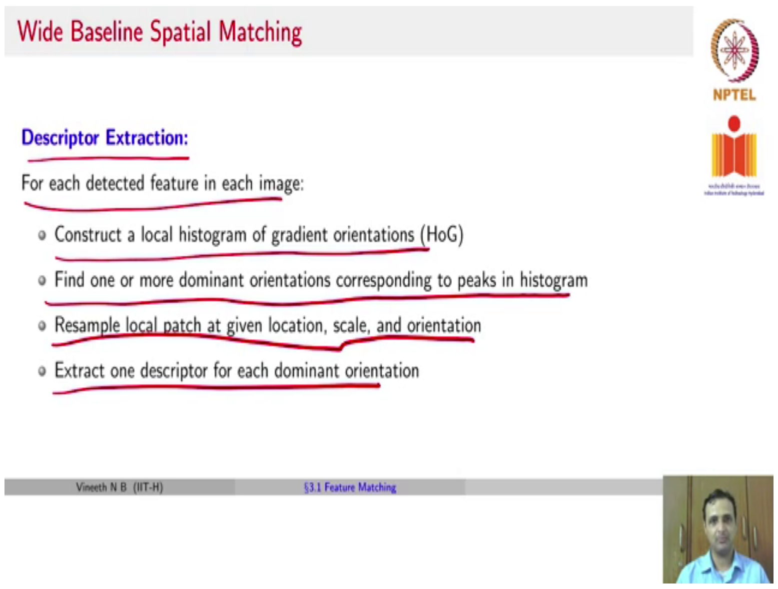
Vineeth N B (IIT-H) §3.1 Feature Matching

So, once you get these tentative correspondences, we try to assume a certain geometric model. For example, we can say that we know that in our particular domain only a translation is possible or only a translation and rotation is possible because in my camera there is no zoom in or zoom out, it could happen.

So, if you knew what were the conditions under which a particular capture was taken, so you know what could be the transformation that could have taken place between the first image and the second image or you assumed a certain rigid transformation, and you find among those pairwise correspondences that we saw on the previous slide, which of them would hold true to this kind of rigid transformation that I assume?

We will come a bit later in this lecture as to how that rigid transformation is represented and how we find points that are in line. We will come back to this in a few slides from now, but this is the overall idea. So among all of those correspondences you narrow down to a few which satisfy your hypothesis of what could have happened.

(Refer Slide Time: 16:38)



Wide Baseline Spatial Matching

Descriptor Extraction:

For each detected feature in each image:

- Construct a local histogram of gradient orientations (HoG)
- Find one or more dominant orientations corresponding to peaks in histogram
- Resample local patch at given location, scale, and orientation
- Extract one descriptor for each dominant orientation

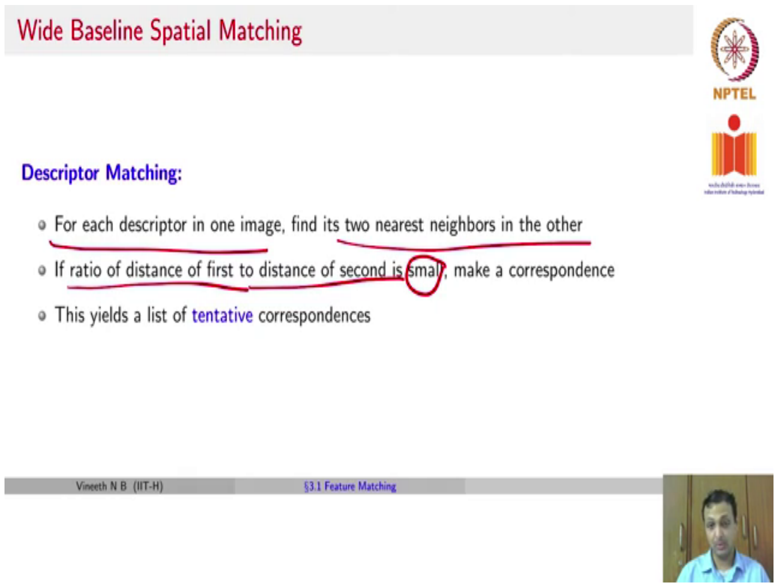
Vineeth N B (IIT-H) §3.1 Feature Matching

And then once you get that subset of inlier correspondences you can simply match and find the transformation and align one image on top of the other. So, let's talk about this in more detail over the next few slides. So, we first extract descriptors from the key points in each image, so for each detected feature you could do something like construct a local histogram of gradient orientations, you could do other kinds of things too, this is just an example, you find one or more dominant orientations corresponding to the peaks of the histogram. Remember, in SIFT, we talked about finding the orientation of each key point, that's what we're talking about here.

At that point, you may want to resample the local patch at a given location, scale or orientation, based on what feature detector you used. You could have a scale for that particular key point. So, you could have a location for that key point, you could have a scale, you could also have an orientation so you could resample the local patch. When we say resample if it's a rotated patch you may want to resample it by doing some interpolation so on and so forth.

You can, you resample the local patch and then you find a descriptor for each dominant orientation. That gives you your descriptors. Remember, again, just like how we spoke for sift, you could take multiple descriptors for each corner key point if there are different orientations that are dominant. We talked about this earlier too.

(Refer Slide Time: 18:06)



Wide Baseline Spatial Matching

Descriptor Matching:

- For each descriptor in one image, find its two nearest neighbors in the other
- If ratio of distance of first to distance of second is small, make a correspondence
- This yields a list of tentative correspondences

Vineeth N B (IIT-H) §3.1 Feature Matching

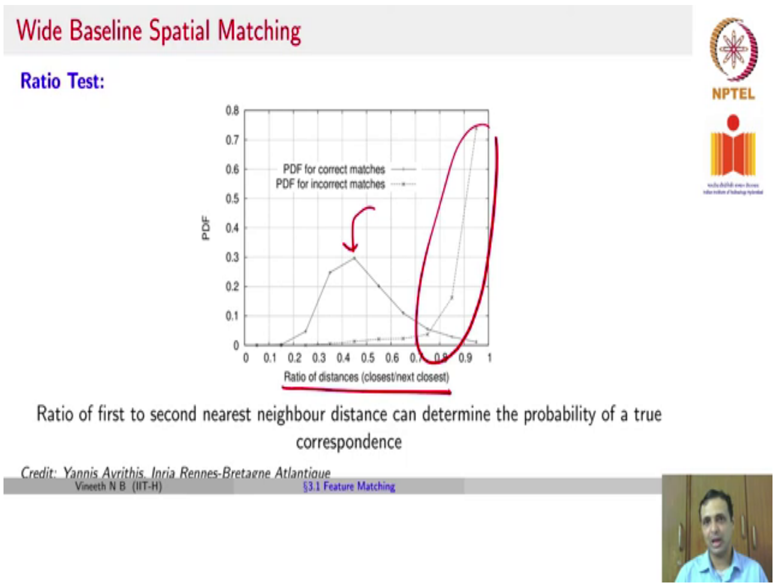
Okay, now at the end of that step we have a bunch of descriptors in image one, a bunch of descriptors in image two. How do we go forward? For each descriptor in one image we find its two nearest neighbors in the next image. Why two? It's just one method you can also take other kinds of nearest neighbors if you like.

If you in this method we take two nearest neighbors and we then evaluate the ratio of the distance of the first to the distance of the second, so you have a distance between the descriptor the first image to the first match in the second image and the distance of the descriptor from the first image the same descriptor to the second closest match. If the ratio between the two is one, which means both are good matches.

If in one case the distance is very low, but in the second case the distance is very high you perhaps now know which of them is significantly closer, you can threshold to find out which of them are strong matches. So, whenever this ratio is small you know that you have found a very strong match, because the second year distance is very far away that is what this ratio would measure.

So, whenever you have a strong match, you are going to consider that a correspondence, and then after you do all these pairwise matchings you have a list of correspondences between image one and image two. What do we mean by correspondences? We are simply saying that descriptor D1 in image one corresponds to descriptor D10 in image two, something like that. You can just write out a table of correspondences between these between the descriptors of these two images.

(Refer Slide Time: 19:52)



Wide Baseline Spatial Matching

Ratio Test:

Ratio of first to second nearest neighbour distance can determine the probability of a true correspondence

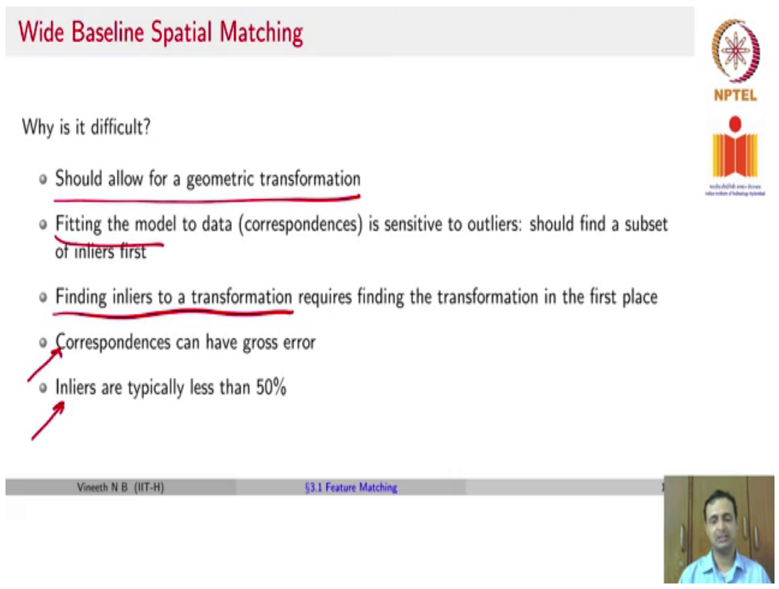
Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

Okay, here is an illustration of the ratio test. So, you can see here that for correct matches you can see that the ratio of distances forms this kind of a distribution, it is much smaller. Whereas for incorrect matches the ratio keeps going up and further towards one, in the incorrect matches the ratio is going to be close to one, which means the first match is as good as the second match.

Then you are not very sure whether the match is strong enough. When the first match distance is much lesser than the second match's distance you know that you are doing a good job. You can, as I said, also expand this to more nearest neighbors and expand the concept of ratio if you would like to get a better idea of the robustness of this match.

(Refer Slide Time: 20:40)



Wide Baseline Spatial Matching

Why is it difficult?

- Should allow for a geometric transformation
- Fitting the model to data (correspondences) is sensitive to outliers: should find a subset of inliers first
- Finding inliers to a transformation requires finding the transformation in the first place
- Correspondences can have gross error
- Inliers are typically less than 50%

Vineeth N B (IIT-H) §3.1 Feature Matching

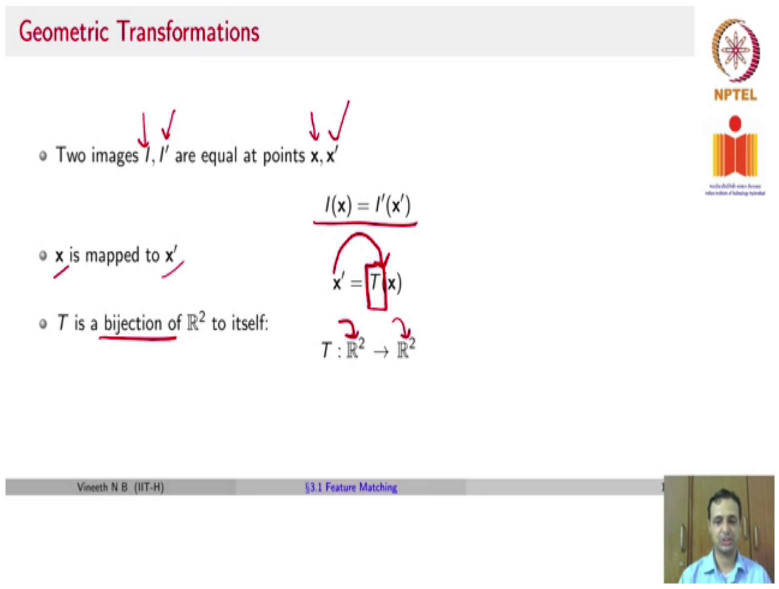
Once you have identified these good matches we move on and then try to estimate which of them are inliers with the rigid transformation that we assumed. Before we go there let us first try to find out why this is a difficult process by itself? Okay, we have so far spoken about a few steps. Firstly, we have to choose key points or these kinds of correspondences which allow for a geometric transformation that may not be trivial in several images.

Fitting the model or the geometric transformation to the correspondences that we have found, could be sensitive to outliers. It is possible just by chance that your correspondence could have been wrong because in the new image maybe there was a newer artifact that came in which was not there in the first image, which ended up matching the key point of the first image in that particular case could simply be an outlier match, which could make fitting your geometric model a little harder.

To find inliers to a transformation you first of all need to find a transformation. So far, I kept telling you that you can assume a transformation, but assuming a transformation is not trivial. You need domain knowledge, you may perhaps need to do something more to be able to find out what should be the transformation in the first place before fitting these correspondences towards transformation.

In certain cases such as outliers, correspondences can also have a grace error. It is likely that in certain cases the correspondences can lead to mistakes; it is possible that HOG may not have been the right descriptor to get correspondences for certain features, so you could have errors in these kinds of cases. An inliers are often less than 50 percent of your total correspondences. Generally, even lesser, but they're typically less than 50 percent. So, which means the number of inliers that you are going to be left with at the end is very few that you actually can play with.

(Refer Slide Time: 22:44)



Geometric Transformations

- Two images I, I' are equal at points $\mathbf{x}, \mathbf{x}'$ $\quad I(\mathbf{x}) = I'(\mathbf{x}')$
- $\mathbf{x}$ is mapped to $\mathbf{x}'$ $\quad \mathbf{x}' = \boxed{T}(\mathbf{x})$
- T is a bijection of $\mathbb{R}^2$ to itself: $\quad T : \mathbb{R}^2 \to \mathbb{R}^2$

Vineeth N B (IIT-H) §3.1 Feature Matching

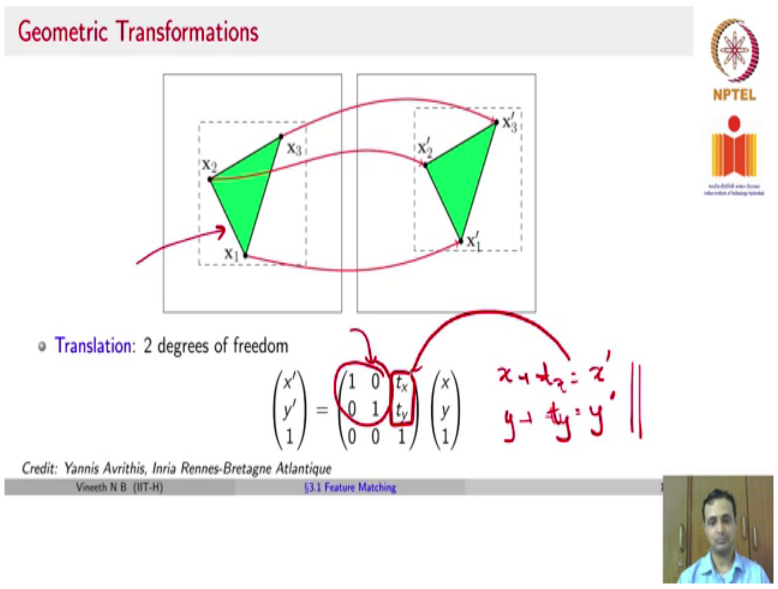
So for the next part, to be able to understand how do you match these correspondences to the rigid transformation model? Let us actually talk about what we mean by geometric transformations here. What do we mean by rigid transformations here, and then we will come back and then try to align the correspondences to one particular transformation.

Given two images, i and i prime are equal at two data points x and x prime, we know that i of x is equal to i prime at x prime. This simply says that across these two images you could map the point x to the point x prime in the second image or rather you can write this as x prime is some transformation of x.

We got the point x prime by perhaps rotating the first image or by translating the first image or by zooming into the first dimension. We are going to refer to all those kinds of transformations such as rotation, translation, and scaling as a transformation matrix T. And what does T do? T is an operation that takes you from a vector in R^2 and gives you another vector in R^2. Remember any matrix can be looked at as a transformation in this perspective.

So, given a point at a coordinate location $(x,\ y)$ in image 1, the transformation matrix T takes you to another point $(x',\ y')$ in your second image. And this transformation is going to be a bijection, which means it's a one to one match between image one and image two. Every point in image one matches to only one point in image two and every point in image two matches to only one point in image one, it's going to be a bijection.

(Refer Slide Time: 24:35)



Geometric Transformations

- Translation: 2 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$x + t_x = x'$
$y + t_y = y'$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

Let us try to study what T looks like. T is a transformation, we said it's a matrix. So, for a certain set of common transformations, T is fairly well defined, especially rigid body transformations, and this has been extensively studied, especially in the graphics-based vision that we talked about in the first lecture.

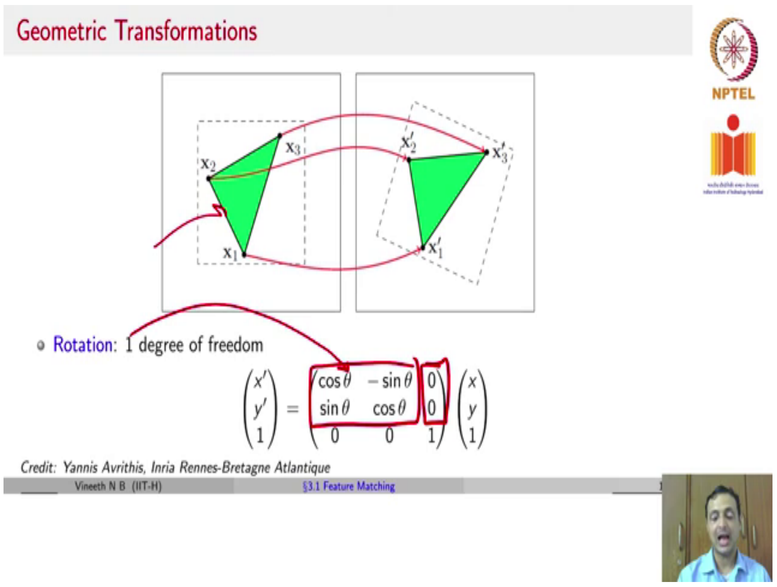
So, we will briefly talk about this now to understand how the matching is done. So, suppose you have this green triangle in the first image, and you translate this rather you just move it slightly along the $x - axis$ or $y - axis$ or on both these axes it moves to a slightly different location in the second image.

In this particular case, you would define the transformation to be given by a 3 x 3 matrix, which is given a 1001, which is the top $2x2$ of this matrix, then you have t_x, t_y which corresponds to the translation along the $x - axis$ and translation along the $y - axis$. If you work this out and whenever you apply this transformation on $(x,\ y,\ 1)$. 1 is simply used as a normalized coordinate to represent this transformation we get an outcome which is $\hat{x}$, $\hat{y}$, 1.

Why so? Let us analyze this a bit carefully. It's simply a matrix vector transformation. If you simply did a matrix vector translation you will actually see that this is just another way of writing a system of equations and the system of equations says $x' = x + t_x$. Similarly, you have $y' = y + t_y$. The third one doesn't matter, you are just going to have 1 is equal to 1, it does not matter, but this is exactly what we're looking for.

This is just another way, there is just a system of equations and we are simply writing the system of equations in terms of a matrix vector transformation, a matrix transformation on a vector to give you another vector. This is a translation. Let us see one more.

(Refer Slide Time: 26:39)



Geometric Transformations

- Rotation: 1 degree of freedom

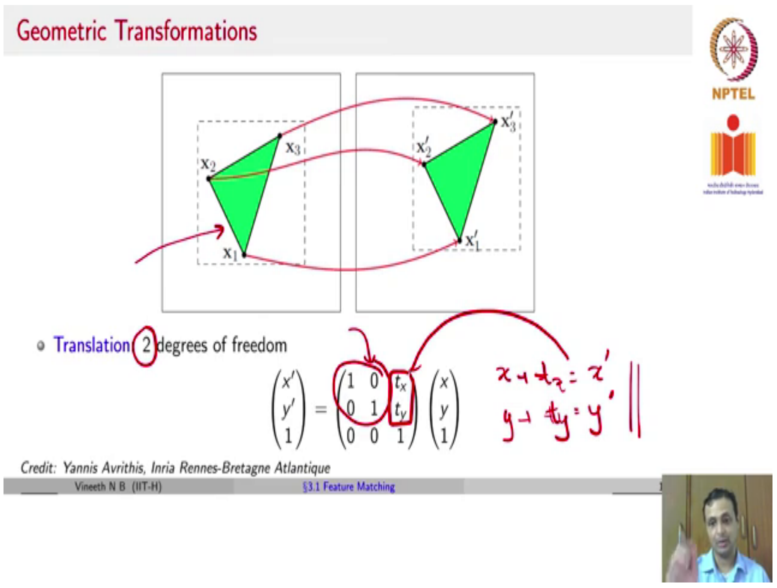
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

If you took rotation, this green triangle is now simply rotated. There is no translation, it is only rotated. You can see that zero zeros are put here for the translation, which means there is zero translation, but there is rotation. And in this case it is given by $cos(\theta)$, $-sin(\theta)$, $sin(\theta)$, $cos(\theta)$ in the upper 2x2 this 3x3 matrix. I will let you look at this more carefully it's a simple expansion again you would have $x' = xcos(\theta) - ysin(\theta)$ and $x' = xsin(\theta) + ycos(\theta)$ that simply represents the new coordinates based on your rotation angle θ.

(Refer Slide Time: 27:30)



Geometric Transformations

- Translation: 2 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

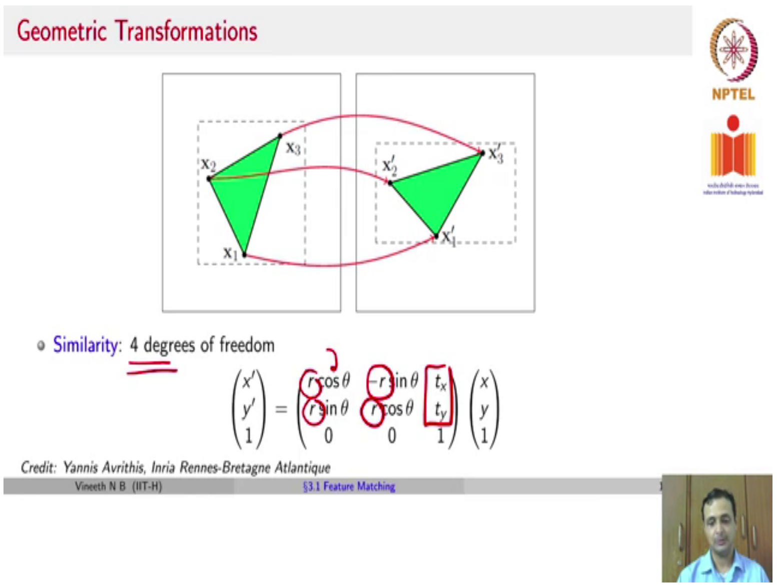
$x + t_x = x'$
$y + t_y = y'$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

So, you can see here that if you went back to the previous slide in translation there are two degrees of freedom t_x and t_y. In rotation, just one degree of freedom is given by the angle theta.

(Refer Slide Time: 27:43)



Geometric Transformations

- Similarity: 4 degrees of freedom

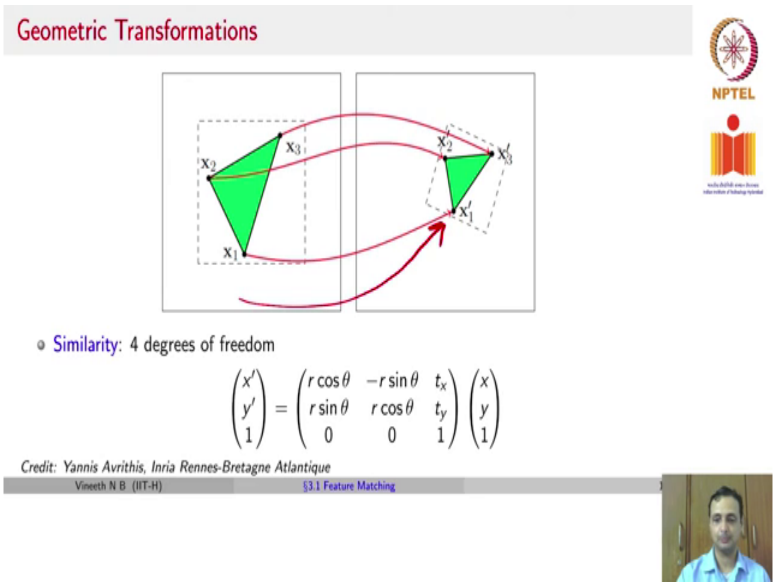
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} r\cos\theta & -r\sin\theta & t_x \\ r\sin\theta & r\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

Another transformation is called the similarity transformation, which has four degrees of freedom, which combines rotation has two degrees of freedom due to translation, but you also have a scaling aspect here which is given by r, which can change the size of the object in the second image. And we see size or scale, remember it would correspond to zoom in or zoom out in terms of the camera parameters. So, now you have r, θ, t_x and t_y four degrees of freedom in this geometric transformation.

(Refer Slide Time: 28:23)



Geometric Transformations

- Similarity: 4 degrees of freedom

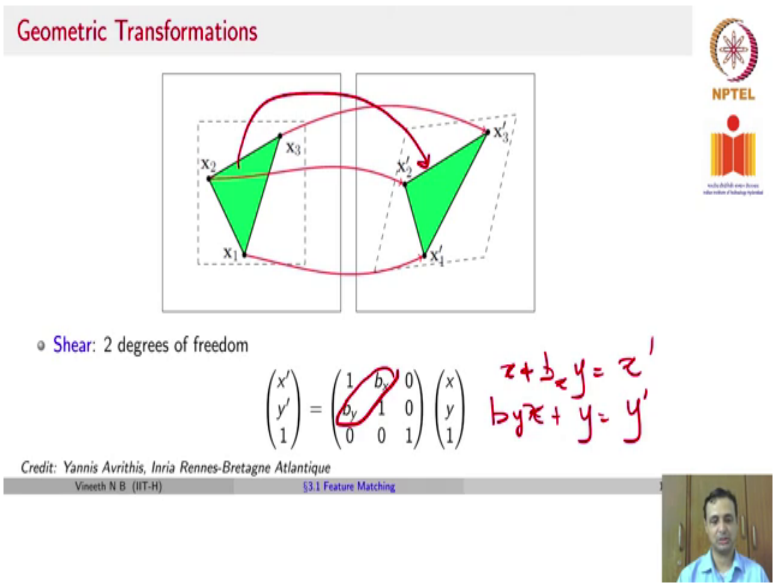
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} r\cos\theta & -r\sin\theta & t_x \\ r\sin\theta & r\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

Going forward, this is another example of a similarity transformation where you can see the zoom out in action where the r has a non-zero value or a non-one value to be able to show a similarity transformation where r is operational.

(Refer Slide Time: 28:42)



Geometric Transformations

- Shear: 2 degrees of freedom

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & b_x & 0 \\ b_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

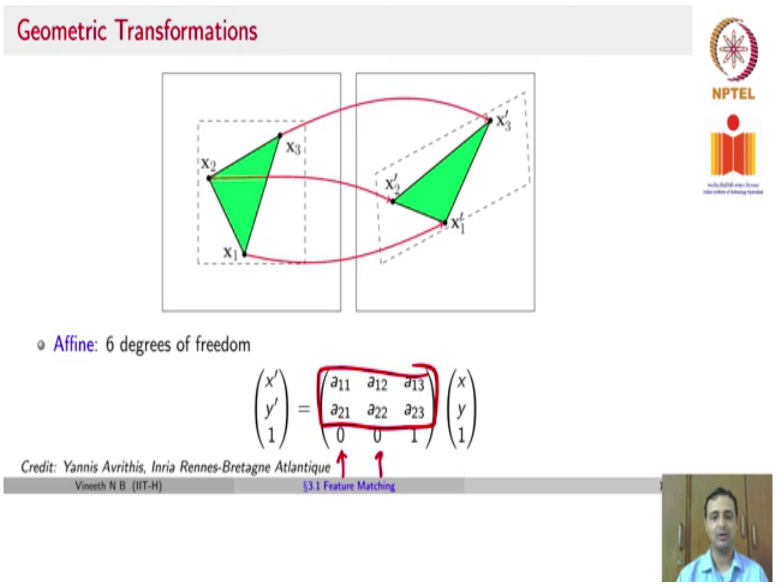
$$x + b_x y = x'$$
$$b_y x + y = y'$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

Another transformation is known as the shear transformation you can see here as to how the triangle gets transformed between image one and image two. This is known as shear where you apply pressure on one of the sides of the triangle and extend it and keep the other side constant. So, this is given by changing just these quantities bx, by in your transformation and the rest of them stay one, so then it is called shear. You can write out the equations of shear as $x + b_x y = x'$, $b_y x + y = y'$. This is simply a linear system of equations and a way of writing the transformation.

(Refer Slide Time: 29:37)



Geometric Transformations

- Affine: 6 degrees of freedom

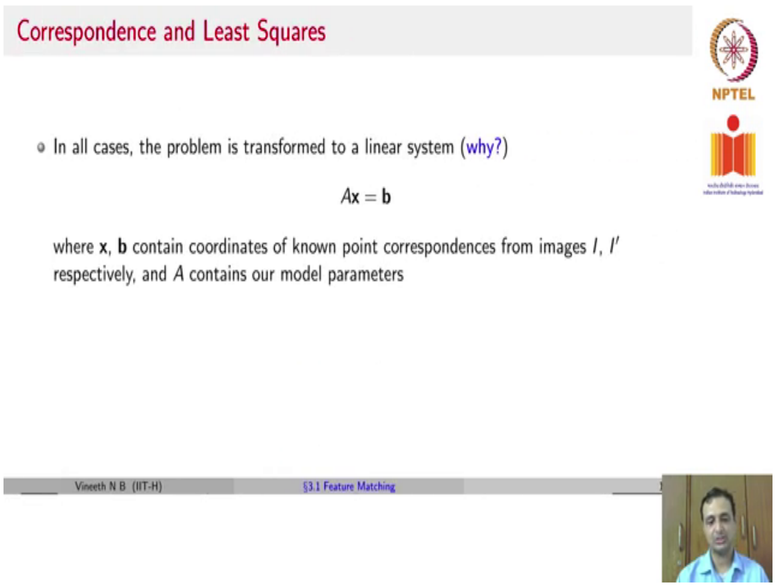
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

Furthermore, a popular transformation known as the affine transformation is given by six degrees of freedom, where you can have values for any of those six spots in your transformation matrix that we spoke about. We are going to stick to these sets of rigid body transformations at this time. There are certain transformations that also use these values at the bottom which are going to projective transformations perspective transformations, we were not going to get into it at this particular point in time, we were going to stick to affine transformations.

(Refer Slide Time: 30:07)



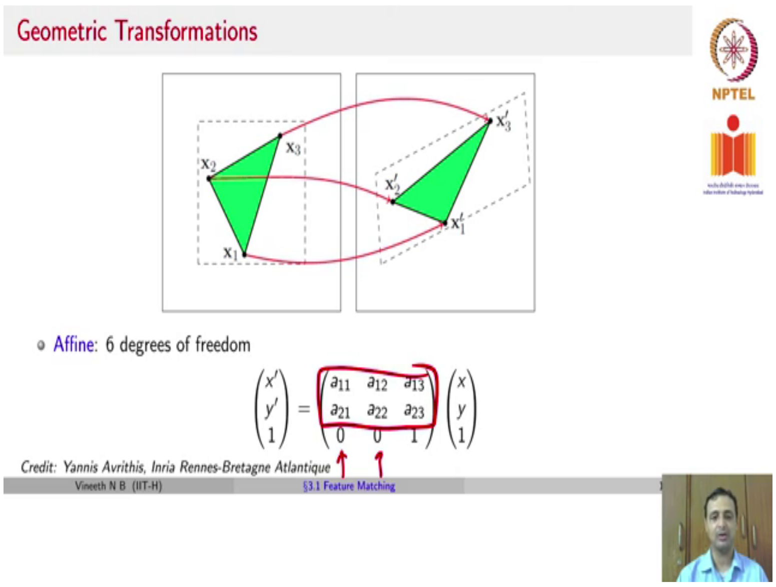
Correspondence and Least Squares

- In all cases, the problem is transformed to a linear system (why?)

$$A\mathbf{x} = \mathbf{b}$$

where $\mathbf{x}$, $\mathbf{b}$ contain coordinates of known point correspondences from images I, I' respectively, and A contains our model parameters

Vineeth N B (IIT-H) §3.1 Feature Matching



Geometric Transformations

- Affine: 6 degrees of freedom

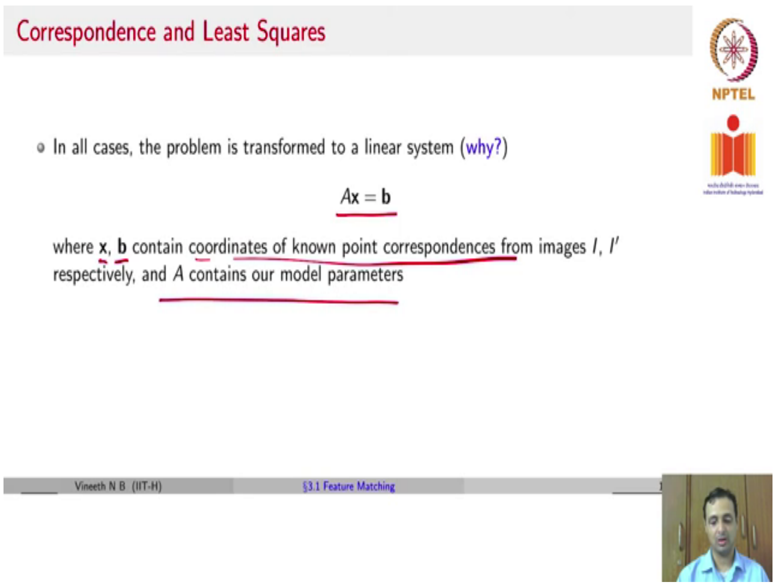
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \boxed{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

So, in all of these cases, as you can see using those tentative correspondences that we get between two images we can find out which x prime matches to x and y in your image one. So, (x', y') in image 2 could be matching with (x, y) in image 1. So, we already have a list of correspondences based on those matching of descriptors. Our job is to find out what are the parameters of this transformation T, that's what we want to look for. Clearly, this is about solving a linear system of equations.

(Refer Slide Time: 30:44)



Correspondence and Least Squares

- In all cases, the problem is transformed to a linear system (why?)

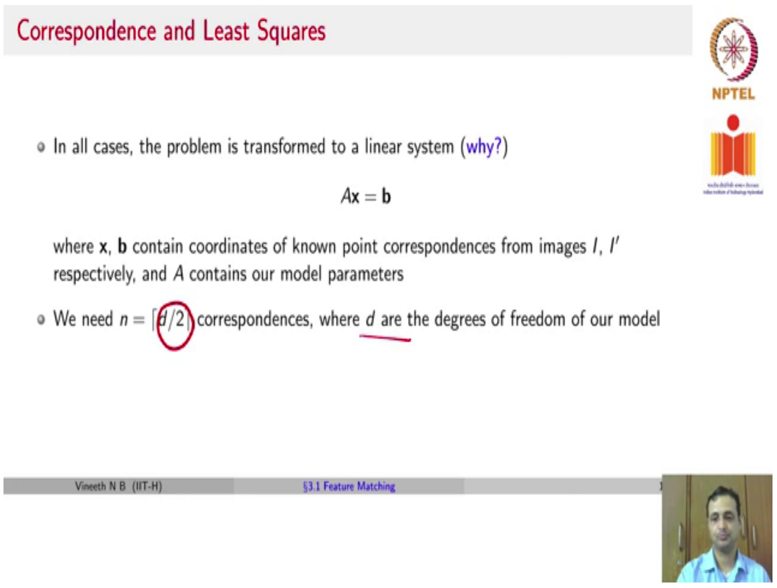
$$A\mathbf{x} = \mathbf{b}$$

where $\mathbf{x}$, $\mathbf{b}$ contain coordinates of known point correspondences from images I, I' respectively, and A contains our model parameters

Vineeth N B (IIT-H) §3.1 Feature Matching

So, we want to solve a linear system, $Ax = b$, where x and b are the coordinates of the known point correspondences from images, I and I' and A contains our model parameters that we want to learn.

(Refer Slide Time: 31:00)


Correspondence and Least Squares

- In all cases, the problem is transformed to a linear system (why?)

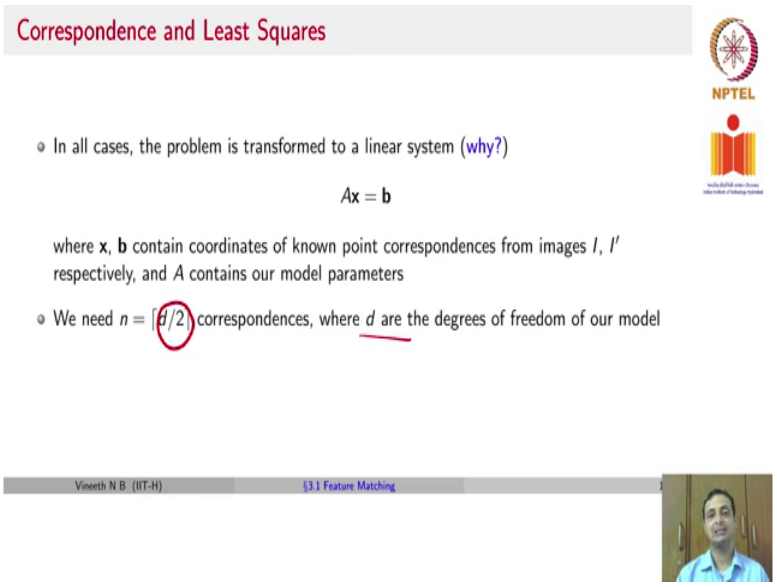
$$A\mathbf{x} = \mathbf{b}$$

where $\mathbf{x}$, $\mathbf{b}$ contain coordinates of known point correspondences from images I, I' respectively, and A contains our model parameters
- We need $n = \lceil d/2 \rceil$ correspondences, where d are the degrees of freedom of our model

Vineeth N B (IIT-H) §3.1 Feature Matching

Ideally speaking, if we had d degrees of freedom in a given transformation you ideally need the ceiling of $d/2$ correspondences. For example, for translation, two degrees of freedom which means you need only one correspondence. If you have one point in one image and another point in the second image you can find both t_x and t_y because you would know how much you moved in x and how much you moved in y. So for a given the d degrees of freedom you need about $d/2$ ceiling as the number of correspondences from descriptors.

(Refer Slide Time: 31:44)


Correspondence and Least Squares

- In all cases, the problem is transformed to a linear system (why?)

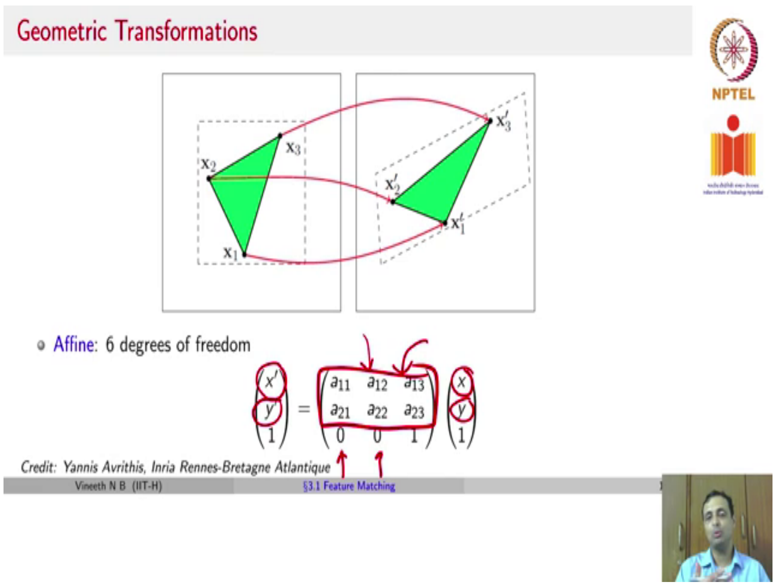
$$A\mathbf{x} = \mathbf{b}$$

where $\mathbf{x}$, $\mathbf{b}$ contain coordinates of known point correspondences from images I, I' respectively, and A contains our model parameters
- We need $n = \lceil d/2 \rceil$ correspondences, where d are the degrees of freedom of our model

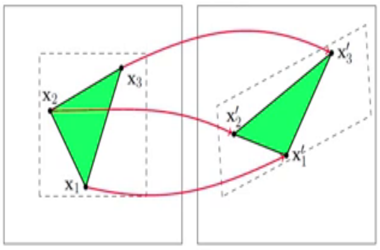
Vineeth N B (IIT-H) §3.1 Feature Matching

Okay, now, how do you solve this right? So we know now that just to recall, repeat what we have talked about so far. We have found key points in each of the images. We found descriptors and then we matched the descriptors between these two images and then based on the nearest neighbor approach we prune those descriptor matches to a few set of descriptor matches which are strong, and among those we now want to find out which of them will suit my rigid body model that I am going to assume for my transformation between the two images.

(Refer Slide Time: 32:29)



Geometric Transformations



- Affine: 6 degrees of freedom

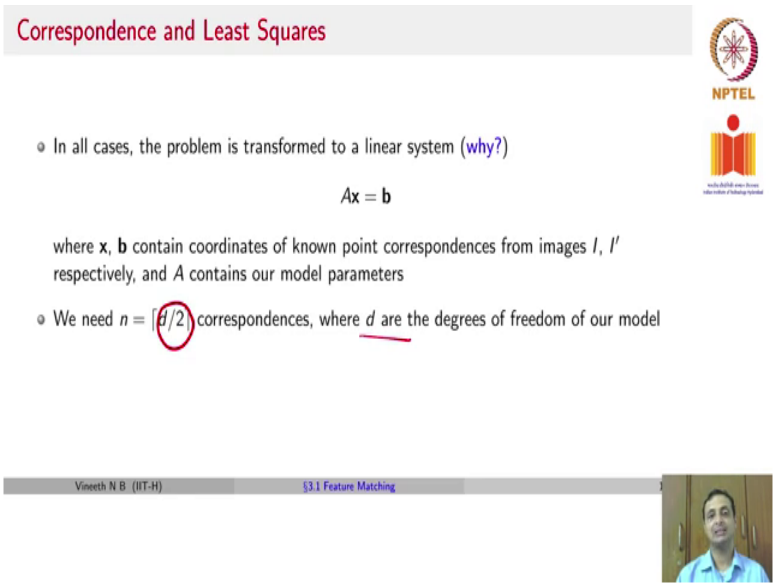
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

So, if I assume and affine transformation norm, using those set of correspondences that I have I ideally have to solve for these six values as my transformation, and once I have solved for these values, I know what was the transformation between these two images, so I can simply place one on one image on top of the other using the transformation again, and be able to blend them and create a panorama.

(Refer Slide Time: 32:50)



Correspondence and Least Squares

- In all cases, the problem is transformed to a linear system (why?)

$$Ax = \mathbf{b}$$

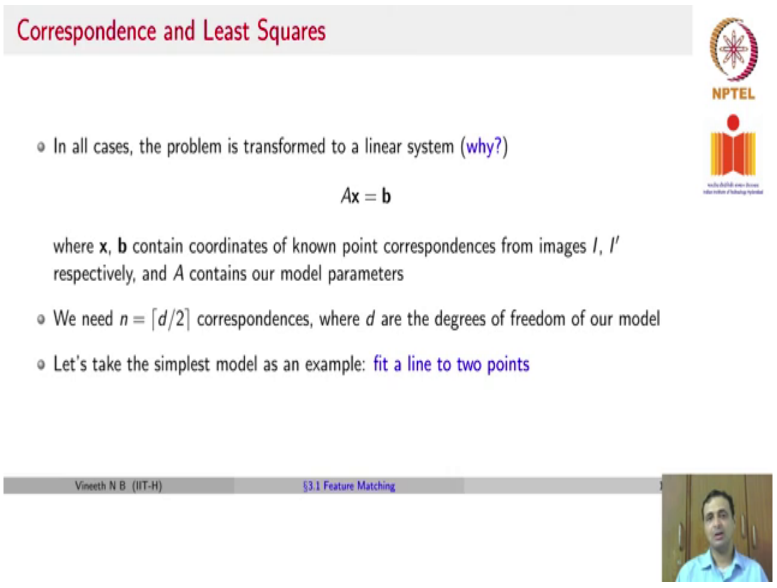
where $\mathbf{x}$, $\mathbf{b}$ contain coordinates of known point correspondences from images I, I' respectively, and A contains our model parameters

- We need $n = \lceil \boxed{d}/2 \rceil$ correspondences, where $\underline{d}$ are the degrees of freedom of our model

Vineeth N B (IIT-H) §3.1 Feature Matching

So, we are left with one task as to how do you actually estimate those parameters given those correspondences?

(Refer Slide Time: 32:54)



Correspondence and Least Squares

- In all cases, the problem is transformed to a linear system (why?)

$$Ax = \mathbf{b}$$

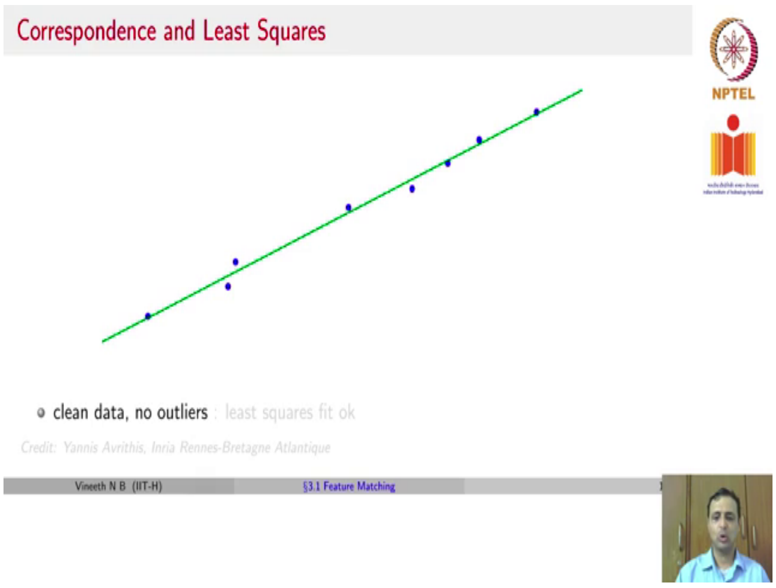
where $\mathbf{x}$, $\mathbf{b}$ contain coordinates of known point correspondences from images I, I' respectively, and A contains our model parameters

- We need $n = \lceil d/2 \rceil$ correspondences, where d are the degrees of freedom of our model
- Let's take the simplest model as an example: fit a line to two points

Vineeth N B (IIT-H) §3.1 Feature Matching

Let's start with the simplest approach that we all know is if you have two points that fit a line, that is the simplest approach that we all know, the simplest model that we can imagine. Let us say the approach that we are going to use, but let us try to describe this further.

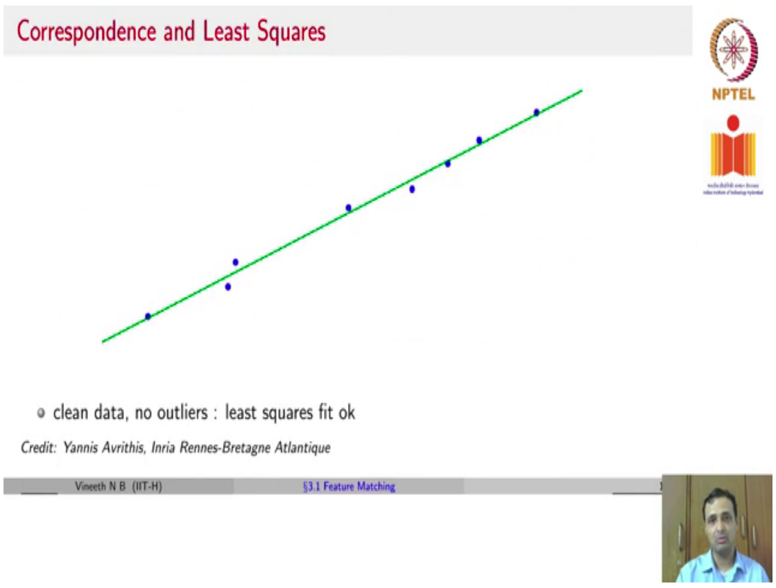
(Refer Slide Time: 33:09)


Correspondence and Least Squares

- clean data, no outliers : least squares fit ok

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching


Correspondence and Least Squares

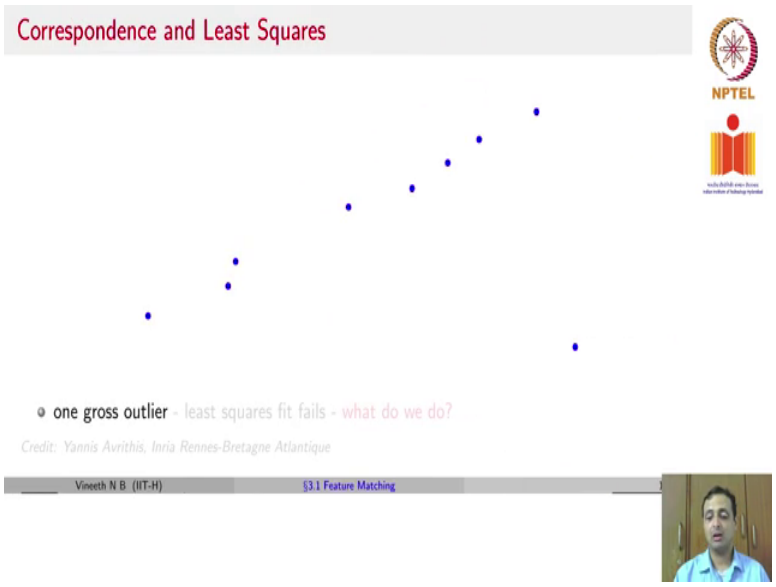
- clean data, no outliers : least squares fit ok

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

So, if you had a least squares approach to fitting correspondences this is what you would have. If you have a bunch of correspondences here this is clean data, not many outliers. The least squares fit would give you a fairly good equation for the line. We're just talking about the transformation in a slightly more abstract sense now, but we will come back and make it clear as to how you really estimate the parameters of the transformation.

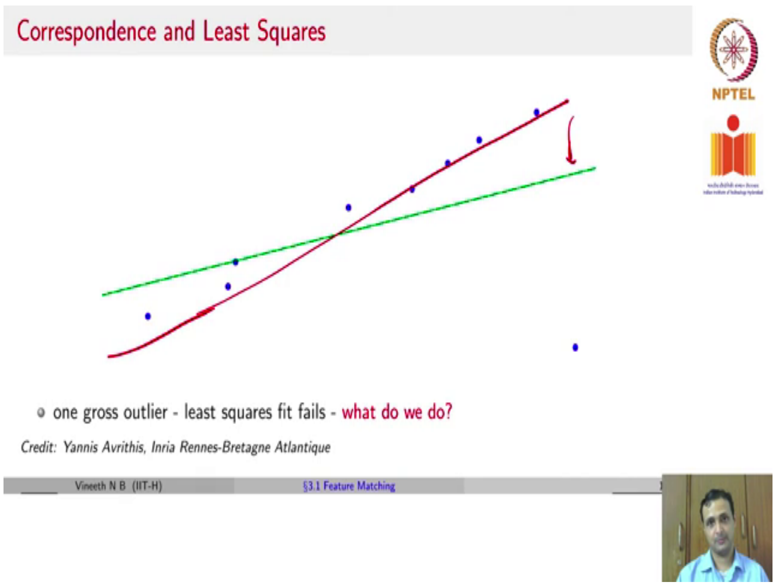
(Refer Slide Time: 33:37)


Correspondence and Least Squares

- one gross outlier - least squares fit fails - what do we do?

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching


Correspondence and Least Squares

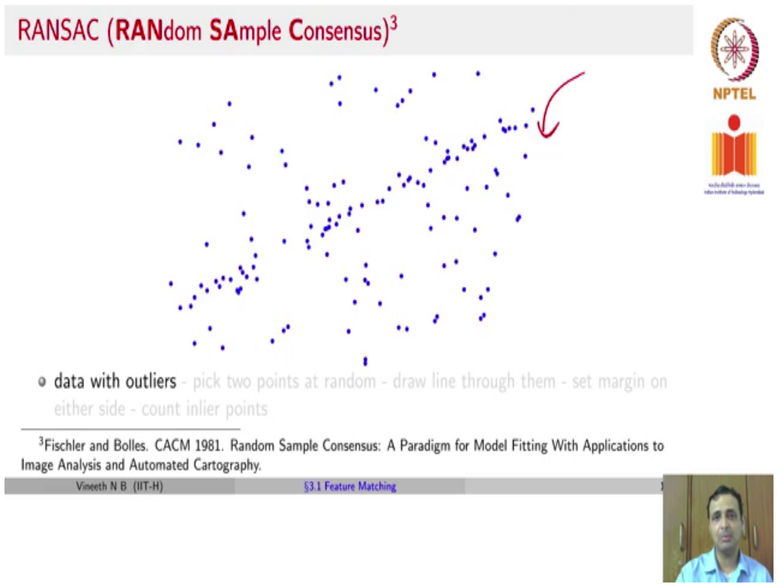
- one gross outlier - least squares fit fails - what do we do?

Credit: Yannis Avrithis, Inria Rennes-Bretagne Atlantique

Vineeth N B (IIT-H) §3.1 Feature Matching

However, if there are outliers in your matches then the least square fit fails and gives a very different answer compared to what should have been the right answer. So, what do we do here?

(Refer Slide Time: 33:58)


RANSAC (RANdom SAmple Consensus)[3]

- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

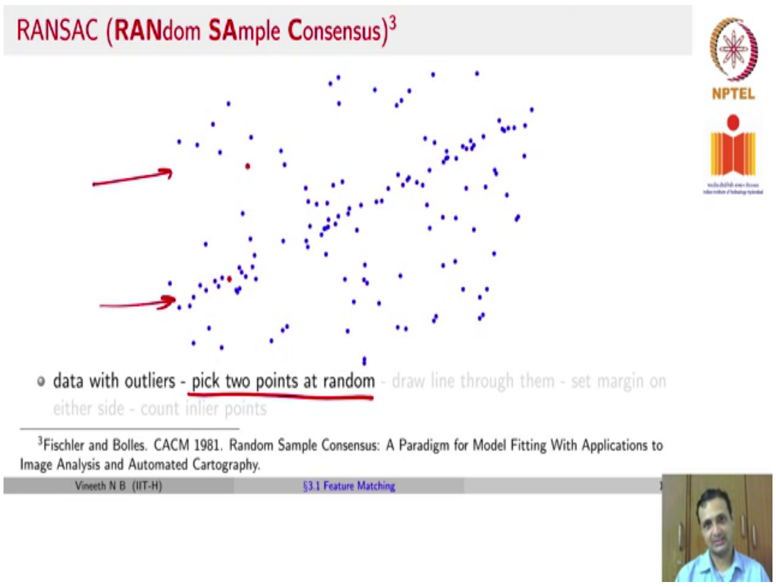
[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

At this point comes to rescue one of the most popular methods for matching features between images, which is known as RANSAC or stands for Random Sample Consensus. A very simple iterative method can be slightly computationally intensive, but works very, very well and has been used for many decades to be able to match correspondences between two images.

For that matter it can match correspondences not just between images between any set of observations in data. So, let us assume these are a set of correspondences that we have. These are data with some outliers and we ideally want to fit a line to this particular model. So, we will talk about fitting a line to data at this point. We will then later come back to the analogy of taking this to fitting an affine transformation or any other rigid body transformation to a set of correspondences in images.

(Refer Slide Time: 35:05)


RANSAC (RANdom SAmple Consensus)[3]

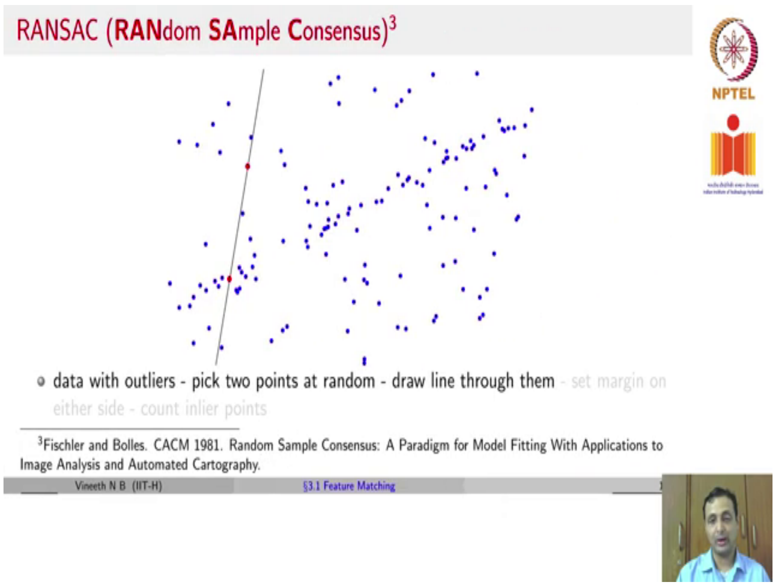
- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

So, if you have such a set of data points and we want to fit a line to this particular set of data, what random sample consensus suggests to us is, you take two points at random, any two points, you can see two points that have been picked in red. We know how to fit a line for two points.

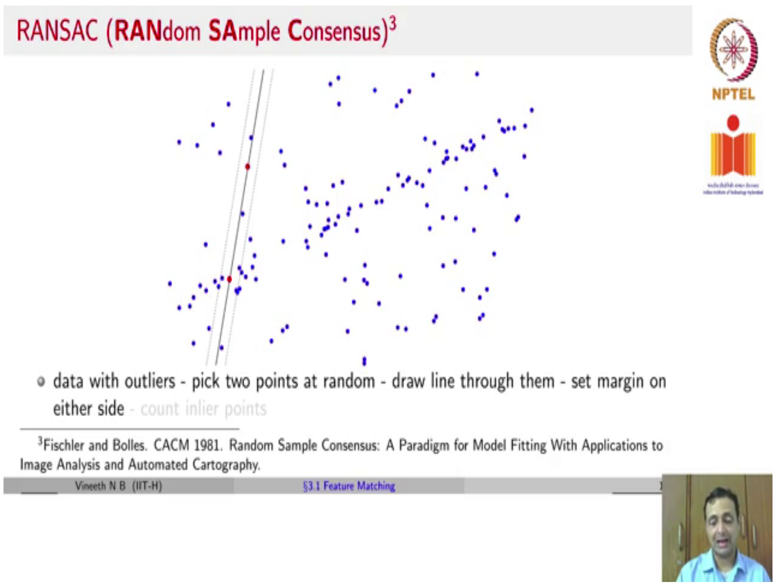
(Refer Slide Time: 35:18)


RANSAC (RANdom SAmple Consensus)[3]

- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

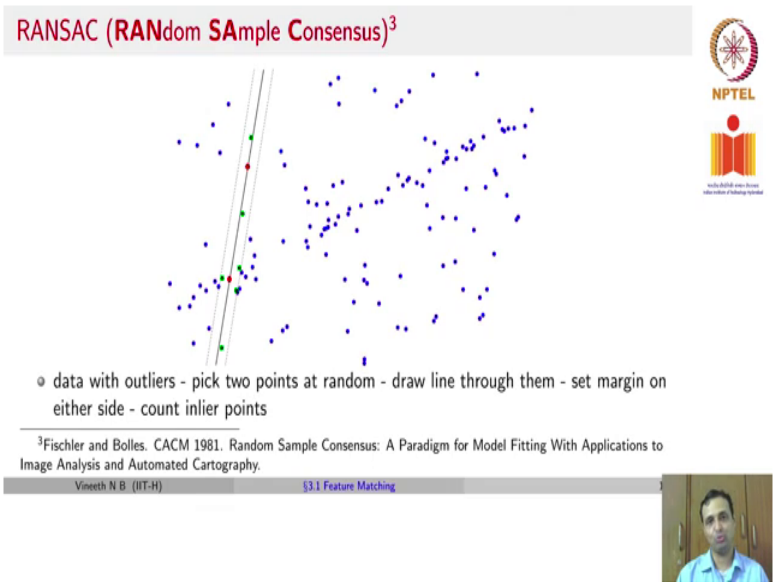


RANSAC (RANdom SAmple Consensus)[3]

- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching



RANSAC (RANdom SAmple Consensus)[3]

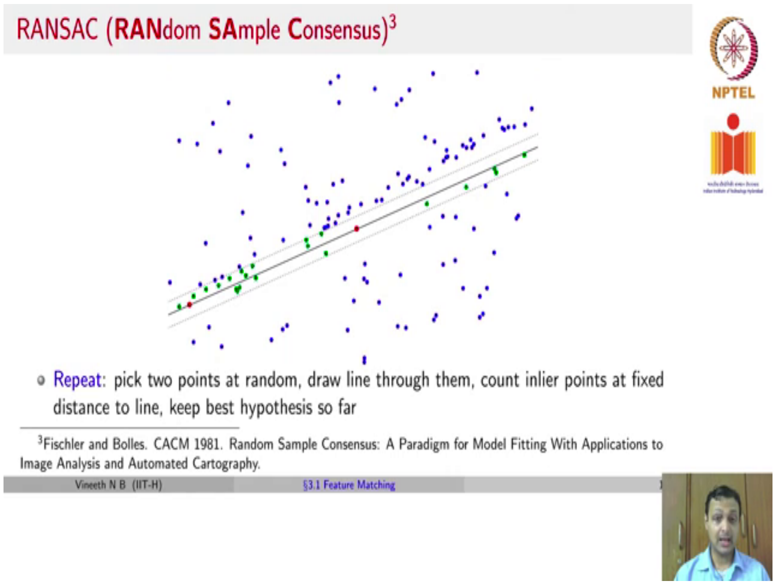
- data with outliers - pick two points at random - draw line through them - set margin on either side - count inlier points

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

So you fit a line for those two points. You take a neighborhood of that line on both sides and see how many points in that neighborhood are inlier points or how many of them correspond to points that you are looking for. You can see here in green that there are about six points that are in the neighborhood of this line that fit on these two points.

(Refer Slide Time: 35:47)



RANSAC (RANdom SAmple Consensus)[3]

- Repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

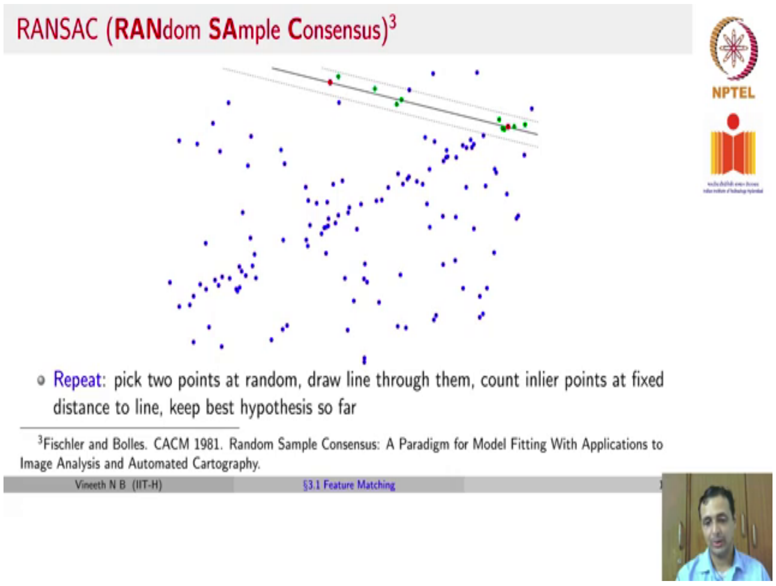
[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

Now, you take another two points randomly. Once again, fit a line through those two points, take a neighborhood and now count how many of four of your given data points lie inside this neighborhood. Now, you can see that this is going to be a fair few, it's going to be about 15, or 20, which is greater than the 6 that we got in the previous random pick that we had.

What does this tell you? That this line may be a better fit than the previous line. So among all the lines that we have seen so far we try to retain the line which has the highest number of inliers, and throw away the previous lines that we came across. This is the line that we have now.

(Refer Slide Time: 36:32)

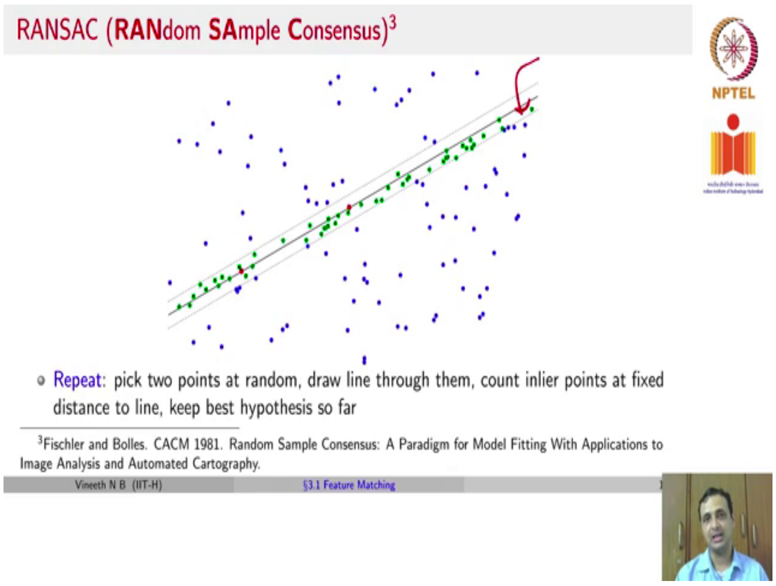


RANSAC (RANdom SAmple Consensus)[3]

- Repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching


RANSAC (**RAN**dom **SA**mple **C**onsensus)[3]

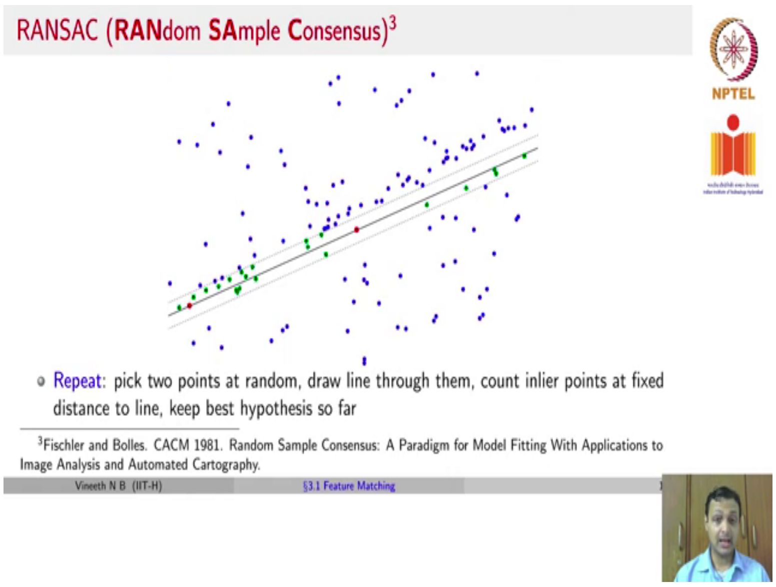
- Repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

You again, repeat this process, pick another two points, pick a line, see how many are in the neighborhood of that line. Once again, you get a lesson number for such a line and you keep repeating this process over and over again, but keep your best hypothesis saved with you. And you see that in this particular example, as you keep repeating this over and over again this line that you have here which goes through a set of random picks is the best possible hypothesis and that is the hypothesis that you are going to go with, that the model that you're going to say that the set of outliers fits.

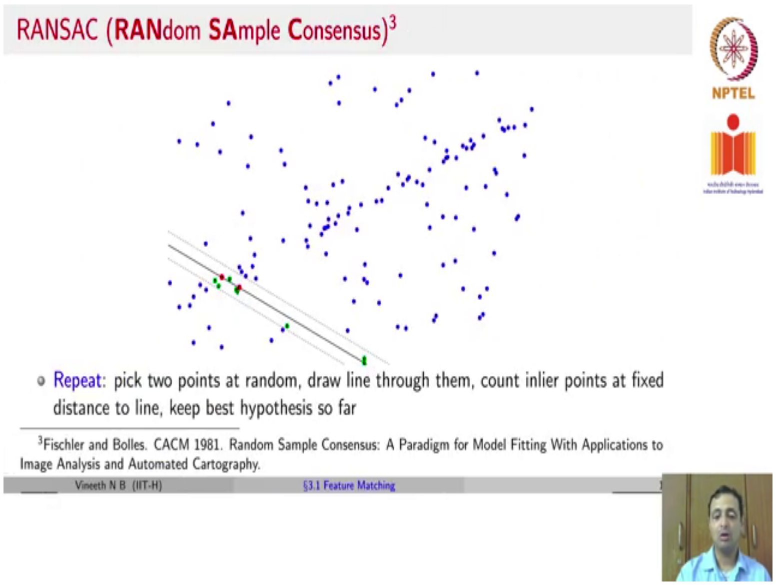
(Refer Slide Time: 37:07)


RANSAC (**RAN**dom **SA**mple **C**onsensus)[3]

- Repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching

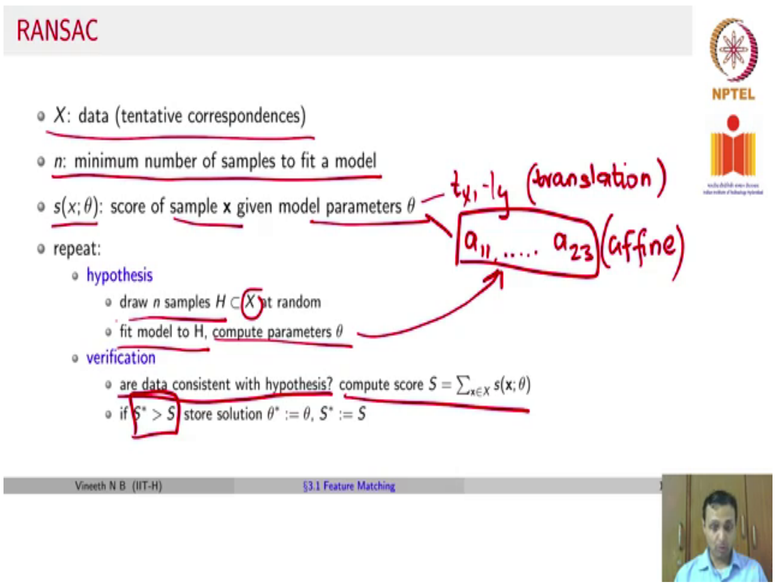


RANSAC (RANdom SAmple Consensus)[3]

- Repeat: pick two points at random, draw line through them, count inlier points at fixed distance to line, keep best hypothesis so far

[3]Fischler and Bolles. CACM 1981. Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography.

Vineeth N B (IIT-H) §3.1 Feature Matching



RANSAC

- X: data (tentative correspondences)
- n: minimum number of samples to fit a model
- $s(x;\theta)$: score of sample **x** given model parameters θ — t_x, t_y (translation); $\boxed{a_{11}, \ldots, a_{23}}$ (affine)
- repeat:
  - hypothesis
    - draw n samples $H \subset X$ at random
    - fit model to H, compute parameters θ
  - verification
    - are data consistent with hypothesis? compute score $S = \sum_{x \in X} s(\mathbf{x};\theta)$
    - if $\boxed{S^* > S}$ store solution $\theta^* := \theta$, $S^* := S$

Vineeth N B (IIT-H) §3.1 Feature Matching

So here are other examples of lines that you can draw. And in all of those cases, you know that the line that you had in the middle has the highest number of inliers in this neighborhood. And that's the one that you will go with. Lets try to write out RANSAC more formally in terms of an algorithm.

So, given a set of tentative correspondences now we are looking at the correspondences that we started with so you have a set of correspondences which is your data. Let's look at n, which is the minimum number of samples to fit a model. Once again, remember, if you are trying to fit a model for rotation it has one degree of freedom, so you at least need one set of correspondences to be able to solve the problem that is what that one set of correspondences is what we are referring to as n here.

And let us assume that in each of these cases you are going to have a score $s(x;\ \theta)$, which gives us the score of sample x given model parameters, θ. θ is just the model parameters that we are trying to learn. If it's rotation you're going to have just one parameter θ. If it's translation, then theta would be t_x, t_y.

If it's, this is for translation. If it's for affine we already saw that you are going to have a_{11}, all the way to a_{23}, 6 six such values, that is going to be affine, so on and so forth. Okay, those are your model parameters that you want to learn. So, what RANSAC says is, you start with the hypothesis verify the hypothesis, and keep iterating and storing the best hypothesis so far.

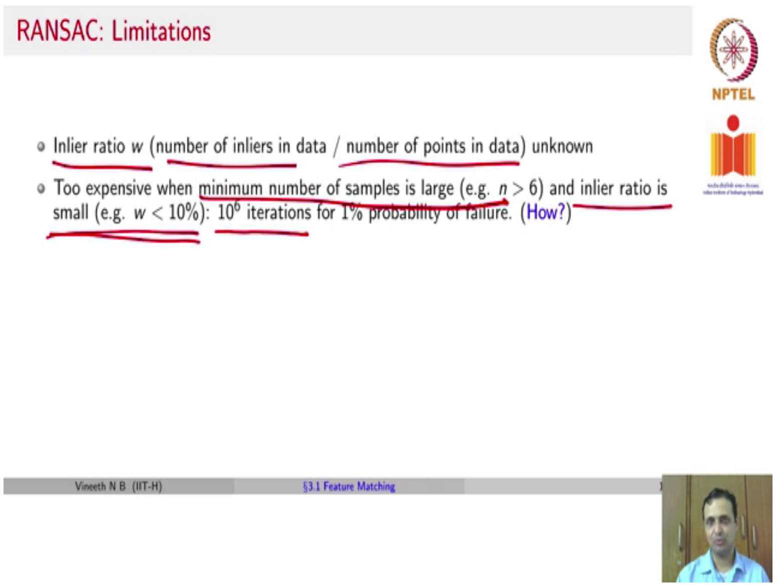
So, you draw n samples from x at random. Let's call that set of n samples as H. You fit the model to H and compute the parameters. If your data are consistent with this hypothesis, most of your data are consistent with this hypothesis you compute a score. If this score was better than the earlier score you make the new hypothesis the best hypothesis, and you keep repeating this process.

Let's talk about this in the context of matching features. So, in this case let us say you drew about say two different or let's take an affine transformation, which has six three parameters, so you need about three sets of correspondences to be able to get a good match between these two, so you draw three pairs of correspondences, the best correspondences that you have, you fit an affine transformation model, which means by fitting you mean you can actually solve for $a_{11}$to a_{23} through using a linear system of equations.

Once you solve for using a linear system of equations using at least regression fit you will get a bunch of values $a_{11}$to a_{23}. Now, you try to see among all your correspondences, which of them would stay within an ϵ neighborhood of this particular transformation. Rather, you take all the points from image 1, which you were considering in your inliers which you are considering in your tentative correspondences, apply this transformation, the transformation for which you now have $a_{11}$to a_{23} values, you would see where those points go to in image 2.

If for many of your correspondences these matches lie within an ϵ neighborhood you are going to consider that a good match has a good number of inliers and the best hypothesis so far. And you just keep repeating this by randomly drawing three, three correspondences each time in case of an affine transformation. For a translation transformation you just have one correspondence that you have to keep drawing to be able to fit these models. That's the main idea of RANSAC.

(Refer Slide Time: 40:54)



RANSAC: Limitations

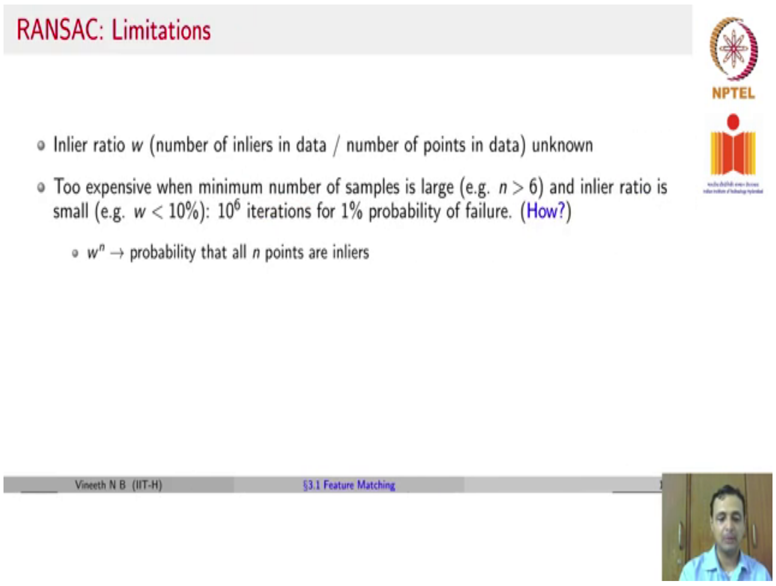
- Inlier ratio w (number of inliers in data / number of points in data) unknown
- Too expensive when minimum number of samples is large (e.g. $n > 6$) and inlier ratio is small (e.g. $w < 10\%$): 10^6 iterations for 1% probability of failure. (How?)

Vineeth N B (IIT-H) §3.1 Feature Matching

Some limitations of RANSAC is when the inlier ratio which means the number of inliers in data divided by the number of points in data is unknown, then you could have problems in being able to fit. Also when the inlier ratio is very small, and the minimum number of samples to be drawn is large.

Let us assume that your definition of a model is such that you need to let’s say there are 12 free parameters, which means you need at least six correspondence that you need to draw to learn each model and then you need to check for how many of them lie within a particular epsilon distance of the correspondences so on and so forth. If your original inlier ratio itself was small in these images then you could have a tough time, in this particular case it will be equivalent to having about 10^6 iterations to ensure 1% probability of failure. Just to expand on this I am going to leave this for you to work this out by just giving a few hints and you can work it out later.

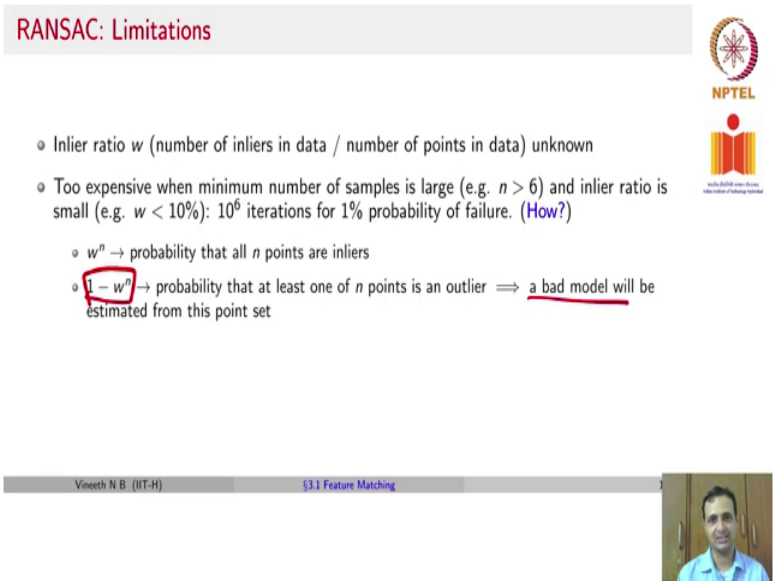
(Refer Slide Time: 42:05)



RANSAC: Limitations

- Inlier ratio w (number of inliers in data / number of points in data) unknown
- Too expensive when minimum number of samples is large (e.g. $n > 6$) and inlier ratio is small (e.g. $w < 10\%$): 10^6 iterations for 1% probability of failure. (How?)
    - $w^n \rightarrow$ probability that all n points are inliers

Vineeth N B (IIT-H) §3.1 Feature Matching



RANSAC: Limitations

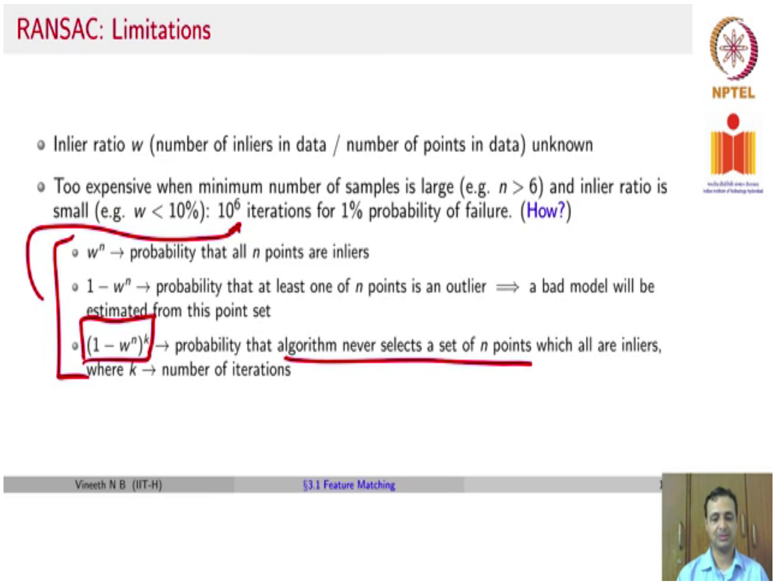
- Inlier ratio w (number of inliers in data / number of points in data) unknown
- Too expensive when minimum number of samples is large (e.g. $n > 6$) and inlier ratio is small (e.g. $w < 10\%$): 10^6 iterations for 1% probability of failure. (How?)
    - $w^n \rightarrow$ probability that all n points are inliers

    - $\boxed{1 - w^n} \rightarrow$ probability that at least one of n points is an outlier $\implies$ $\underline{\text{a bad model will}}$ be estimated from this point set

Vineeth N B (IIT-H) §3.1 Feature Matching

Remember that w is an inlier ratio, which means, w^n is the probability that all n points that you picked are inliers, so you pick n points that is the number of points that you have to pick to be able to solve for this model and w^n is the probability that all n points are inliers, which means, $1 - w^n$ is going to be the probability that at least one of those points is an outlier. And when one of those points is an outlier we are going to get a bad model which may not have too many correspondences in its neighborhood.

(Refer Slide Time: 42:46)



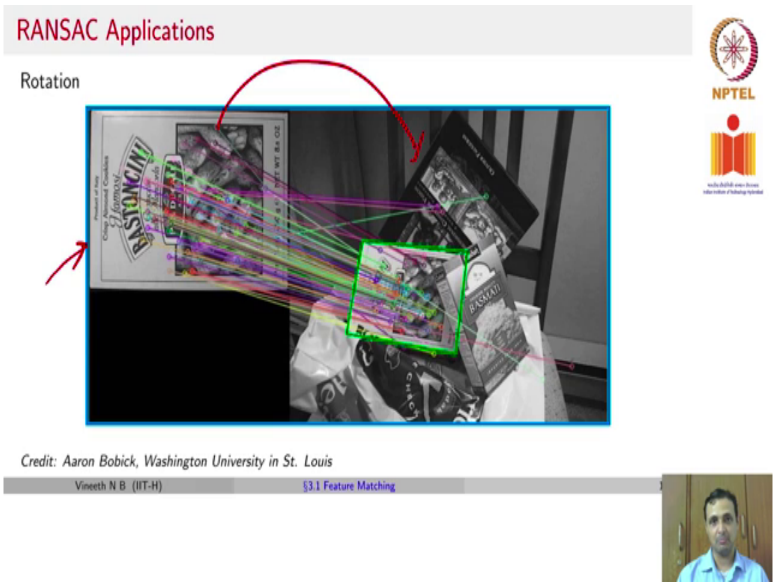
RANSAC: Limitations

- Inlier ratio w (number of inliers in data / number of points in data) unknown
- Too expensive when minimum number of samples is large (e.g. $n > 6$) and inlier ratio is small (e.g. $w < 10\%$): 10^6 iterations for 1% probability of failure. (How?)
  - $w^n \rightarrow$ probability that all n points are inliers
  - $1 - w^n \rightarrow$ probability that at least one of n points is an outlier $\implies$ a bad model will be estimated from this point set
  - $\boxed{(1 - w^n)^k} \rightarrow$ probability that algorithm never selects a set of n points which all are inliers, where $k \rightarrow$ number of iterations

Vineeth N B (IIT-H) §3.1 Feature Matching

If you round k different iterations, then the probability that the algorithm never selects a set of n points which are inliers your probability is going to be $(1 - w^n)^k$. This you are seeing is straight forward. Using these three terms you should be able to work out why this would happen as an example here. Try working out by yourself.

(Refer Slide Time: 43:16)



RANSAC Applications

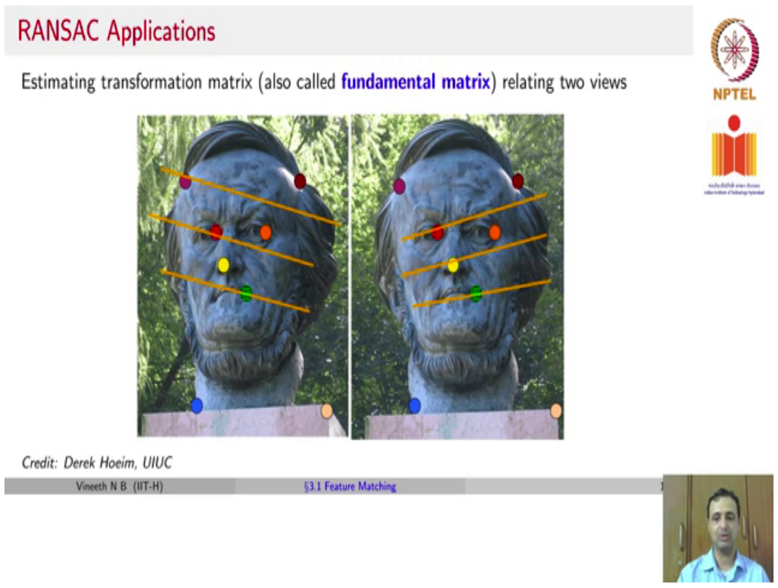
Rotation

Credit: Aaron Bobick, Washington University in St. Louis

Vineeth N B (IIT-H) §3.1 Feature Matching

Here are a few visual illustrations of how well RANSAC works for different kinds of transformations. Here is an example of rotation, this is the original book rotated to a certain degree and you can see that it is not a book sorry I think its a food box and this is the foot box placed in a different place in the second image and RANSAC finds fairly good transformations between these two settings.

(Refer Slide Time: 43:43)


RANSAC Applications

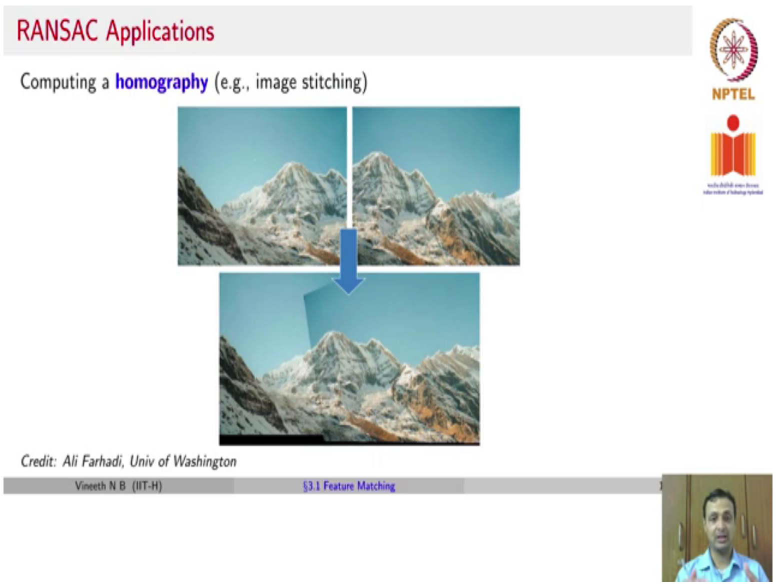
Estimating transformation matrix (also called **fundamental matrix**) relating two views

Credit: Derek Hoeim, UIUC

Vineeth N B (IIT-H) §3.1 Feature Matching

It also works well at estimating what is known as a transformation matrix or a fundamental matrix, when you relate to two views of the same image, if you have two different views remember, this is how you would build a 3D model of a given scene. And if you wanted to build a 3D model of say the statue you would ideally take multiple images by slowly moving around this particular 3D object and you would get a 3D model. And in each of those cases between every pair of images that you capture you have to estimate these transformation matrix, which is also known as the fundamental metrics in this particular case.

(Refer Slide Time: 44:22)



RANSAC Applications

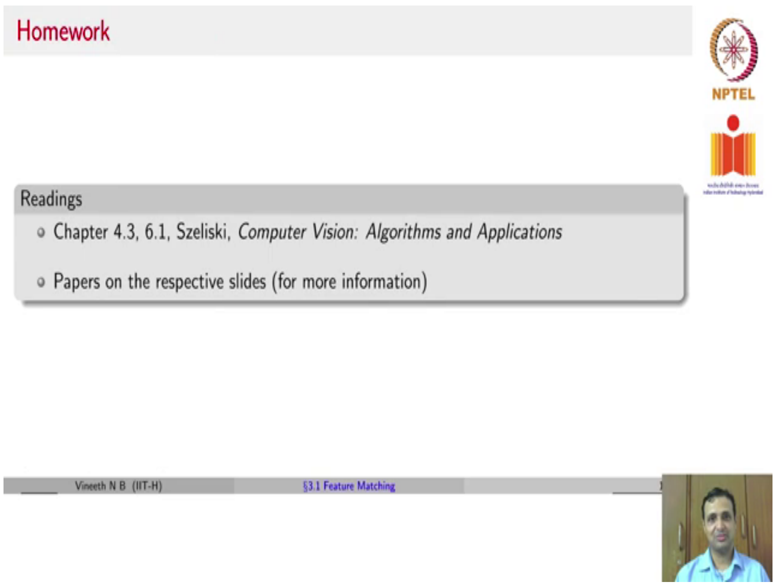
Computing a **homography** (e.g., image stitching)

Credit: Ali Farhadi, Univ of Washington

Vineeth N B (IIT-H) §3.1 Feature Matching

RANSAC is also used for what we started with in this lecture, which is to compute what is known as a homography, which is typically the term used for the transformation matrix in cases of image stitching. We said that it could be an affine transformation, translation, a rotation so on and so forth that transformation metrics is typically called a homography when you do it in the context of image stitching.

(Refer Slide Time: 44:50)

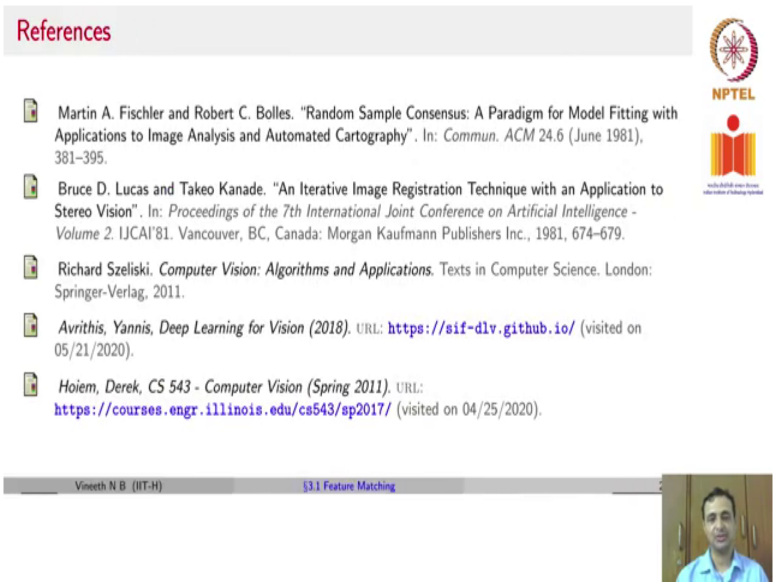


Homework

Readings

- Chapter 4.3, 6.1, Szeliski, *Computer Vision: Algorithms and Applications*
- Papers on the respective slides (for more information)

Vineeth N B (IIT-H) §3.1 Feature Matching



References

- Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), 381–395.
- Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, 674–679.
- Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.
- *Avrithis, Yannis, Deep Learning for Vision (2018).* URL: https://sif-dlv.github.io/ (visited on 05/21/2020).
- *Hoiem, Derek, CS 543 - Computer Vision (Spring 2011).* URL: https://courses.engr.illinois.edu/cs543/sp2017/ (visited on 04/25/2020).

Vineeth N B (IIT-H) §3.1 Feature Matching

That concludes this first lecture on feature matching. We will see a few other methods also as we go, but your readings for this lecture are going to be chapter 4.3 and chapter 6.1 of Szeliski's book. As well as if you are interested the papers on the respective slides for more illustration and the references are here.