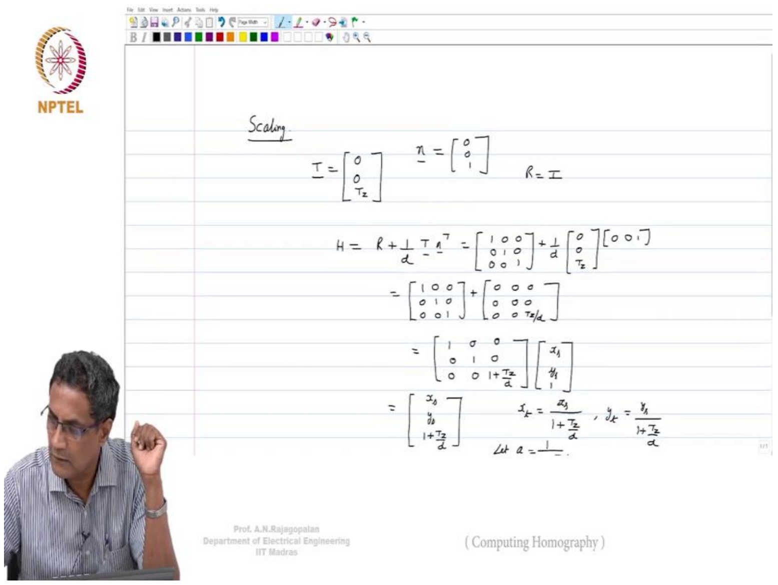
Image Signal Processing
Professor A. N. Rajagopalan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Lecture - 13
Computing Homography

(Refer Slide Time: 00:16)



Scaling

$$\underline{T} = \begin{bmatrix} 0 \\ 0 \\ T_z \end{bmatrix} \quad \underline{n} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad R = I$$

$$H = R + \frac{1}{d}\underline{T}\,\underline{n}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \frac{1}{d}\begin{bmatrix} 0 \\ 0 \\ T_z \end{bmatrix}\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & T_z/d \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1+\frac{T_z}{d} \end{bmatrix}\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_s \\ y_s \\ 1+\frac{T_z}{d} \end{bmatrix} \quad x_t = \frac{x_s}{1+\frac{T_z}{d}}, \; y_t = \frac{y_s}{1+\frac{T_z}{d}}$$

Let $a = \frac{1}{-}$.

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

(Computing Homography)

So let us talk about Scaling. So what this actually means is now again, we have to be able to show that those equations that you saw earlier, xt is equal to some a times ax and yt is equal to a times, what was that a times ys, that is what we had. That was a notation that we had used. So we had assumed that there is some a that is there.

Now we want to actually show what camera motion and what kind of a plane normal, how the plane should be with respect to the camera so as to be able to yield a motion like that. So for that what we will do is we will kind of look at this situation where it is a fronto parallel plane. So it is like 0 0 1. And you have a translation vector, the camera. No, the camera has to move. So the camera does now translate along the, along the optical axis. So your Tx is 0, Ty is 0 and then you have only a Tz component.

So such, like saying that, you have a camera like that and you have a fronto parallel plane because n is taken to be 0 0 1. And now you are, maybe you are say translating about the optical axis, there is no translation like that, it is only going along the optical axis. And then R will be identity, we are not going to rotate or anything. So R is going to be identity.

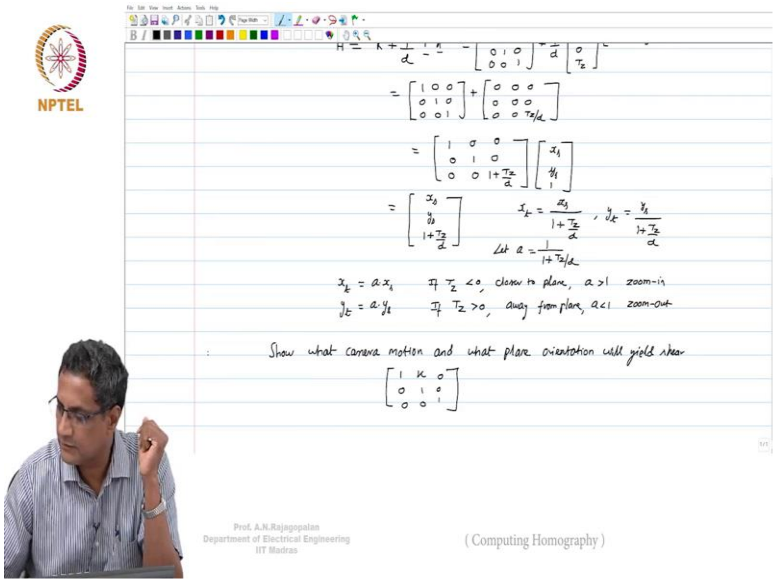
Under these conditions will show that we will end up with those equations that we saw earlier. So what that actually means is that now let us again go back and then, until now that I have been assuming the f to be 1 by the way, all along, I do not know whether I explicitly told you that focal length, I have assumed it to be 1, and reality I will talk about that a little bit toward later on. That matrix is actually a little more complex than what I kind of see, what I have said here.

We will not go into the full details, but I just wanted to kind of say, touch upon the fact that the intrinsic camera matrix is a little more complicated than what I have assumed till now. Just for understanding sake, I tried to keep that away for the time being. So our K is identity right now. So all that we have, we are worried about is R plus, so we are sort of assuming that K is identity, so we are just looking at R plus 1 by d T in transpose. So this yields you 1 0 0, 0 1 0, then 0 0 1 plus 1 by d. Now T is 0 0 Tz, there is only a z component and then there is an n transpose with this 0 0 1 because it is a fronto parallel plane.

So under this condition you will get 1 0 0, 0 1 0, 0 0 1 plus this will be 0 0 0, 0 0 0, again 0 0 Tz by d. So add the two up and you will get 1 0 0, 0 1 0, 0 0 1 plus Tz by d. Now add these on some xs ys 1 in order to know where it goes. So your xt, now this will be equal to what you get, xs ys and 1 plus Tz by d, which means that if I want my real image coordinate because it is in still in homogeneous, we are dealing with homogenous coordinates; for in order to get my image coordinates, I should scale my xt yt that I really need. The actual xt yt that I need are xt will be then xs upon 1 plus Tz by d.

This I told you already you have to divide by this, the third component, this is dz by d and yt will then be that means on the image, on the image space, on the image grid you will lead to coordinate where yt is equal to ys upon 1 plus Tz by d and imagine that a is, let a be equal to 1 by 1 plus Tz by d.

(Refer Slide Time: 04:45)



$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & T_z/d \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1+\frac{T_z}{d} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_s \\ y_s \\ 1+\frac{T_z}{d} \end{bmatrix} \qquad x_t = \frac{x_s}{1+\frac{T_z}{d}}, \quad y_t = \frac{y_s}{1+\frac{T_z}{d}}$$

Let $a = \frac{1}{1+T_z/d}$

$x_t = a.x_s$ If $T_z < 0$, closer to plane, $a > 1$ zoom-in

$y_t = a.y_s$ If $T_z > 0$, away from plane, $a < 1$ zoom-out

Show what camera motion and what plane orientation will yield shear

$$\begin{bmatrix} 1 & K & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

(Computing Homography)

Then what this means is that if you have got like your, so we are kind of back to that equation that we have talked about a times xs, yt is equal to a times ys, and a is, of course, 1 by 1 plus Tz by d. And the Tz being, if Tz is actually if Tz is less than 0 that means if it is negative, that means we are coming closer to camera, then clearly a will be greater than 1, which means you will be zooming in. Of course, the maximum that you can come is Tz negative will be like till you can come up to d because it is d away. After that, you will actually hit the plane itself, so Tz negative can only go till d.

Then on the positive side, if Tz is greater than 0, you can have Tz, you can go as farther away as you can and you are going away from the camera. Sorry, closer to plane, not camera. Closer to plane and away from plane and we got a less than 1, which is zoom out.

So the idea was that at the time, we just we just assumed that there is an a, and we did not really bother about where that a was coming from, what will influence a, and so on. But now we know that what camera motion and how the plane should be in order for you to be able to see something like a scaling happening. This is small example, though I am going to leave it as an example.

Show, this is going to be a small little homework for you. Show what camera motion and what plane orientation will yield shear. Suppose you assume it to be like 1 K 0 1, this is like the X-Shear. Well, what plane orientation will, we also saw, no? Shear told you it was a special case of affine. Affine is like A B C D, Tx Ty, 0 0 1, but A B C D, if they turn out to be special like 1K 0 1 then it is an X-Shear. So I am just asking it because this is exactly the example that I gave in the class last time.

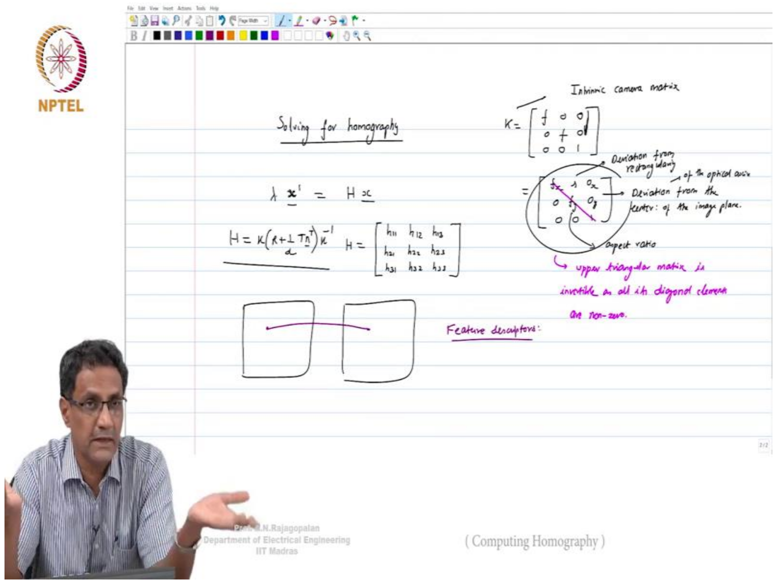
Imagine, now this you should be able to show by simply imagining the case where you have a car, imagine that, imagine that this is your X-axis, you are kind of looking out of the car, that is why you are, so the camera is pointed this way. So the camera optical axis is Z this way, Y is down. Imagine that you are moving this way, so the camera is sort of translating along X, and then it is actually looking at this kind of ground plane. So the ground plane is actually below the camera and you are actually imaging the ground.

So what camera motion and so I just wanted to write up the correct normal and the correct d, R. Well, I do not want to give away the answer. Whatever, it is a small little answer that you can just figure it out and so what camera motion, what plane orientation will yield shear? Okay, now let me put up the kind of the kind 1 K 0 1 0 0 1 0 0 1.

So again, so this is again going from the homogeneous to the things that you saw when you considered very simplistic cases. Now, what we have not talked about until now is the fact that how do you really arrive at this homography. Now right here, I am just assuming that somebody gives you all this.

In reality, if I give you two images and then they are taken from the same camera and I ask you to relate the two views, to find out what homography will take you from one to the other, nobody will tell you what, now somebody does not sit there and tell you that this point will go, is going there and that this point is going there and so on. If somebody told you, of course, it will be easy, but then in general, you cannot sit and do that.

(Refer Slide Time: 08:58)



Solving for homography

$\lambda \underline{\mathbf{x}}' = H \underline{x}$

$H = K\left(R + \frac{1}{d} T \underline{n}^T\right) K^{-1}$ $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$

Intrinsic camera matrix

$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$= \begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$

Deviation from rectangularity

Deviation from the center of the image plane. of the optical axis

aspect ratio

upper triangular matrix is invertible as all its diagonal elements are non-zero.

Feature descriptors:

So solving for the homography, let us do solving for homography. There is also a special homography which is a rotation homography. What we are looking at is really a planar homography. So rotation homography I will do later. For the time being, I will stay away from that.

And one more thing, let me just mention that. Yeah, maybe at this point of time, I will also mention about this intrinsic camera parameter matrix. See this K that we have taken, we have said that it is f 0 0, 0 f 0, 0 0 1. This is actually a very, very simple situation. Normally the way this K looks like, the real intrinsic sort of a camera parameter, a camera this one, this is called the intrinsic camera matrix.

Intrinsic because it is intrinsic to that camera, it stays unchanged, intrinsic camera matrix. The other one that you had which involved R and T, which was the camera motion that is called the extrinsic camera matrix, because that is like, that is that is not inherent to the camera, you can take a camera, move the way you want so it is extrinsic; this is intrinsic. So in reality what you really have is something like fx s ox, then 0 fy oy but those are only for information.

So what this actually means is you know, so in a real camera, you are kind of assuming that, when you put f 0 0 0 0 0 0 1, what you are assuming is that the optical axis, it coincides exactly with the image plane center. It need not, because optical axis of the lens, when you, when it passes through the image plane, we are assuming when you put the 0 0 right here, this 0 0 means that you do not have any kind of a deviation. It is exactly that meets the center of the image plane.

In case it does not and if there is a drift then this ox and 1o these are again things that nobody may be able to give you and these are supposed to be estimated. So that is why it is called a calibration procedure. We are not going to go into that but typically if I want to use a camera then you should know these parameters. If you are just doing a homography you do not need any of this. There is also something that I want to emphasize that if you are just solving for a homography between two images, you do not care about all these intrinsic parameters and all.

All that you need is the, is that matrix that is going to take you from one to the other. But if you want to do some high-level tasks and all like for example, if you want to take a camera, you want to estimate depth, 3D depth, and all that, then in that case this intrinsic camera information we will need.

So solving for this, so right now we are assuming that K is of this form and f is 1, and so on. So ox oy is like deviation from the center, it is like a deviation from the center or in other words a center, center of the image plane. Deviation of the optical axis from the center of the image plane, from the center of the image plane. So we just typically assume this coincides and so on, but it may need not, there could be some errors when they actually manufacture.

You could get a small shift so that you have to account for. There is fx and fy, they are supposed to be the aspect ratio. See, normally the focal length is not really, and also, we will assume that it is f, f; but fx and fy they write. What they really mean is if there is an aspect ratio, there is a difference in the aspect ratio, then this fx and fy will take care of the aspect ratio and this s is something that actually takes care of deviation from what we call a rectangularity.

This will have something like this shear, this s, this parameter is something like a shear. Exactly like, I mean if you relate this to what we had when we were doing the homography. So s is exactly playing a similar role, except that this is like a hardware thing. So this is like inside the, if the sensors if there is some kind of deviation from a rectangularity then this s is summoned. This may not happen in the sense that you may simply if there is no deviation, s will be 0.

But if there is, then it needs to be accounted for. So this is like deviation from rectangularity, deviation from rectangularity. So these, these are things that if you look at Ka typically, it will be of this form. So it will have a kind of I unknown sitting inside it. But as far as we are concerned and anywhere since it is an, it is an upper triangular matrix, you see this is an upper triangular matrix. And you know that an upper triangular matrix is all invertible as long as the diagonal elements are non-zero.

And fx fy and all are, I know you can relate them to focal length of the aspect ratio, so basically this is none of these entries is 0. So this matrix is always invertible. So that K inverse that I kept the writing, I kept writing it for f 0 0 situation but the same thing is also valid, even if you take a real intrinsic camera matrix where, maybe there are some other entries that you need to account for, there is no issue, there is always invertible. There is an upper-triangular matrix with non-zero diagonal entries, upper-triangular matrix.

In fact, this is also a reason why, why let us say, when these people, when these people do a camera calibration, have you heard about QR decomposition? In matrix, that there is a QR decomposition. What does it do? It kind of gives out a matrix as a product of an upper-triangular and an orthogonal matrix. So that upper-triangular is the K guy, that is the way they do the camera calibration. And then the orthogonal matrix is your R, R as an orthogonal matrix.

So just saying that, so people do all this camera calibration with all, there are, there are kind of nice ways to do that. It is a upper-triangular matrix, always invertible, is invertible. As invertible as all its diagonal elements are non-zero, elements are non-zero.

Then now let us get a comeback. So what we are saying is we have this, so we said that we have x dash, which is like which is this homogeneous coordinate of your target. And we are saying that the H acts on x, where let us say, x is the source. So x is like xs ys 1. So if you think about it this is like, and I am just going to write x because I do not want to carry this s, s always.

So this is like applying the homography on the source grid and then it takes you to a target grid which is x dash and in order to get back the, get back your, so in fact, I will even multiply this by lambda just to, just to indicate the fact that when you do Hx, the, I mean third entry here need not always be 1. You can end up with some lambda, which you will have to use to divide the first two entries.

So you have something like lambda x dash is equal to H times x. This is how you, this is what you have now. The point is if you look at, if you look at your homography H, let us say that H is, H has a following entries and now h11. And this could be the case that you have an intrinsic matrix also sitting there whose values you do not know. You do not know any of these parameters, let us say. You have this, you have this most general case. It is still valid because all that we are saying is this H is some K.

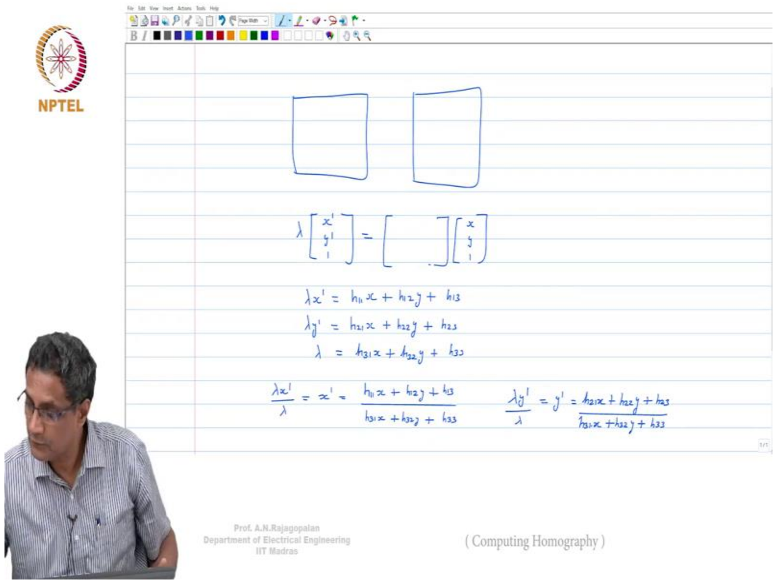
So H is K R plus 1 by d Tn transpose and then what was that, K inverse. So we are saying that let that K be there whatever it is, finally, I would, I just need to map this image onto that. As far as a homography is concerned, I am not worried about what are intrinsic parameters and all because all that will get absorbed in my H.

But if you want some other task and all, you cannot always ignore and move on. You may have to go back and look at what is my camera, what was the intrinsic parameter but for a homography, it does not matter. So now what we are saying, so let this h11 be, now let h be h11 h12 h13, h21 h22 h23, h31 h32 h33 and we said that h has 8 unknown, you can only estimate it up the scale factor and which is anyway being shown here by that lambda.

So now since there are 8 unknowns and so what this means is that you need at least four feature correspondences. Because one feature correspondence, now I mean, in order to see that, let us start writing. Let us say that I have an image here and then I have a source and then I have a target and suppose somebody says that, well, there is one point here that matches that goes here. Matches in the sense that I know that, that this, imagine that probably this is a corner of some of this one table or something and I can kind of identify that corner in the second image.

Now, this is not typically done manually. So what is done is typically you use some feature descriptors. You use what are called feature descriptors and there are several of them, in fact, people have spent their whole, spend a whole lifetime trying to find out what kind of features are good to match.

(Refer Slide Time: 19:00)



$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\lambda x' = h_{11}x + h_{12}y + h_{13}$$

$$\lambda y' = h_{21}x + h_{22}y + h_{23}$$

$$\lambda = h_{31}x + h_{32}y + h_{33}$$

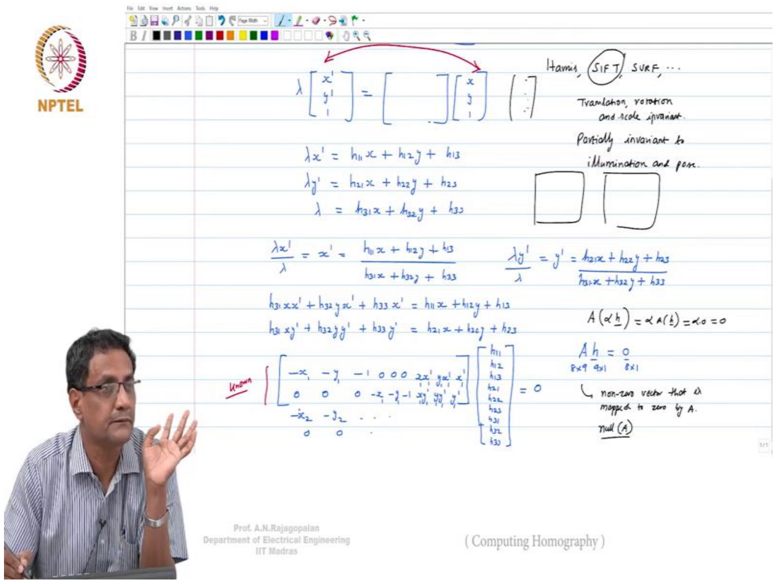
$$\frac{\lambda x'}{\lambda} = x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad \frac{\lambda y'}{\lambda} = y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

So I was saying that, that you have these two images and I mean, like I was saying there are these, there is a lot of work which has actually gone into trying to match features because that is one of the fundamental things that you need in several applications, especially when you are trying to solve for a homography because if you try to see what is happening here, you have 8 unknowns sitting in that homography. And if you had four feature correspondences, then it will yield you 8 equations. I will show you why.

So let us say that you have lambda x dash y dash 1 is equal to h which is h11, whatever right; and then we have x y 1. So if you just expand this, you will get lambda x dash is equal to h11 x plus h12 y plus h13 and that is what is sitting in this h, then lambda y dash is equal to h21 x plus h22 y plus h23, and finally, you have lambda is equal to h31 x plus h32 y plus h33.

Which then means that but which then means that I can do lambda x dash by lambda, which will give me my x dash. So that is h11 x plus h12 y plus h13 upon lambda, which is h31 x plus h32 y plus h33. Similarly, we have lambda y dash by lambda giving y dash is equal to h21 x plus h22 y plus h23 upon h31 x plus h32 y plus h33.

(Refer Slide Time: 21:14)



$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} & & \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Harris, SIFT, SURF, ...

Translation, rotation and scale invariant.

Partially invariant to illumination and pose.

$$\lambda x' = h_{11}x + h_{12}y + h_{13}$$
$$\lambda y' = h_{21}x + h_{22}y + h_{23}$$
$$\lambda = h_{31}x + h_{32}y + h_{33}$$

$$\frac{\lambda x'}{\lambda} = x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad \frac{\lambda y'}{\lambda} = y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

$$h_{31}xx' + h_{32}yx' + h_{33}x' = h_{11}x + h_{12}y + h_{13}$$
$$h_{31}xy' + h_{32}yy' + h_{33}y' = h_{21}x + h_{22}y + h_{23}$$

$$A(\alpha \underline{h}) = \alpha A(\underline{h}) = \alpha \cdot 0 = 0$$

$$\text{known} \left\{ \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & \cdots \\ 0 & 0 & \cdot \end{bmatrix} \right. \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

$$\underset{8\times 9}{A}\ \underset{9\times 1}{\underline{h}} = \underset{8\times 1}{\underline{0}}$$

non-zero vector that is mapped to zero by A.

null (A)

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

(Computing Homography)

So if you can cross multiply, so what will we have? We will have, I am going to write this as h31 xx dash plus h32 yx dash plus h33 x dash is equal to h11 x plus h12 y plus h13. And for the other one, you will get h31 xy dash plus h32 yy dash plus h33 y dash is equal to h21 x plus h22 y plus h23. Correct. So now the unknowns we know are these h11 to h33 sitting there.

So what we can do is, we can in turn write this matrix. So we can write it as, let me go with the standard notation. So that is like, so what we will have is minus x, minus y, minus 1. So I am trying to get all the terms on one side. Then see, so here it assume that, that you have this matrix multiplying h11 h12 h13 h21 h22 h23 h31 h32 h33. So you can put all your unknowns onto this particular vector.

And you know that this vector needs to be found only up to a scale factor. So minus x minus 1, so if you see, what I regard h11 x therefore you pull it to the left side it will become minus x. Then h12 is getting multiplied by minus y, h13 is getting multiplied by minus 1, then h21 nothing, so 0 0 0. And then h31 is xx dash, h32 is yx dash and h33 is x dash. And similarly what you will get from the other one is, h11, there is nothing, so you have like 0 0 0 and then h21 minus x minus y minus 1 and h31 is xy dash. So this is x into y dash, these all coordinates that you are multiplying yy dash and y dash. This whole thing is equal to 0.

So now the point is to see, so all these entries we know these are all known quantities. This is all known to us because when will it be known when somebody tells me that this point is going there. Because I have assumed that somebody has given me a correspondence between x dash and x, somebody has said that if you apply h on this xy1, it will take you to x dash y dash 1.

So which then means that with just one point correspondence, I have two equations and with, and since I have 8 unknowns, I will need four-point correspondences. So somebody has to give me not just one, but people have to give me at least four-point correspondences so that I can solve for h because h has only 8 unknowns and it is known up to a scale factor.

So then you can think this as you can add an x1 now, so you can say that for let us say one feature point you have x1 y1 x1 dash, and sorry, x1x1 dash y1x1 dash x1 dash, x1 y1 x1y1 dash y1y1 dash y1 dash. And similarly, follow it up with minus x2 minus y2 blah, blah, blah, 0 0 0 and go on. So you will have x1 x2 x3 x4, so you will have 8 rows minimum, you can of course use more.

So this whole thing we can write this as a multiplying h equals 0, where 0 is again actually a vector. So a is 8 cross 9, h is 9 cross 1, 0 is 8 cross 1 because 0 is a vector it is not a scalar, ah is equal to 0. So it is also called a homogeneously squares. So what you are asking is, now the point is that you might ask who will to give me those point correspondence and so I was just talking about that.

So basically, people have spent a lot of time trying to get these correspondences and there are various things for that. So one is called a, I mean there are various such features that let us say people have found out, one is called the Harris detector, it is called the Corner detector. Harris Corner Detector, something called SIFT, which is a scale-invariant feature transform, it stands for scale-invariant feature transform. And there is something called SIRF, which stands for speeded-up robust features. And there are kind of see, many of them.

Now among these, SIFT is the one that is most common, this was actually proposed by a guy called David Lowe. So this is the most common and the nice thing about this is, this is completely a translation, rotation, and scale invariant. And it is partially invariant to, partially invariant to illumination and pose.

What this means is that even if you have a pose change, it will still be able to reasonably identify that it is exactly that point. So it can tell that between these two images that you have, it can, so if you have to run a SIFT code, this is available, in fact, when you do the assignment, we will give you that code. So in this course, I do not go through the entire details about how SIFT works and so on when we do a vision course, computer vision course there we do.

But here, just keep in mind that there are many such things that can allow you to identify features, identify feature correspondences across images, and SIFT is one of them and it is pretty fast and it is used, it is used all over.

But these are all called handcrafted features by the way. These are not your deep learning kind of features, these are all handcrafted, somebody figured out that if we do this if you do Gaussian scale space and so, you do some scale-space processing, you will be able to identify features.

So in terms of saying that if you had a scene captured under some illumination, you capture the same scene after some time from a different viewpoint, but then even that is a slightly different illumination, you still match those, match those correspondences and you will still be able to tell how the two are aligned, what geometric transformation can bring one to the other. So we will assume that there is something like a SIFT available with us which will kind of so, you can just run it, I mean we will just give you the code, you can run it, you will see that it would not just pick three or four and all you pick many.

Now the point is, for us, it looks like if I had just four feature point correspondence, I am done because a being 8 cross 9 and as long as you make sure that not three, no three points are co-linear, then you can be assured that it a has a rank 8. Now which actually means that because of the fact that at ah is equal to 0, that means there is at least one kind of non-zero vector which vector that is mapped to 0 by a, because a is 8 cross 9.

So you know that there is at least one non-zero vector that is mapped to 0 by a. So one way to do it is you do a simple SVD, so if you use MATLAB or something, you will ask for null of a. So you will say null of a. I mean that is that one-dimensional vector, I do not know which one just have dimensional space in which this particular say, vector lies. All scaled versions of that are okay.

So when you do null of a, so typically what MATLAB does is it will do a SVD on a and then pick the, pick that Eigenvector corresponding to the smallest singular value. The zeroth similar singular value in fact. Because you know for sure that assuming that it, all these are that, I mean in this case we are assuming that you do not have three points lying on the line and so on. You just have to avoid those things, but otherwise, you can just pick any of these four points and you can run ah equal to 0, ask for null of a this will return an h for you, which is non-zero, which when acted upon by a will actually yield you the, I mean zero vector.

Now you can also clearly see that if you have any scaled versions of h, that will also go to zero because a alpha h is nothing but alpha ah, and but then, alpha, but then ah we know is 0. So this is like alpha into 0, so this is 0. So all scaled versions of h, so we can only estimate h up to a scale factor, which is fine because as we said originally, we have only 8 unknowns in h. Anyway we are okay with estimating h up to a scale factor.

So whatever h that is actually returned is what you can directly use now and you can actually align the images now. So that h you go back and put it back in your h matrices h11 h12 h13 h21 h22 h23 h31 h32 h33. And then if you multiply it with any xy1, what will happen is you may, you will get some, you will get actually a 3D, I mean, that is a three-dimensional vector and this need not be 1. So you read a scale the first two by this and that will give you the image coordinate.