Deep Learning for Computer Vision
Prof. Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 13
Other Feature Space

(Refer Slide Time: 0:15)

Deep Learning for Computer Vision

Other Feature Spaces

Vineeth N Balasubramanian

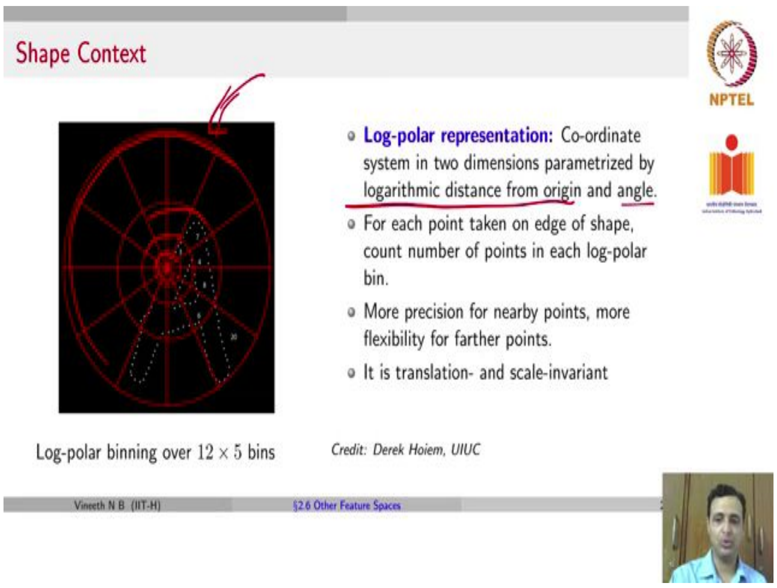
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

We started this week by talking about edges. Then we went from edges to blobs to corners, and then talked about a few varieties of corner detectors and talked about an important corner detector called sift, talked about the feature descriptor of how you would describe your corner as a vector. And then in the last lecture, we also talked about image segmentation.

Now, we are also going to talk about other kinds of feature spaces that people worked on in computer vision, before deep learning came into the picture. That is going to be the focus of this lecture.

(Refer Slide Time: 1:04)



Shape Context

- **Log-polar representation:** Co-ordinate system in two dimensions parametrized by logarithmic distance from origin and angle.
- For each point taken on edge of shape, count number of points in each log-polar bin.
- More precision for nearby points, more flexibility for farther points.
- It is translation- and scale-invariant

Log-polar binning over 12×5 bins

Credit: Derek Hoiem, UIUC

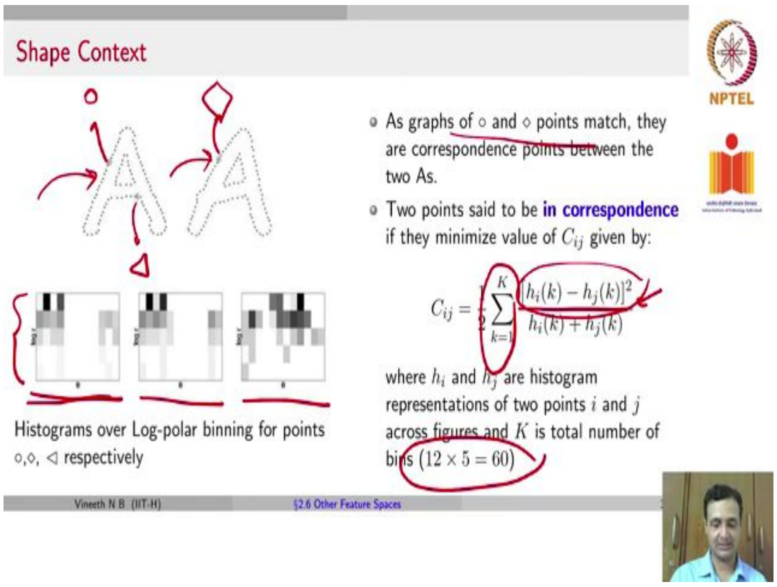
Vineeth N B (IIT-H) §2.6 Other Feature Spaces

One of the other important feature representations that was developed was known as shape context. Shape context was about taking a pixel and placing and getting a log-polar representation of the neighborhood around the pixel using this kind of a representation. This kind of representation. So we define a coordinate system in two dimensions, that is parameterized by logarithmic distance from the origin, which is the pixel at which you are trying to get the representation as well as the angle.

So for each point that you consider, you would count the number of points in each log-polar bin around that particular pixel. So you would, so if this log-polar representation you can see that there are perhaps something like one, two, three and four regions in terms of angular distance in terms of log distance and then 12 bins in terms of the angels themselves.

So, as you can see, there is more precision for nearby points, very similar to the Gaussian kernel, where you give more weightage to nearby points, and then more flexibility for farther points on that pixel where we are considering the representation. To some extent, this method, this representation is translation and scale invariant, depending on the size of the neighborhood you take.

(Refer Slide Time: 2:53)


Shape Context

- As graphs of ○ and ◇ points match, they are correspondence ~~points between~~ the two As.
- Two points said to be **in correspondence** if they minimize value of C_{ij} given by:

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

where h_i and h_j are histogram representations of two points i and j across figures and K is total number of bins $(12 \times 5 = 60)$

Histograms over Log-polar binning for points ○,◇, ◁ respectively

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

Let us study this in a bit more detail. So let us consider two different images. So, now, if you consider this pixel, there are if you can now see carefully, this is a circle, this is a triangle and this is a rhombus. So, if you can see here you can find the first one is the representation of the circle, the second one is the representation of the rhombus and the third one is the representation of the triangle.

How did these representations come? There are five bins in terms of $\log R$ of the log radial distance and 12 bins in terms of your θ or your angles around the center. So, we can see here that the circle and the rhombus have very similar representations so one could consider these two points as correspondence points between these two representations or these two images of A.

And when we say two points are in correspondence? We say two points are in correspondence if they minimize a value of Cij where Cij, i and j being two pixels that you are trying to measure and

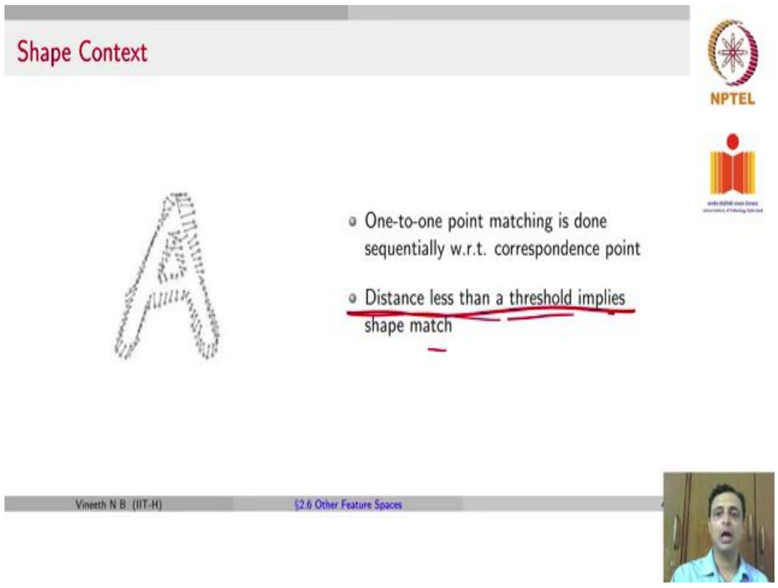
$$C_{ij} = \frac{1}{2}\sum_{k=1}^{K}\frac{(h_i(k)-h_j(k))^2}{h_i(k)+h_j(k)}$$

and $h_i(k)$ and $h_j(k)$ are the histogram locations in the ith, around the ith pixel and in the representation around the jth pixel, h_i is the histogram of the number of pixels in each of those bins around the ith pixel and h_j similarly for the jth pixel. $h_i(k)$ would correspond to the count in one of those bins and what is small k or capital K, capital K is the total number of bins, right. So for $h_i(k)$ would be the count in one of those bins.

So, when would C_{ij} get minimized when the numerator is zero or the histograms exactly match or when the histograms are fairly close to each other, you would ensure that these values will turn out to be low.

The denominator ensures that you are also considering the number of pixels considered while evaluating this, very similar to how we spoke about a normalized cut and min-cut the denominator helps in normalizing based on the number of pixels that you are considering for your studies.

(Refer Slide Time: 5:31)



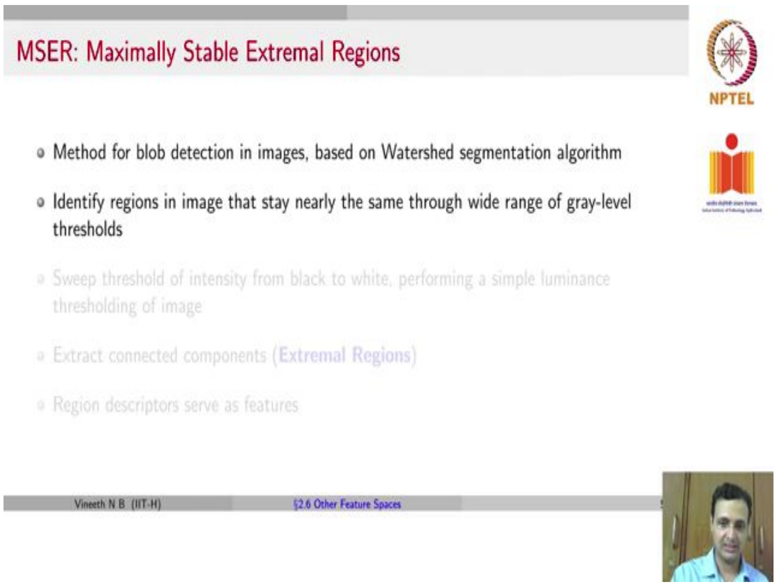
Shape Context

- One-to-one point matching is done sequentially w.r.t. correspondence point
- Distance less than a threshold implies shape match

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

So once you get these correspondence points between two images based on the representation, you could then do a one to one matching sequentially between these points between these correspondence points and when a distance is less than a threshold this kind of an approach was used to check for a shape match, not just a point match, but matching shapes between two different images. That is the reason it was called shape context.

(Refer Slide Time: 5:58)



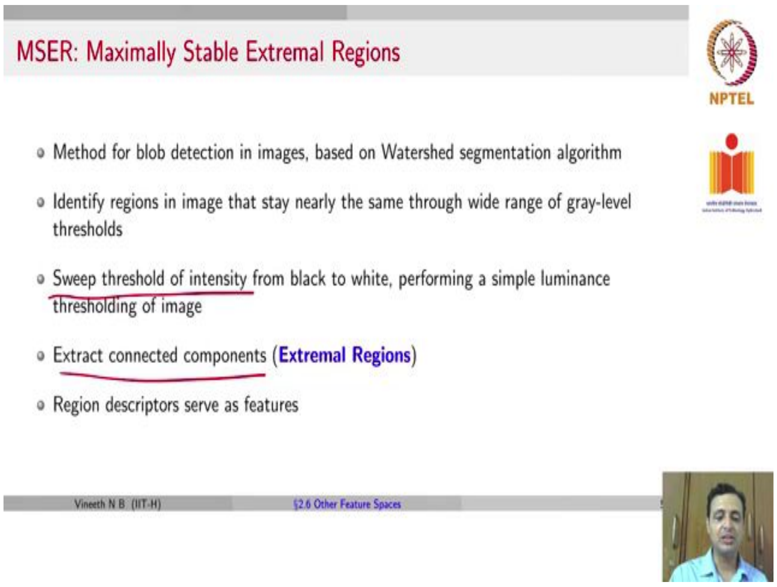
MSER: Maximally Stable Extremal Regions

- Method for blob detection in images, based on Watershed segmentation algorithm
- Identify regions in image that stay nearly the same through wide range of gray-level thresholds
- Sweep threshold of intensity from black to white, performing a simple luminance thresholding of image
- Extract connected components (**Extremal Regions**)
- Region descriptors serve as features

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

Another feature space or important method that was proposed is known as the MSER or the maximally stable extremal regions. This was a method for blob detection based on watershed segmentation. So, we just saw image segmentation in the previous lecture using one of those methods, which is simple watershed segmentation that method is used to come up with a method for MSER or maximally stable external regions. And how does this work? You identify regions in an image that stay nearly the same as you keep varying a threshold.

(Refer Slide Time: 6:42)



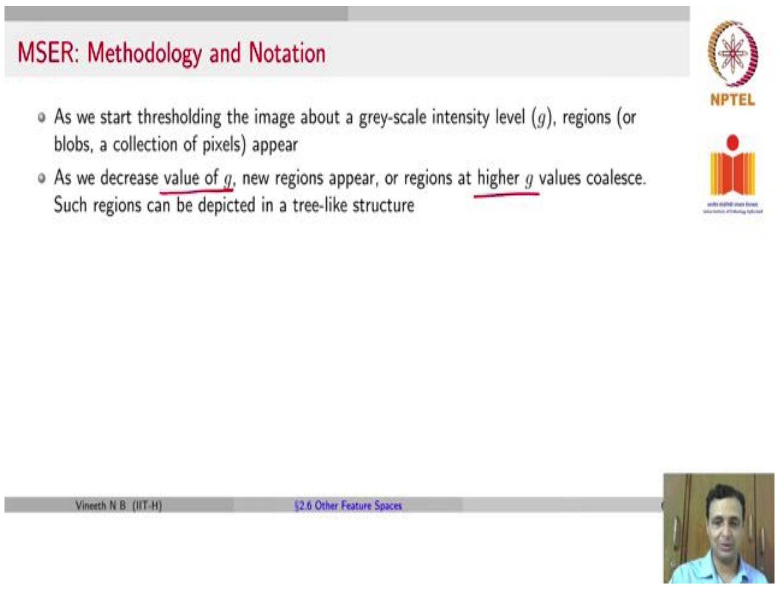
MSER: Maximally Stable Extremal Regions

- Method for blob detection in images, based on Watershed segmentation algorithm
- Identify regions in image that stay nearly the same through wide range of gray-level thresholds
- Sweep threshold of intensity from black to white, performing a simple luminance thresholding of image
- Extract connected components (**Extremal Regions**)
- Region descriptors serve as features

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

Let us try to describe this. So you would ideally sweep the threshold of an intensity going from white to black, or black to white, and in each step, you are simply thresholding based on that particular intensity value. So if you have an intensity of 75, everything above that is white and everything below that is black a simple thresholding operation. Then you extract connected components within that thresholded image, and the final region descriptors serve as the features.

(Refer Slide Time: 7:12)



MSER: Methodology and Notation

- As we start thresholding the image about a grey-scale intensity level (g), regions (or blobs, a collection of pixels) appear
- As we decrease value of g, new regions appear, or regions at higher g values coalesce. Such regions can be depicted in a tree-like structure

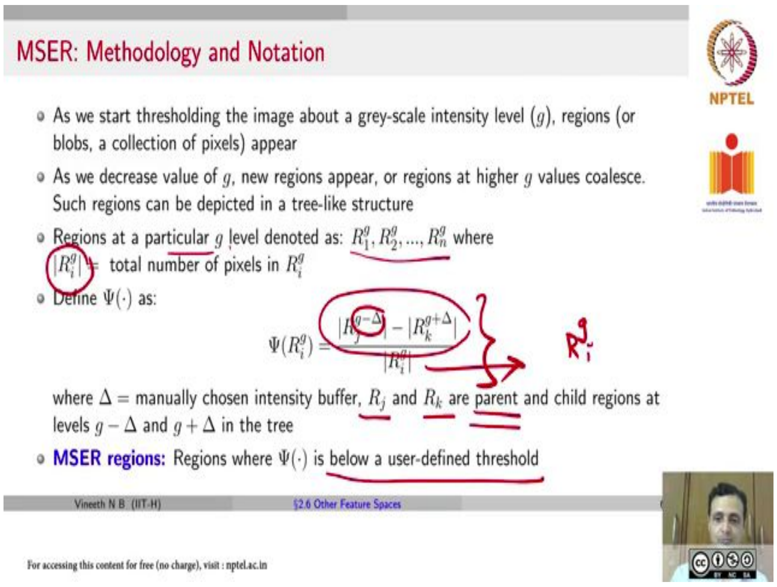
Vineeth N B (IIT-H) §2.6 Other Feature Spaces

Let us see this more clearly with some illustrations. So let us say we start thresholding the images about a particular intensity level g, so which means you considered a threshold g, and everything above that is considered to be white, and everything below that is considered to be black. So your threshold binding threshold, your image, based on the threshold g. Clearly, when you threshold, you are going to see a bunch of blobs or collections of pixels appear.

They are going to be cohesive regions that appear together when you threshold. So and as you increase the value or decrease the value of g whatever it be. In this case, if you say decrease the value of g, newer regions would start appearing. If you decrease the value of g you are now giving a lower threshold for considering regions to be white and the rest of them to be black, so there would be more regions that start appearing with white pixel values in the thresholded image.

And if you increase g, things would start coalescing, and these regions that get developed as you keep changing g values can be depicted in a tree like structure. It is very similar to region merging and regions splitting, as you reduce g some regions may merge as you increase g regions may split based on their intensity values.

(Refer Slide Time: 8:36)


MSER: Methodology and Notation

- As we start thresholding the image about a grey-scale intensity level (g), regions (or blobs, a collection of pixels) appear
- As we decrease value of g, new regions appear, or regions at higher g values coalesce. Such regions can be depicted in a tree-like structure
- Regions at a particular g level denoted as: $R_1^g, R_2^g, \dots, R_n^g$ where $|R_i^g|$ = total number of pixels in R_i^g
- Define $\Psi(\cdot)$ as:

$$\Psi(R_i^g) = \frac{|R_j^{g-\Delta}| - |R_k^{g+\Delta}|}{|R_i^g|}$$

where Δ = manually chosen intensity buffer, R_j and R_k are parent and child regions at levels $g-\Delta$ and $g+\Delta$ in the tree
- **MSER regions:** Regions where $\Psi(\cdot)$ is below a user-defined threshold

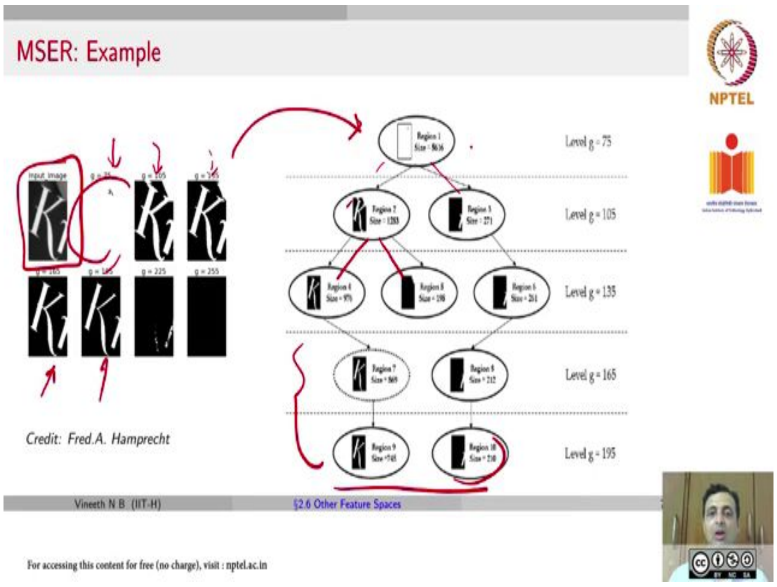
So now, what do you do with this? So regions at a particular g level be denoted as $R_1{}^g$ to $R_n{}^g$, let us say there are small in regions, each corresponding to one particular threshold g. Where the cardinality of $R_i{}^g$ denotes the total number of pixels in one of those regions $R_i{}^g$, then we define a quantity ψ, which is given by $\frac{|R_j{}^{g-\Delta}|-|R_k{}^{g+\Delta}|}{|R_i{}^g|}$. Rj and Rk are parents and children of Ri of that corresponding region at slightly different thresholds, g minus delta and g plus delta.

Let us try to study this a bit more carefully. Remember, again, that $R_i{}^g$ is one of the regions that you are considering at the threshold value g. Now, if you subtract a little value g minus delta, then you would get a parent region which is perhaps larger than Ri.

Remember when you subtract the threshold, the region would get, the white region would probably get bigger. So you could have a parent region Rk, which is defined at g minus delta. And at g plus delta when you increase the threshold the region may get smaller, the white region may get smaller, you are going to define that as a child and that is defined by Rk.

And if this quantity ψ, which is the difference between Rj and Rk is below a user defined threshold, you would call those regions to be maximally stable, extremal regions. Remember, if it is smaller than a threshold, it simply means that there is almost no change between the parent and the child. If there is a lot of change, then you probably need to expand a little bit further before calling it a stable or an extremal region.

(Refer Slide Time: 11:02)



MSER: Example

Credit: Fred.A. Hamprecht

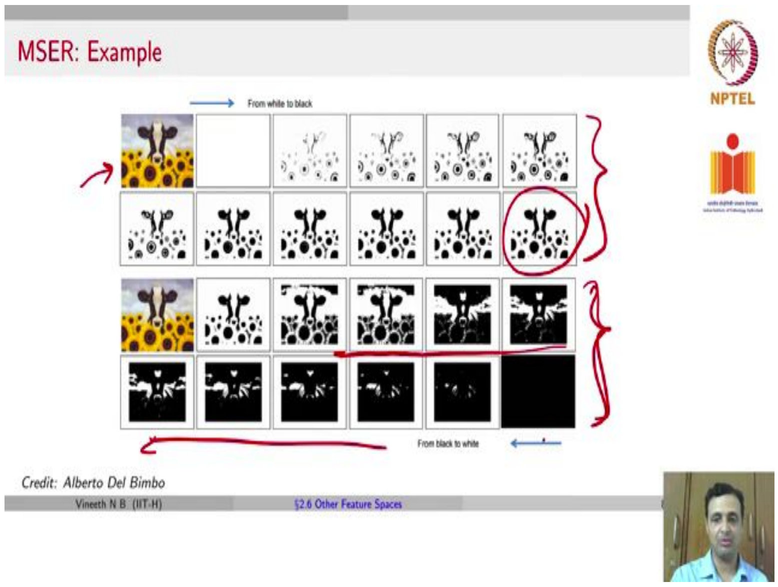
Vineeth N B (IIT-H) §2.6 Other Feature Spaces

For accessing this content for free (no charge), visit : nptel.ac.in

Here is a visual illustration. So, this is the input image, as you can see it has some characters in it. And when you set your threshold to be 75, assuming again that you use 8 bits to represent pixel so value is between 0 and 255 you see the regions not very clearly defined, but when you go to g is equal to 105, you see regions getting slightly more well defined. And you can see that at 135 there are a few changes, but not much at 165 very minor changes not much again, at 195, almost no change, at 225 things again start going to black because the threshold is very high.

So you could now represent that as a tree, where you start with a region, that region becomes two regions at level g is equal to 105 and region 2 becomes region 4 and region 8, at level g is equal 135, and so on and so forth to build the rest of the tree. So finally, as you can see at level g is equal to 195 you have two regions, one with the k, and the other with an r, which is what you are originally looking for. So this gives you one way of separating different regions. So this is not exactly a corner detector, but another method to separate regions.

(Refer Slide Time: 12: 29)


MSER: Example

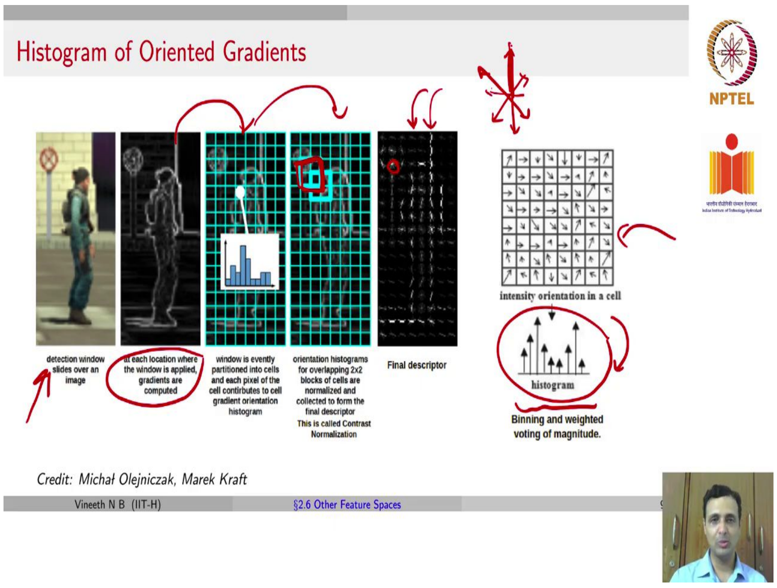
Credit: Alberto Del Bimbo

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

So another way, we said that it could also be used as a blob detector, and here is an example for doing blob detection using maximally stable single regions. So given this input image if you go from white to black or black to white the top two rows depict when you go from white to black, and the bottom two rows depict when you go from black to white. If you go from white to black, you start with a high threshold and then you decrease the threshold more regions in black start showing up, you decrease the threshold more regions in black start showing up and region start merging and you finally left with as, even as you change the threshold when regions do not change, you know that that is actually a stable blob representation of your original image.

Remember, when you see it from black to white, you are going to look at it in terms of merging. So you are going to say that when it is black, basically, there is no region and you slightly increase certain regions start showing up, and as you keep increasing the threshold this way, and then go up to the third row, you can see more stable regions showing as you go forward. So this is another approach to find out blobs or regions in an image called maximally stable extremal images.

(Refer Slide Time: 13:48)



Histogram of Oriented Gradients

detection window slides over an image
at each location where the window is applied, gradients are computed
window is evently partitioned into cells and each pixel of the cell contirbutes to cell gradient orientation histogram
orientation histograms for overlapping 2x2 blocks of cells are normalized and collected to form the final descriptor
This is called Contrast Normalization
Final descriptor
intensity orientation in a cell
histogram
Binning and weighted voting of magnitude.
NPTEL

Credit: Michał Olejniczak, Marek Kraft

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

Another famous method you probably have already been exposed to this when we talked about sift, but this method was also independently developed to be able to detect humans in image is known as histograms of oriented gradients or popularly known as hog or HOG. So this originally was proposed in a paper that was used for human detection in different kinds of images.

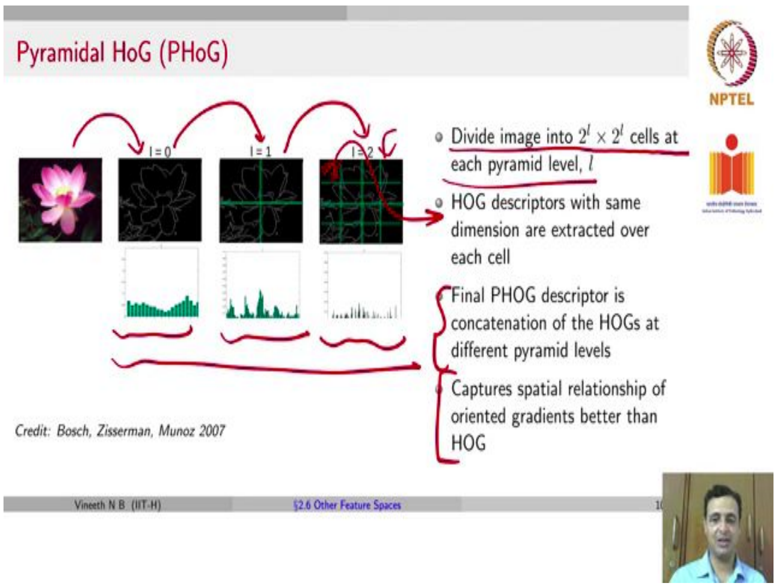
So, let us say you had a portion of an image or an image containing a human, so at each location where the window is applied you compute gradients. Remember, computing gradients is simply an edge detector whatever is detector and you want to choose. And then you divide the entire gradient image into a uniformly spaced grid. And then for every 2 cross 2 block in this grid you compute your orientations of the gradients, very similar to how we talked about it for sift you would have say multiple orientations.

Let us say you define 8 different orientations you see how many pixels in that block had orientations of gradients in certain bins and you bin them to get a histogram for each 2 cross 2 block. And you do this overlapping 2 cross 2 blocks, which gives you the histograms that you see on this image on every grid center.

So you can see here that each of these, each of these is just a different way of drawing a histogram. Remember, again, if you recall, something like this was just another way of drawing a histogram with 8 bins in 8 directions where the length of the arrow denotes the frequency count in each of those bins. This is just another way of representing that or this could be another way of representing a gradient orientations in getting the histogram.

So the histogram of oriented gradients was shown to be very effective for detecting humans in an image in the mid-2000s.

(Refer Slide Time: 16:00)



Pyramidal HoG (PHoG)

- Divide image into $2^l \times 2^l$ cells at each pyramid level, l
- HOG descriptors with same dimension are extracted over each cell
- Final PHOG descriptor is concatenation of the HOGs at different pyramid levels
- Captures spatial relationship of oriented gradients better than HOG

Credit: Bosch, Zisserman, Munoz 2007

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

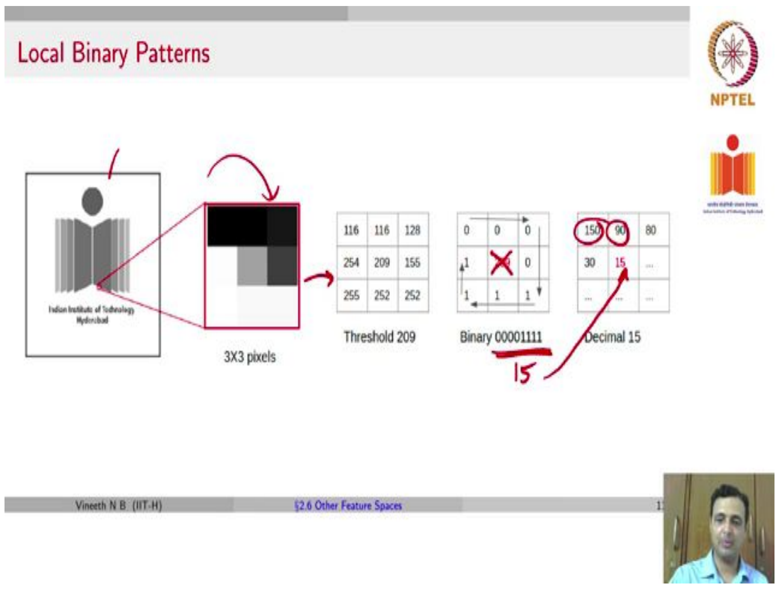
And this was also improved upon to give what is known as pyramidal HOG, a PHOG hog, where in every, at every step, you divided the image into parts and constructed a HOG representation for each part individually, and then concatenated, all of them to form a representation.

How do you do this? You divide an image into 2^l x 2^l cells at each permit level l. So, if l was 1, you would divide it into two cross, the entire image into 2 x 2 cells, 4 cells, or if l was 2 you would divide it into 4 cross 4 cross so on and so forth. Now for each of these cells you extract HOG descriptors just the way we spoke about on the previous slide and then you concatenate the HOGs at different pyramid levels to give an entire histogram of oriented gradients representation.

So as you can see, this can be done in several ways, you would get a histogram of oriented gradients for each cell and concatenate all of them and then you would get a histogram of oriented gradients at l is equals 0, l is equal to 1, and you can concatenate all of this also to get your final representation.

So this method was shown to capture spatial relationships between say objects a little bit better than HOG, which does not do this at a pyramid level.

(Refer Slide Time: 17:33)



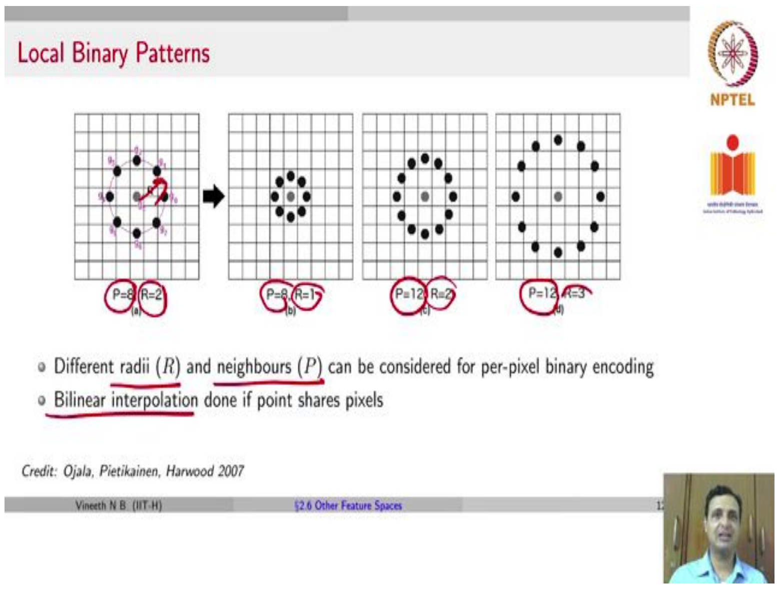
Local Binary Patterns

Another popular method, which was extensively used for face images in particular human faces, was known as local binary patterns. Local binary patterns had an interesting idea where given an image for every pixel you take, for example, a 3 cross 3 neighborhood around a pixel, so this is the IIT Hyderabad logo. So you consider a pixel and a 3 cross 3 neighborhood around the pixel write down the intensity values of those pixels. Now, with respect to the central pixel you decide if each of the other pixels around it were lower or higher.

If it was lower, you would set a 0, if it is equal or higher, you can set it to 1, that now gives you a binary pattern around the central pixel. Once you define a canonical representation, which means you say I will start at the top left, and then go in a circular manner, you can define a binary representation for this particular pixel.

So once again, I will repeat that. So you define a binary representation based on how each pixel is related to the central pixels in density. So now, this value is now 15 in decimal and you replace that pixel's value with 15, other values around it would similarly be obtained by doing an LBP positioned at that particular pixel and so on and so forth. Now, what do you do with this new LBP representation?

(Refer Slide Time: 19:14)


Local Binary Patterns

- Different radii (R) and neighbours (P) can be considered for per-pixel binary encoding
- Bilinear interpolation done if point shares pixels

Credit: Ojala, Pietikainen, Harwood 2007

Vineeth N B (IIT-H) §2.6 Other Feature Spaces

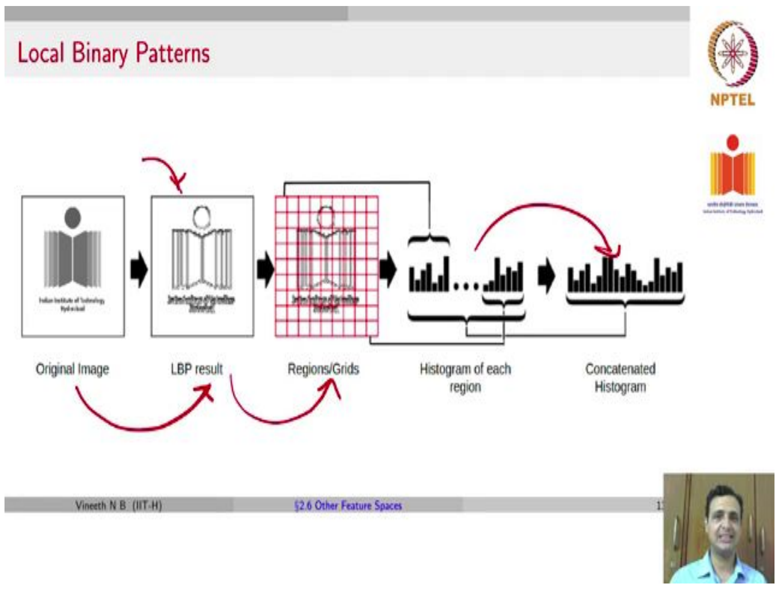
Before we go there, you can also define neighborhoods in multiple ways. We said you would take a 3 by 3 neighborhood and take the 8 pixels around it that may not be always the case, you can vary two different parameters here, you can define your radius r you can also define the number of neighbors P.

So you can have a particular radius where you want to compute your local binary pattern with that may not be the immediate neighbor. It can be a radius of four, five pixels around the central pixel. And you can also define how many neighbors you want at that radius. So you could define p is equal to 8, the number of neighbors is 8 and r is equal to 2, as you can see, that is the example that you see on this particular image.

Similarly, you could have a closer neighborhood, r is equal to 1, but the number of neighbors to be same p is equal to 8. Similarly, you could have an r is equal to 2, you could increase the number of neighbors and make it p is equal to 12 and at r is equal to 3 further out, you could still have 12 neighbors.

Once you define the neighbors, if these neighbors lie in between two pixels, you can use bilinear interpolation to get those values. So once you define R and P, based on those values, you would get a binary pattern around the central pixel. You write that out in a circular manner as a binary number, convert it into decimal and replace the central pixel with that particular value. What do you do with that?

(Refer Slide Time: 20:51)

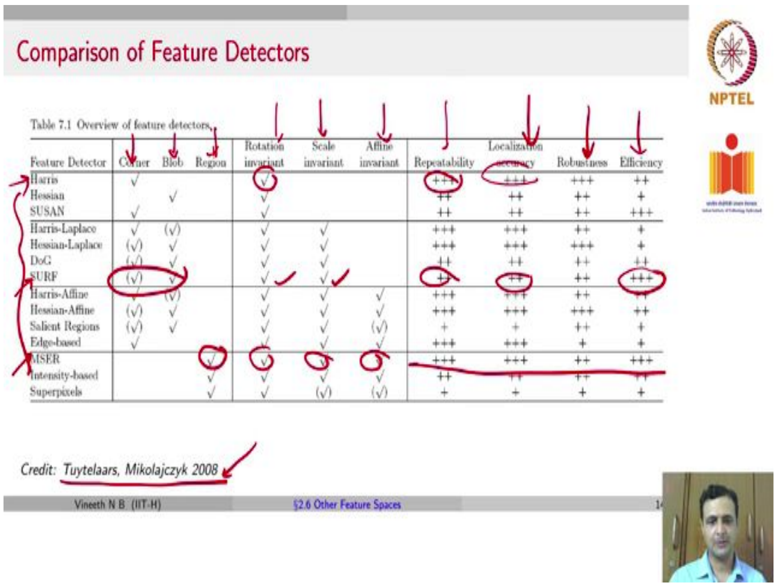

Local Binary Patterns

So once you get an LBP result, which is your LBP intensity for each pixel, remember the previous process that we talked about in the earlier slide gives you one decimal for each pixel around which you considered the neighborhood. So you could now construct such an image for every pixel, you could repeat the same thing and construct such an LBP image.

Once you do this, you divide the LBP result into regions or grids and once again, you can get a histogram of each region, the number of pixels in each region in this particular case, and concatenate all of them to be able to get a single representation for the region or an entire image.

In this case, we are not considering the gradients it is simply the histogram, but you could also improve LBP to consider gradients in each cell, so on and so forth. There have been several improvements of LBP or local binary patterns that have considered these kinds of variations.

(Refer Slide Time: 21:54)



Comparison of Feature Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Credit: Tuytelaars, Mikolajczyk 2008

Vineeth N B (IIT-H) §2.6 Other Feature Spaces 14

So here is a quick summary of various feature detectors. We have seen some of them. We may not have the scope to see all of them, but you are welcome to look at the source of this particular table to understand more features, but you can view all features in terms of whether they are corner detectors, blob detectors or do they help in finding regions whether they are rotation invariant, scale invariant, affine invariant. How repeatable are they remember, repeatability is about ensuring that if you found a particular pixel to be a corner, in a particular image, if you have the same image taken from a different angle, in another picture or same scene taken as a picture in another angle that corner should also be detected as a corner in the second image. This is known as repeatability.

How good is the localization accuracy? Remember, we talked about localization accuracy even for the canny edge detector where we said, is the point exactly where it should be or is it a few pixels away from the actual corner that we know. How robust is the method to noise and how fast is the method?

So these are various parameters in which, using which you can evaluate the goodness of the methods that we have discussed so far. So the Harris corner detector, let us take a few of them and just go over them. The Harris is a corner detector, it is rotation invariant, as we have seen not necessarily scale or affine invariant, as we already saw.

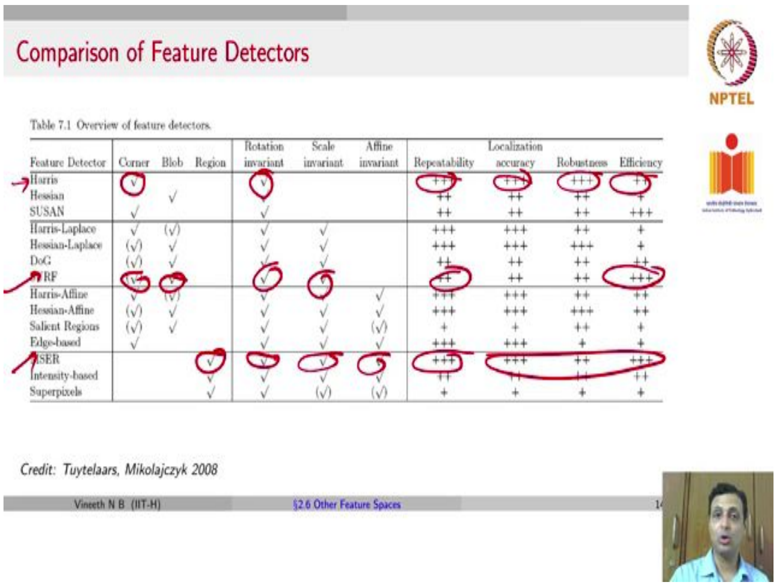
It is fairly repeatable, it is reasonably good with localization, accuracy, robustness, as well as efficiency. Let us take another one. Let us consider SURF, which is an improvement over

SIFT that we saw a couple of lectures back, which is a corner detector or a blob detector, it can be used either way.

It is rotation invariant, and scale invariant that is the purpose of sift and SURF to be scale invariant. It is in the name of sift itself. And it is also fairly repeatable fairly accurate in terms of localization, robustness and is extremely efficient. We know we saw that SURF is almost three times as fast as sift.

So you can similarly look at say MSER, which is the one that we just saw a few slides ago is a region detector, which is reasonably rotation invariant, scale invariant and affine invariant and it is also is repeatable, accurate in terms of localization, robustness and efficiency. More details of other feature detectors please look at this in the references, and you can look at the paper to get a more detailed survey of other feature extractors.

(Refer Slide Time: 24:43)



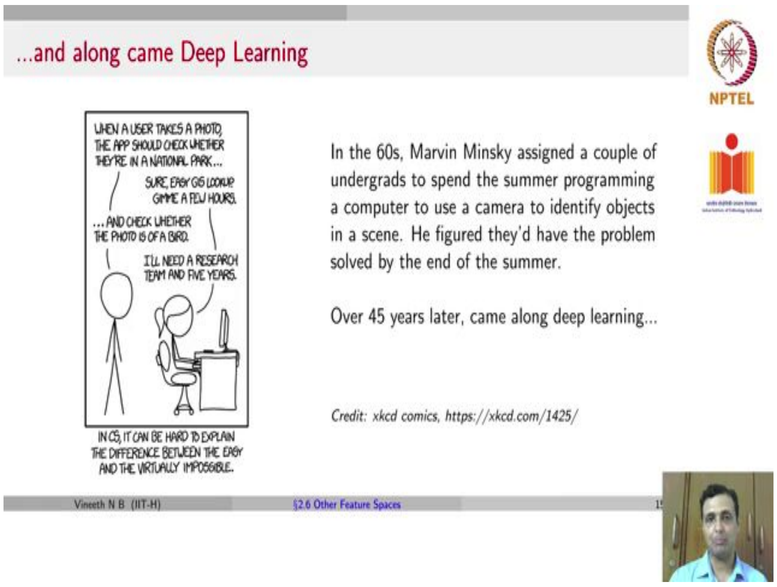
Comparison of Feature Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Credit: Tuytelaars, Mikolajczyk 2008

Vineeth N B (IIT-H) §2.6 Other Feature Spaces 14



...and along came Deep Learning

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

In the 60s, Marvin Minsky assigned a couple of undergrads to spend the summer programming a computer to use a camera to identify objects in a scene. He figured they'd have the problem solved by the end of the summer.

Over 45 years later, came along deep learning...

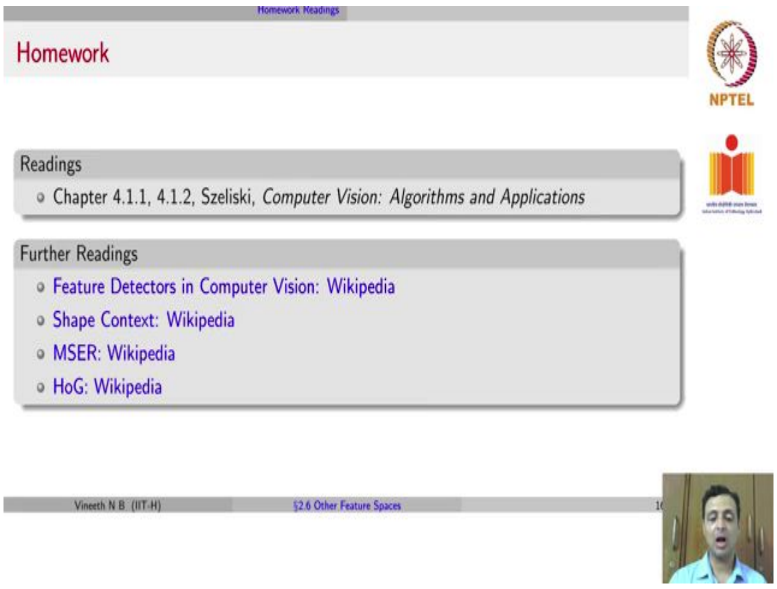
Credit: xkcd comics, https://xkcd.com/1425/

Vineeth N B (IIT-H) §2.6 Other Feature Spaces 15

Why are we not discussing each of these is what we are going to talk about next, which is a lot of representations of images that used to be used in the pre deep learning era have, no longer relevant because of the success of deep learning in extracting features that are extremely good for various computer vision tasks.

Remember when we talked about the history of deep learning, we said that in the mid-1960s, Marvin Minsky started out with a summer project to be able to solve the problem of computer vision, but many decades later deep learning has been able to significantly make that progress at this time.

(Refer Slide Time: 25:34)


Homework Readings

Homework

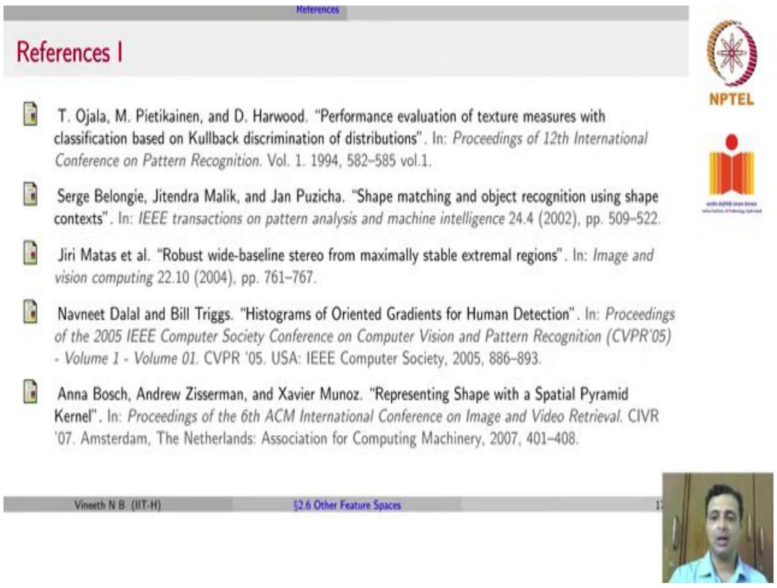
Readings

- Chapter 4.1.1, 4.1.2, Szeliski, *Computer Vision: Algorithms and Applications*

Further Readings

- Feature Detectors in Computer Vision: Wikipedia
- Shape Context: Wikipedia
- MSER: Wikipedia
- HoG: Wikipedia

Vineeth N B (IIT-H) §2.6 Other Feature Spaces 16


References

References I

T. Ojala, M. Pietikainen, and D. Harwood. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions". In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. 1994, 582–585 vol.1.

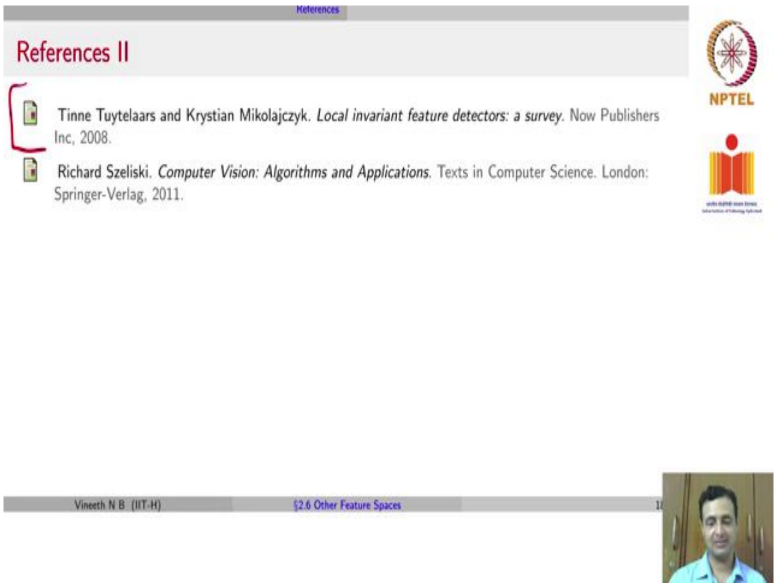
Serge Belongie, Jitendra Malik, and Jan Puzicha. "Shape matching and object recognition using shape contexts". In: *IEEE transactions on pattern analysis and machine intelligence* 24.4 (2002), pp. 509–522.

Jiri Matas et al. "Robust wide-baseline stereo from maximally stable extremal regions". In: *Image and vision computing* 22.10 (2004), pp. 761–767.

Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. USA: IEEE Computer Society, 2005, 886–893.

Anna Bosch, Andrew Zisserman, and Xavier Munoz. "Representing Shape with a Spatial Pyramid Kernel". In: *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*. CIVR '07. Amsterdam, The Netherlands: Association for Computing Machinery, 2007, 401–408.

Vineeth N B (IIT-H) §2.6 Other Feature Spaces 17


References

References II

Tinne Tuytelaars and Krystian Mikolajczyk. *Local invariant feature detectors: a survey*. Now Publishers Inc, 2008.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.

Vineeth N B (IIT-H) §2.6 Other Feature Spaces 1

So the homework for this is going to be reading chapters 4.1.1 and 4.1.2. If you are further interested about other feature detectors, please do read some of these links, as well as some of these references. In particular, this reference gives an overview of various different feature detectors if you are interested.