SIFT Detector

Shree K. Nayar

Monograph: FPCV-2-3

Module: Features

Series: First Principles of Computer Vision

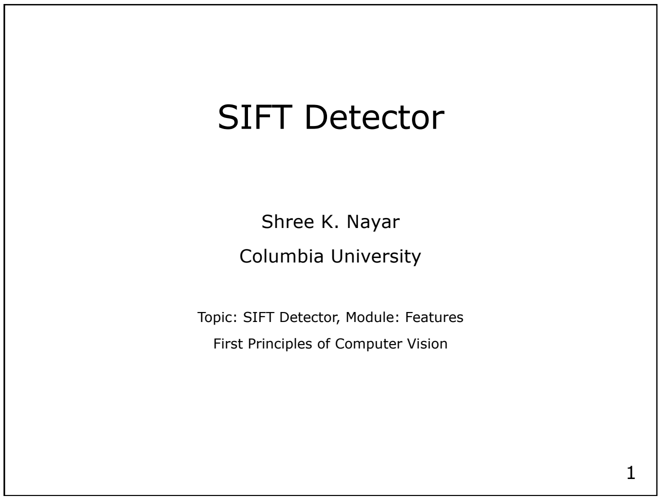
Computer Science, Columbia University

August 31, 2022

FPCV Channel

FPCV Website

We know how to find edges and corners in images, and how to use them to compute object boundaries. This approach is very useful in computer vision. Now let us discuss the problem of recognizing objects, particularly those with complex appearances. In this case, we want to be able to detect and match features that are more intricate than edges and corners. That is what the Scale Invariant Feature Transform (SIFT) does for us.



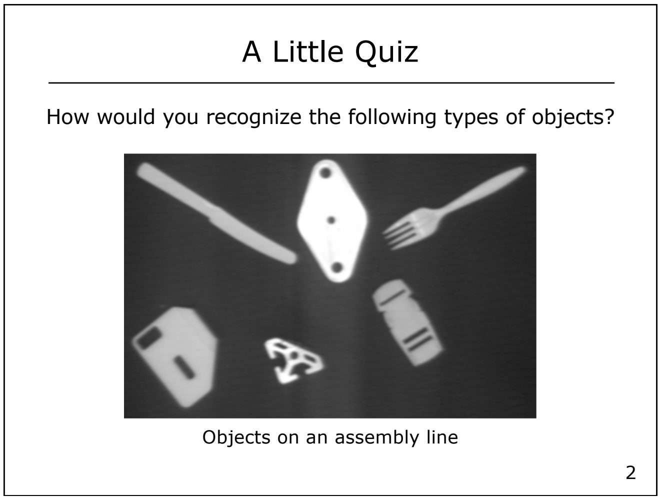
SIFT Detector

Shree K. Nayar
Columbia University

Topic: SIFT Detector, Module: Features
First Principles of Computer Vision

1

Let us start with a little quiz. For this image, what technique would we use to recognize the objects? This is a fairly simple case. From the lecture on binary image processing, we know that we can apply a threshold to get a clean binary image. We can then compute the geometric properties of the objects to recognize them, and find their positions and orientations as well.



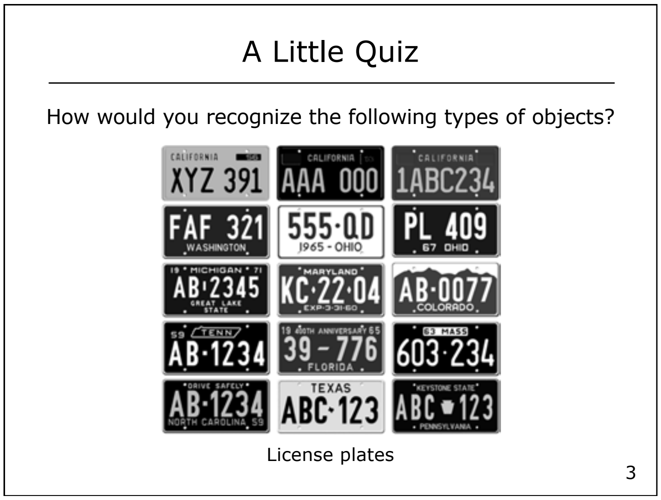
A Little Quiz

How would you recognize the following types of objects?

Objects on an assembly line

2

What about this image? While it may appear somewhat more complex, we can still apply some kind of a thresholding scheme to extract the letters and numbers on the license plates. We can then, once again, use our binary image processing methods to recognize the characters and read the license plates.

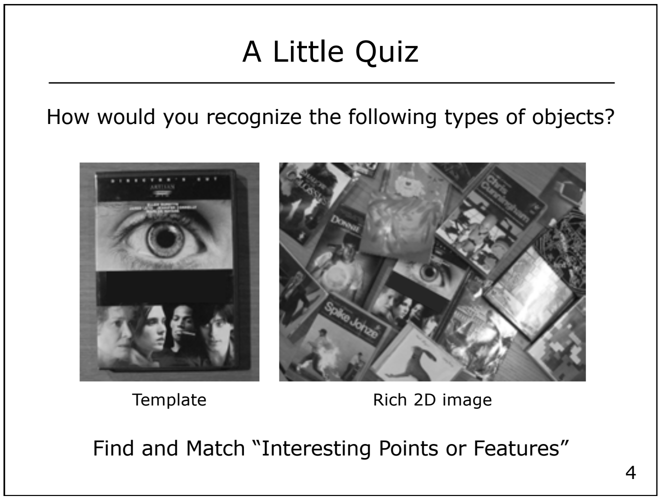


A Little Quiz

How would you recognize the following types of objects?

License plates

3



A Little Quiz

How would you recognize the following types of objects?

Template

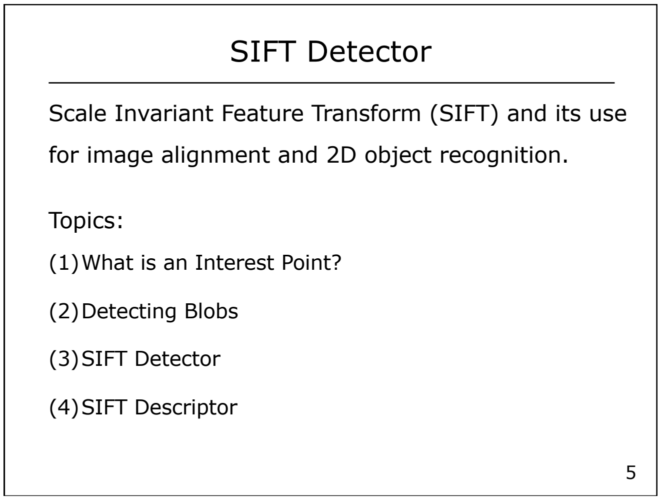
Rich 2D image

Find and Match “Interesting Points or Features”

4

Now let’s consider a more interesting case. Here we have a 2D object on the left, a planar object, but one with a fairly complex appearance. We want to find this object in the image on the right. What technique should we use? Our first instinct may be to apply template matching, which we know how to do using normalized correlation. The problem, however, is that the object does not actually show up in its original form — it is rotated and magnified differently in the image on the right. In order to deal with rotation and scale, we would have to create many templates of the object under a variety of rotations and scales and then apply template matching using each one of the templates. Such an approach would be computationally impractical, especially when the number of objects of interest is large.

There is an even harder problem we need to cope with here, which is occlusion. We can see that the object is partially obstructed by other objects. To deal with this using template matching would require us to create a lot of little (partial) templates of the object. After applying each one, we would then have to make sure that, in our final output, there are enough of the object’s templates that appear in the image and that they lie in a certain geometric configuration. Such an approach, again, would not scale well with the number of objects. Instead, what we would like to do is directly extract highly descriptive features from the image of the object on the left. If we can then find many of these features in the image on the right, and their appear in the same relative configuration with respect to each other, then we have found the object. In fact, we can compute its rotation and scale as well. That is exactly what the SIFT detector enables us to do.



SIFT Detector

Scale Invariant Feature Transform (SIFT) and its use for image alignment and 2D object recognition.

Topics:

(1) What is an Interest Point?

(2) Detecting Blobs

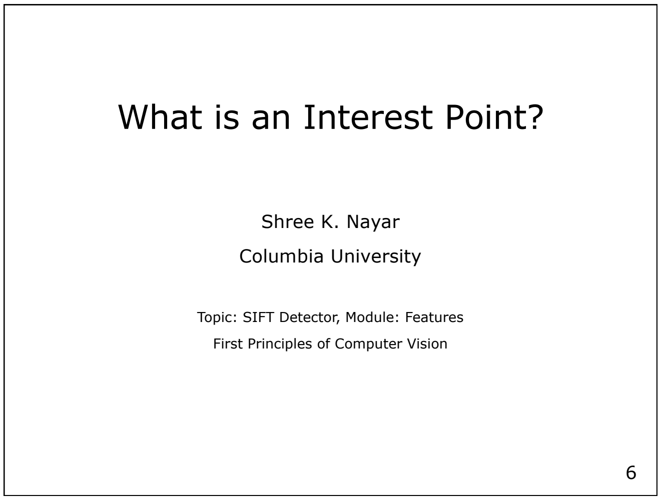
(3) SIFT Detector

(4) SIFT Descriptor

5

We will describe the theory behind the SIFT detector, its implementation, and how it is used to solve vision problems such as image stitching and object recognition. We are going to start by asking the question: what is an interesting point in an image? This is an area of computer vision research that has a long history. The concept of an “interest point” was first put forward by Hans Moravec in the late 1970s. Since then, there has been much work in this area that suggests that edges and corners are not interesting enough in many real-world applications. We need features that are more descriptive. Over time, it was realized that the notion of a "blob," which has some local appearance within it, is potentially a good interest point. Here, we use the term blob loosely – it is simply a patch with a well-defined local appearance. As in the case of edge detection, we will use image derivatives to develop a theory of blob detection. This approach has the advantage that it can detect blobs over multiple scales, or sizes, of the object. We have to be able to deal with scale if we would like to allow the object of interest to lie at any depth with respect to the camera.

Using this theory, we will develop the SIFT detector, which was proposed by David Lowe. We will see how the SIFT detector is implemented in practice. Once SIFT features are detected in an image, we want to be able to match them with features in other images. For this, we need to extract from each feature some kind of signature that describes the local appearance around the location of the feature. That brings us to the SIFT descriptor. We need to ensure that the descriptor is rotation and scale invariant so it can be used for matching. We also need to ensure that it is insensitive to other factors such as the illumination of the scene.

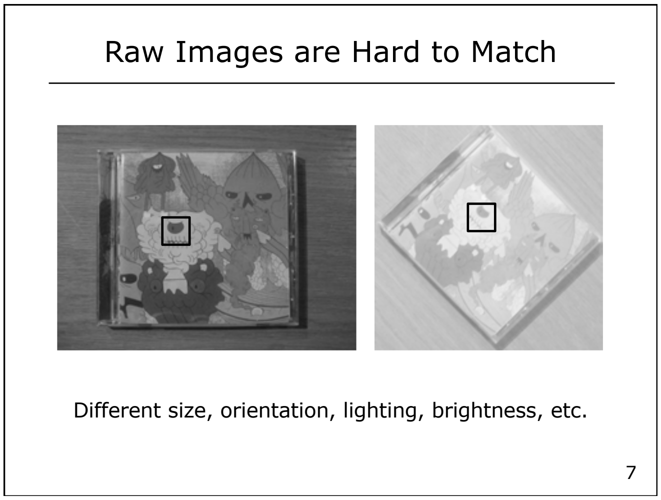


What is an Interest Point?

Shree K. Nayar
Columbia University

Topic: SIFT Detector, Module: Features
First Principles of Computer Vision

6



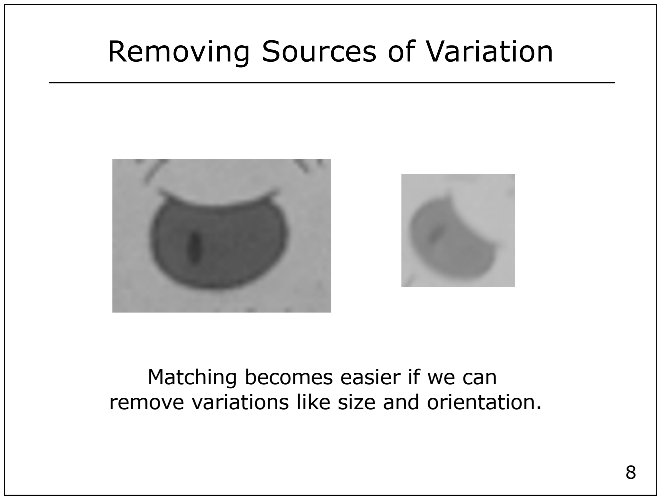
Raw Images are Hard to Match

Different size, orientation, lighting, brightness, etc.

7

Let us discuss what really makes a point interesting in an image. Shown here are two images of an object taken under different conditions. If we consider the outlined patch, we see that there is a difference in size due to the different magnifications of the object in the two images. There are also differences in orientation and lighting. Any interest point detector that we develop should be able to either compensate for, or remove, these variations. Only then can we match a feature in one image to a feature in another image.

In other words, matching interest points becomes easier if we can remove variations due to magnification, orientation and lighting.

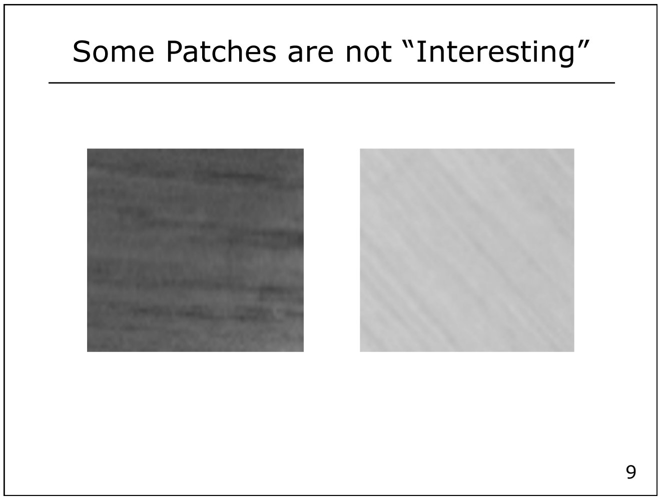


Removing Sources of Variation

Matching becomes easier if we can
remove variations like size and orientation.

8

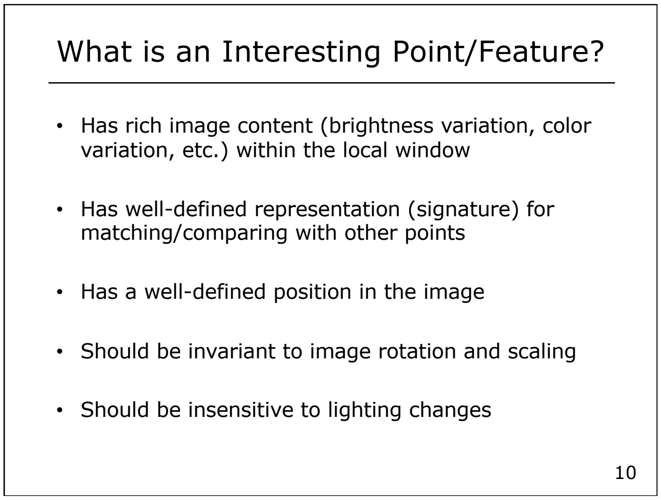
When developing an interest point detector, we do not want it to respond to every patch in an image. In fact, most patches in an image are simply not interesting. Consider the patches shown here – they include weak textures but they are neither unique nor interesting enough to use for matching.



Some Patches are not “Interesting”

9

With the above discussion in mind, let us list some of the desirable attributes of an interesting point. First, it needs to have rich image content around it in terms of brightness and color variation, such that there is a certain degree of uniqueness that can be exploited while matching. Second, it should have a well-defined representation, meaning that we should be able to compute a signature from the appearance around it. Third, it must have a well-defined position — when we recognize an object, we typically want to know where it is located. Fourth, its signature should be invariant to rotation and scaling. Finally, it should be insensitive to the illumination of the scene.

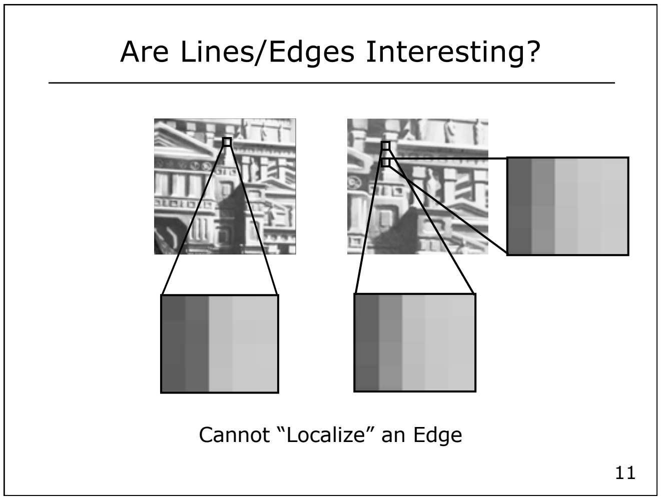


What is an Interesting Point/Feature?

- Has rich image content (brightness variation, color variation, etc.) within the local window
- Has well-defined representation (signature) for matching/comparing with other points
- Has a well-defined position in the image
- Should be invariant to image rotation and scaling
- Should be insensitive to lighting changes

10

Given that we know how to find edges, we may ask why don't we use them as interest points. Shown on the left is an image with one of its edges highlighted. On the right is a slightly more magnified version of the same scene. In this image it is easy to see that there are numerous edges that appear similar to the one in the left image. In short, edges are not unique, or descriptive, enough to use for matching.



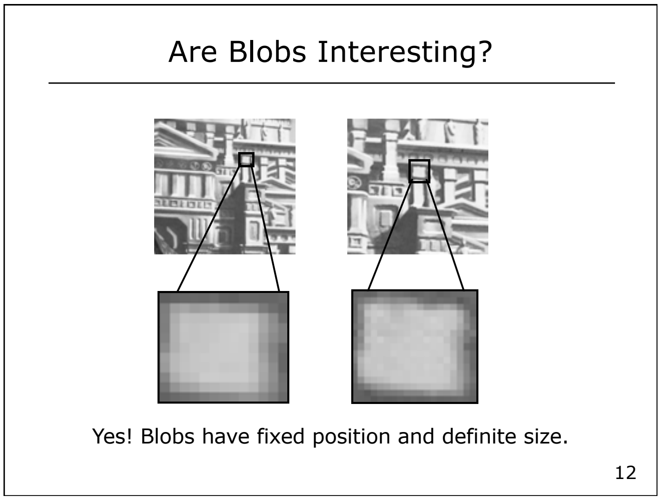
Are Lines/Edges Interesting?

Cannot "Localize" an Edge

11

This might lead us to believe that perhaps corners are good interest points. Indeed, corners are interesting and they have been used for object recognition and other tasks. However, they tend to be useful only in applications that involve simple objects — they are not descriptive enough for recognizing complex objects.

Now let us take a look at something else. Shown here is a patch in the image on the left, which we will call a blob. On the right, is the same blob as it appears in the magnified image on the right. In the image patches shown in the bottom, the two blobs are normalized in scale and we see that they look similar. Even this simple blob is useful as it has some kind of a local appearance due to the brightness variation within it. Unlike an edge, the position of the blob is well-defined. For these reasons, a blob is a good candidate for an interest point.

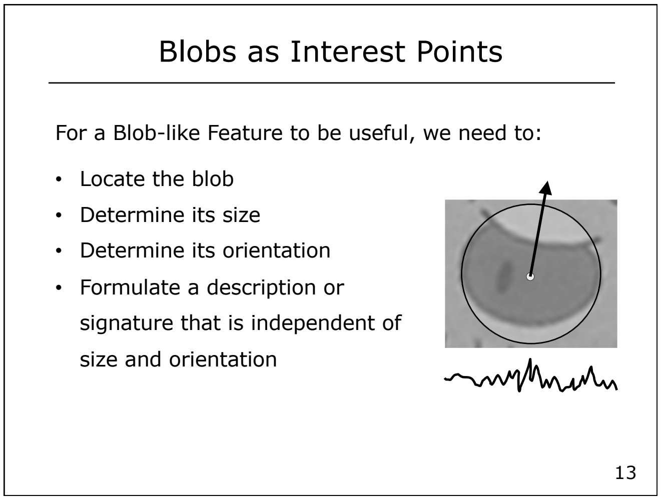


Are Blobs Interesting?

Yes! Blobs have fixed position and definite size.

12

There are certain criteria for a blob-like feature to be useful. First, we need to be able to locate the blob, meaning there should be a position associated with it. The position is not necessarily related to any particular feature inside the blob, but rather emerges from the way we perform blob detection. Second, it needs to have a size associated with it. Size is an interesting concept, as it has nothing to do with the edges or boundaries inside the blob. It is simply the rough scale of its appearance, a fairly abstract concept, as we shall see. Third, since the blob itself can appear in different orientations, we need to be able to compute a principal orientation for it. Finally, we need to be able to extract a description — a signature — which relates to the appearance of the blob. As mentioned earlier, this signature is crucial for matching purposes.



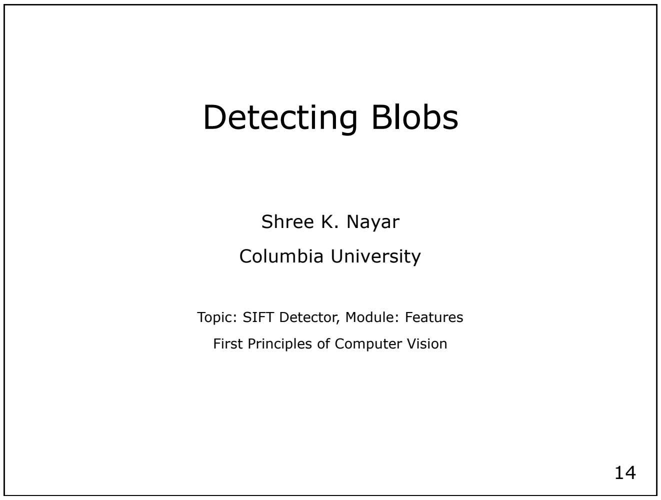
Blobs as Interest Points

For a Blob-like Feature to be useful, we need to:

- Locate the blob
- Determine its size
- Determine its orientation
- Formulate a description or signature that is independent of size and orientation

13

Now, let us look at how we can find blobs in an image. We are going to develop a technique that uses the derivatives of images, so we can use some of the tools that we developed for detecting edges. As with edges, we are going to start with a 1D signal and then extend our analysis to 2D.



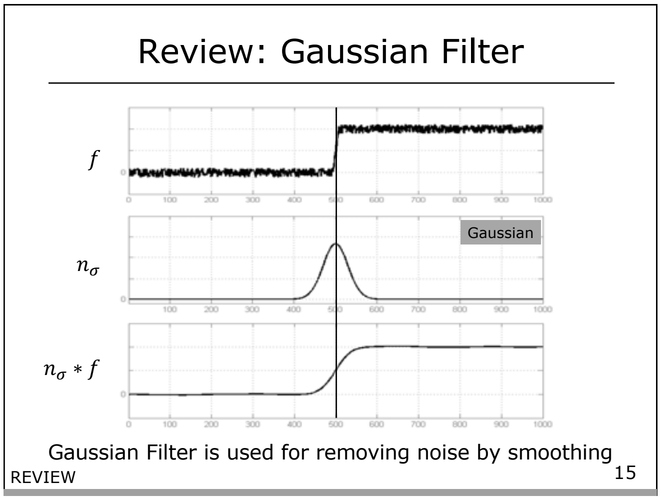
Detecting Blobs

Shree K. Nayar
Columbia University

Topic: SIFT Detector, Module: Features
First Principles of Computer Vision

14

First, we will review some of the tools we used for edge detection. Shown here on top is our 1D image, a noisy signal with an edge where the vertical line is overlaid. To reduce noise, we convolve it with the Gaussian shown in the middle to get the result in the bottom, which is a signal that is more or less free of noise, but one with the edge blurred.

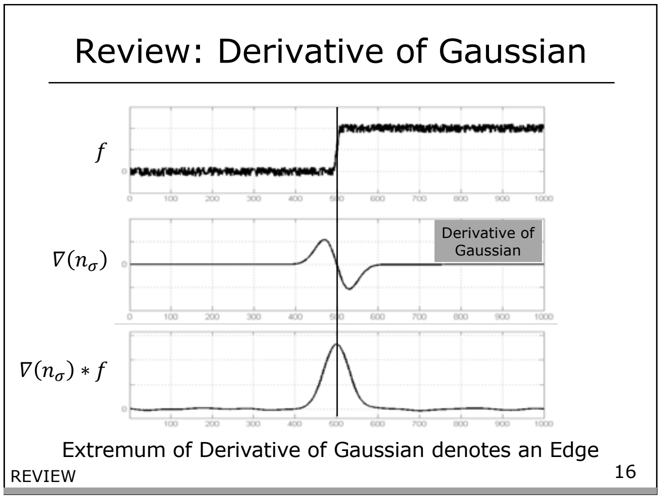


Review: Gaussian Filter

Gaussian Filter is used for removing noise by smoothing

REVIEW 15

Our second tool involves using the first derivative to find the position of an edge. Instead of applying the first derivative to the image, we can find the first derivative of the Gaussian, shown in the middle, and convolve the image with this function. We end up with the output shown at the bottom, where we get a clear peak at the location of the edge.

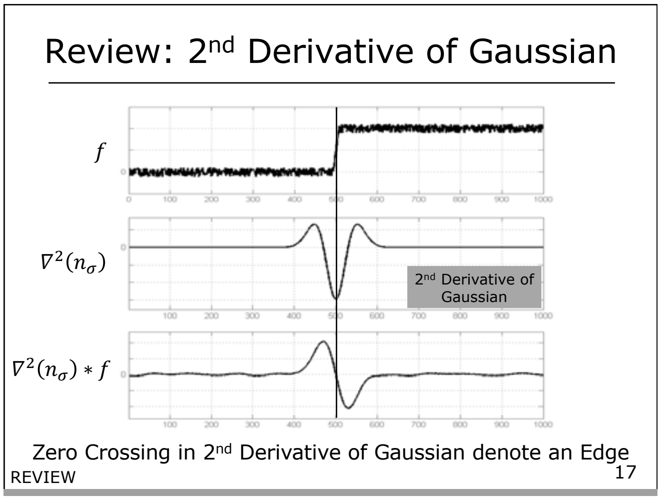


Review: Derivative of Gaussian

Extremum of Derivative of Gaussian denotes an Edge

REVIEW 16

As we know from our lecture on edge detection, we can also locate an edge by using the second derivative. Again, instead of finding the second derivative of the image, we find the second derivative of the Gaussian, which is shown in the middle. This is referred to as the inverted Mexican hat operator. When we apply this operator to the noisy input signal, we get the result at the bottom, which has a zero-crossing at the location of the edge.

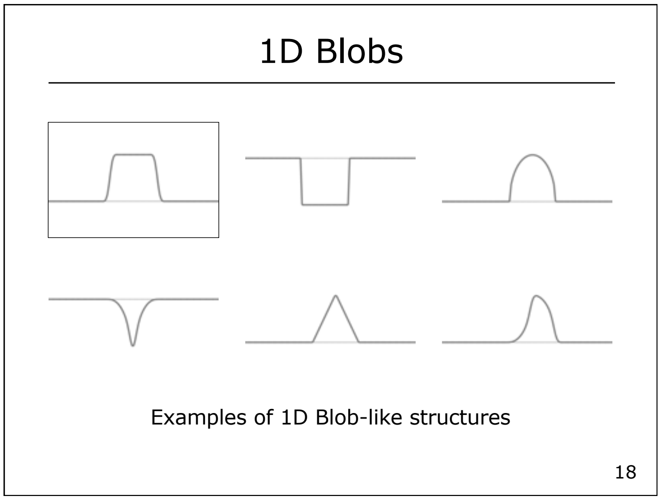


Review: 2nd Derivative of Gaussian

Zero Crossing in 2nd Derivative of Gaussian denote an Edge

REVIEW 17

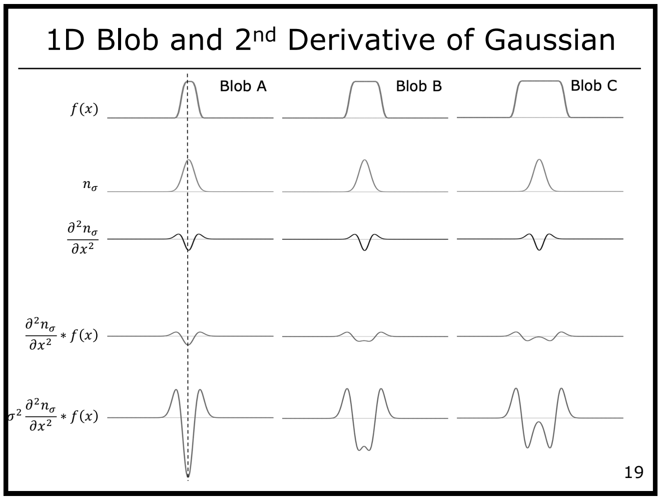
Now, let’s return to blobs. Shown here are some 1D blobs of different shapes. We want to find all of these as blobs in the image. For our analysis here, we will use the simple model shown in the top left corner. Note, however, that the SIFT detector can be used to find all the blobs shown here and more.



1D Blobs

Examples of 1D Blob-like structures

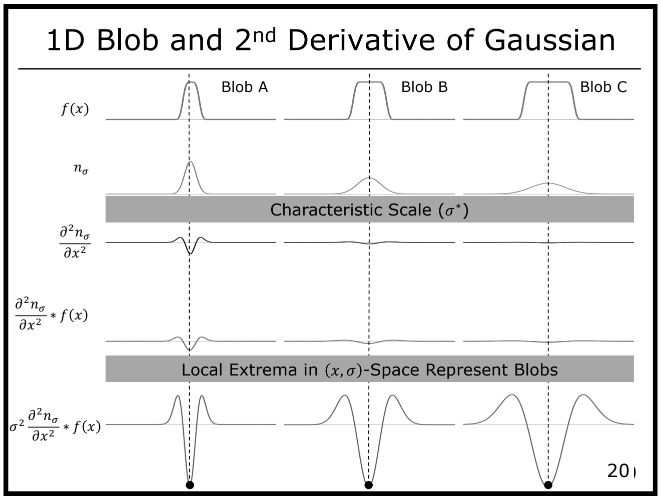
18


1D Blob and 2nd Derivative of Gaussian

First, we will look at a case where we have three blobs, A, B, and C, where blob B is twice the width of blob A, and blob C is three times as wide as blob A. Note that they all have the same maximum values. We will use the σ of the Gaussian to explore what is called a “scale space” associated with the input image.

If we apply the second derivative of the Gaussian shown in the third row to the image in the first row, we end up with the output shown in the fourth row. Note that we get two distinct zero-crossings in the output for blob C, but since blobs A and B are thinner, in these cases, the two zero-crossings overlap.

Note that the strength of the output shown in the fourth row depends on the value of σ we use for the Gaussian. As σ gets wider, the peak value of the output will decrease. To address this problem, we are going to multiply the second derivative of the Gaussian with σ^2. The resulting output, shown in the fifth row, is what is called a σ-normalized output. In our case, all that does is change the scale, or the amplitude, of the responses for all the three blobs.


1D Blob and 2nd Derivative of Gaussian

Characteristic Scale (σ^*)

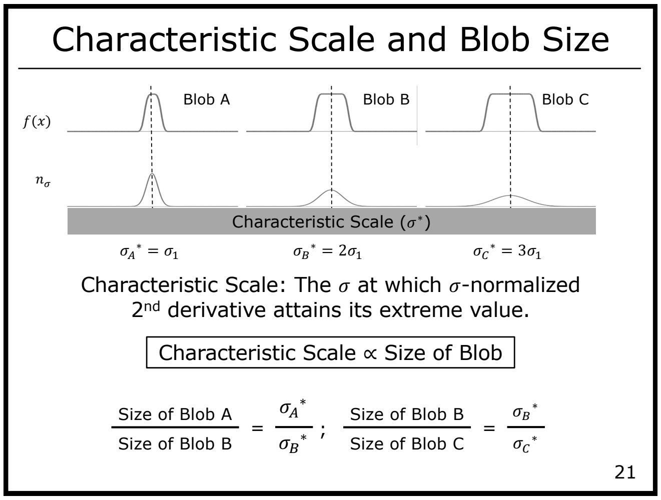
Local Extrema in (x,σ)-Space Represent Blobs

Now let us see what happens when we change the σ of the Gaussian. If we increase it, we see that the response (fifth row) for blob A gets much sharper, with a distinct peak exactly where the center of the blob lies. If we can find this peak, we have found the blob. To find blobs B and C, we continue to increase σ until the peaks for blob B and C appear as well. Note that, as we increase σ in order to find B and C, the response for A will decrease. That is, for each blob, as we increase σ, a peak will emerge and then fade away.

Thus, our approach is to apply the second derivative of the Gaussian using multiple values of σ, which is referred to as the scale parameter. Irrespective of the size of the blob, at some scale, the output will attain a maximum value at the location of the blob. Note that the scales at which we obtain peaks for the three blobs are proportional to the widths of the blobs themselves. At this point, we have a stack of output images corresponding to different scales. This output can be represented with two parameters:

x as the spatial coordinate and *σ* as the scale. We going to identify local extrema in this two-dimensional output. The *x* values where the extrema lie correspond to the locations of the blobs, and the values of *σ* at the extrema correspond to the sizes of the blobs. Shown in the second row are the widths of the Gaussians that produced the maximum outputs (fifth row) for the three blobs. The *σ* values corresponding to these Gaussians are called the characteristic scales of the blobs.

Note that we are not just finding a blob, but also estimating its size or characteristic scale. Continuing with our example signal with three blobs, we find that blob A's characteristic scale is σ_1 while blob B's is exactly two times σ_1, since the width of B is twice that of A. For blob C, we get three times σ_1.



Characteristic Scale and Blob Size

$\sigma_A{}^* = \sigma_1$ $\sigma_B{}^* = 2\sigma_1$ $\sigma_C{}^* = 3\sigma_1$

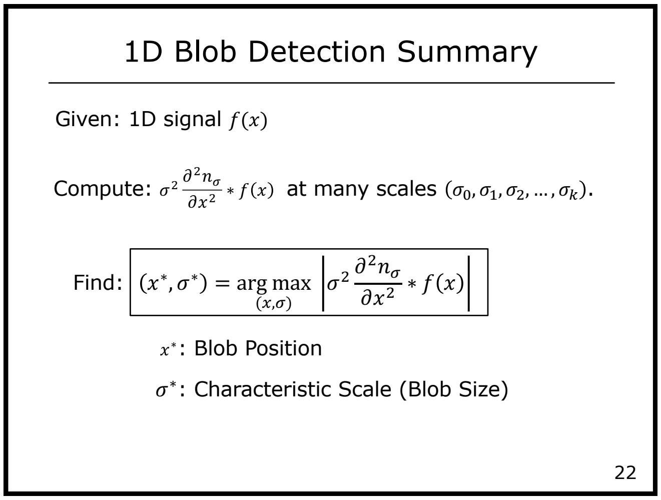
Characteristic Scale: The σ at which σ-normalized 2nd derivative attains its extreme value.

$\boxed{\text{Characteristic Scale} \propto \text{Size of Blob}}$

$$\frac{\text{Size of Blob A}}{\text{Size of Blob B}} = \frac{\sigma_A{}^*}{\sigma_B{}^*}\ ;\ \frac{\text{Size of Blob B}}{\text{Size of Blob C}} = \frac{\sigma_B{}^*}{\sigma_C{}^*}$$

21

In summary, to do blob detection in 1D, we take the 1D image and apply the *σ*-normalized second derivative of the Gaussian at many different scales. This produces an image stack in which we can find the (*x*,*σ*) pairs where the local extrema are located. Here, we denote x^* as the position of a blob and σ^* as its characteristic scale.



1D Blob Detection Summary

Given: 1D signal $f(x)$

Compute: $\sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x)$ at many scales $(\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_k)$.

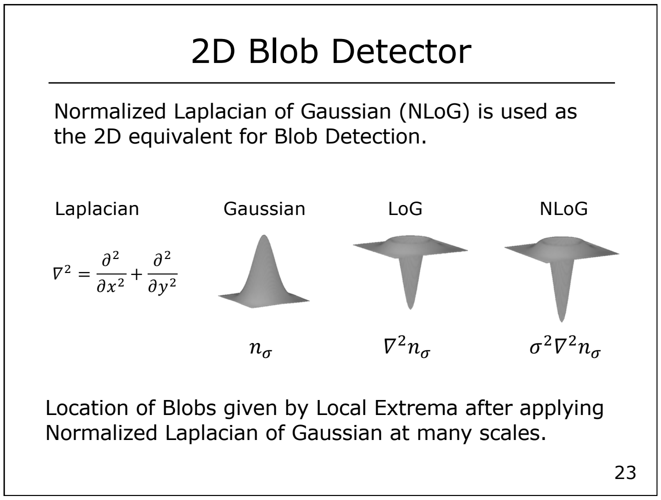
Find: $\boxed{(x^*, \sigma^*) = \underset{(x,\sigma)}{\arg\max} \left| \sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x) \right|}$

x^*: Blob Position

σ^*: Characteristic Scale (Blob Size)

22

It is easy to extend this approach to 2D images. Here we will use the σ-normalized Laplacian of the Gaussian, which is called the NLoG operator. We end up with a stack of images where x and y correspond to the spatial coordinates and the scale σ corresponds to the third dimension. Our goal then is to locate all the local extrema in this stack.



2D Blob Detector

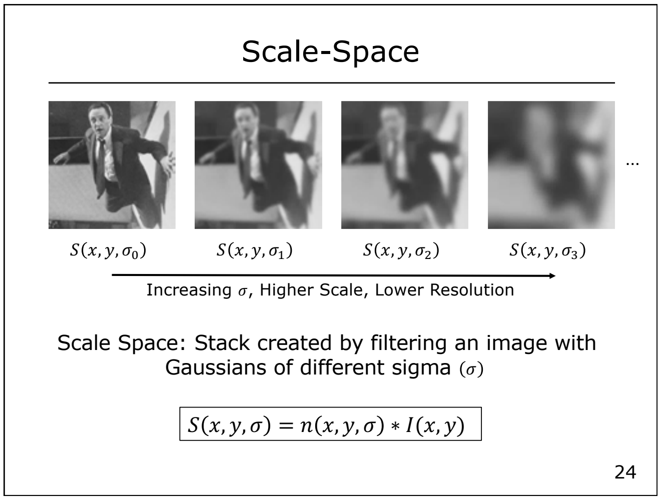
Normalized Laplacian of Gaussian (NLoG) is used as the 2D equivalent for Blob Detection.

Laplacian	Gaussian	LoG	NLoG
$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$	n_σ	$\nabla^2 n_\sigma$	$\sigma^2 \nabla^2 n_\sigma$

Location of Blobs given by Local Extrema after applying Normalized Laplacian of Gaussian at many scales.

23

Let us explore the scale space associated with an image, which is essentially a stack of images where the original image $I(x,y)$ is convolved with Gaussians with different σ values. Such a stack can be denoted as $S(x,y,\sigma)$. Shown here are four images from such a stack, where σ increases from left to right. In effect, the resolution of the image decreases as σ increases.



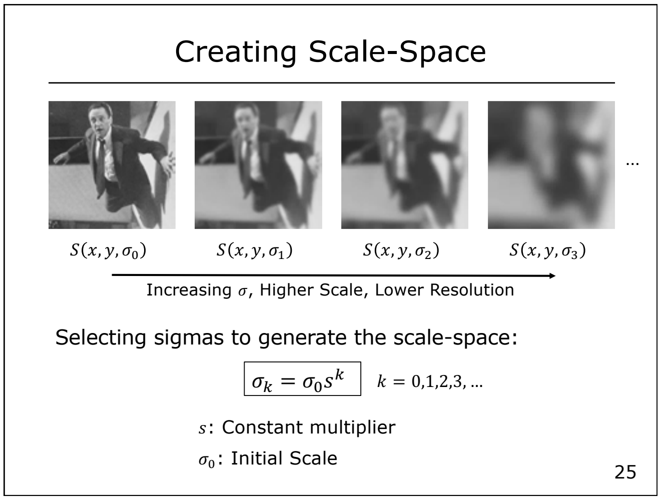
Scale-Space

Scale Space: Stack created by filtering an image with Gaussians of different sigma (σ)

$$\boxed{S(x, y, \sigma) = n(x, y, \sigma) * I(x, y)}$$

24

Since, in practice, $S(x,y,\sigma)$ has to be represented discretely, we need to decide on which discrete values of σ we should use. A convenient approach is to use the expression shown here, where each discrete scale is a constant s multiplied by the previous scale, and σ_0 is the first scale.



Creating Scale-Space

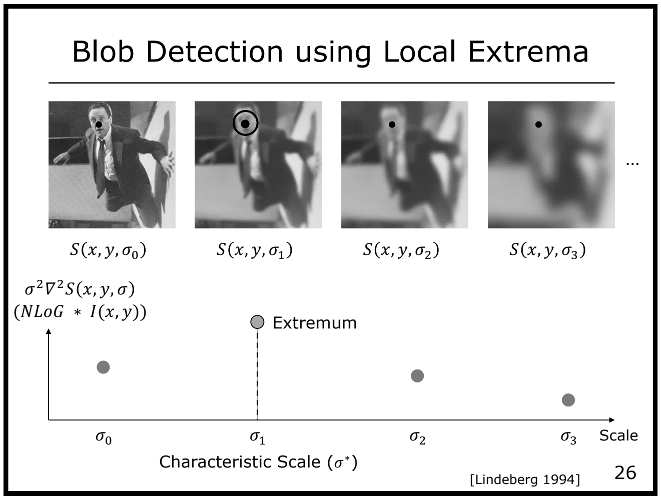
Selecting sigmas to generate the scale-space:

$$\boxed{\sigma_k = \sigma_0 s^k} \quad k = 0,1,2,3, \ldots$$

s: Constant multiplier

σ_0: Initial Scale

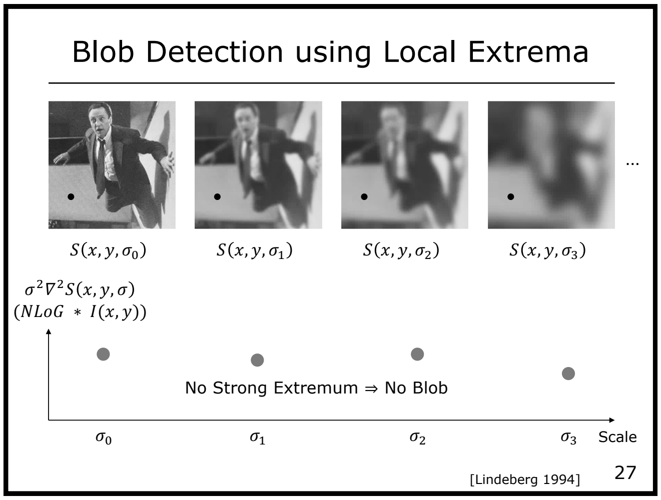
25



Blob Detection using Local Extrema

[Lindeberg 1994] 26

Let us take a look at a couple of points in the original image and see how each one responds to the NLoG operator. Consider this point (black dot), which is relatively rich in terms of its surrounding content. In the bottom, are NLoG operator outputs for the point plotted as a function of scale. Note that a clear maximum emerges at σ_1. A circle with radius proportional to σ_1 is drawn in the corresponding image. If we take a close look at the image content within the circle, we see that the circle itself is not close to a real boundary of any kind in the image. In other words, the blob we have found is not really one in the usual sense of the word, which is, an image patch with uniform brightness and a clear boundary around it. It is a blob in a more abstract sense. It is difficult to simply look at an image and know exactly where the blobs will end up being detected and how large they will be.

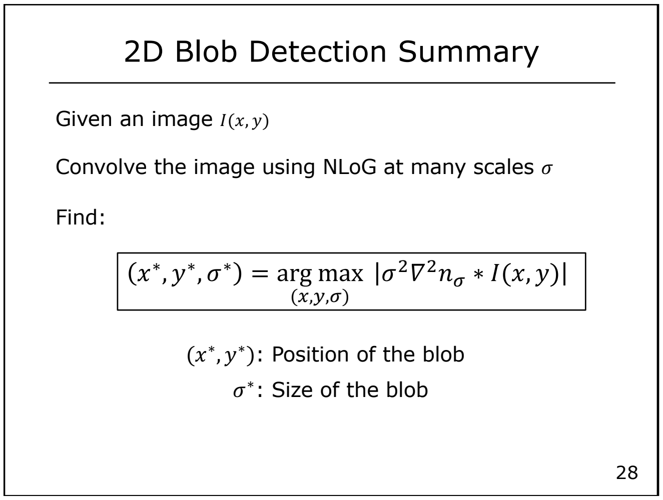


Blob Detection using Local Extrema

[Lindeberg 1994] 27

If we choose another point on a flat region, since the surrounding of the point has little image variation, we end up with low operator values at all scales. That is, the point does not produce a clear extremum and hence will not be detected as the center of a blob.

We can now summarize blob detection in 2D. Given an image *I(x, y),* we convolve the image with the NLoG operator at many different scales, using our simple formula for picking the scales. What we end up with is essentially a stack of images. We will then find local extrema in the stack. The position *(x,y)* of each extremum corresponds to the position of a blob and the corresponding σ represents the size of the blob.



2D Blob Detection Summary

Given an image $I(x,y)$

Convolve the image using NLoG at many scales σ

Find:

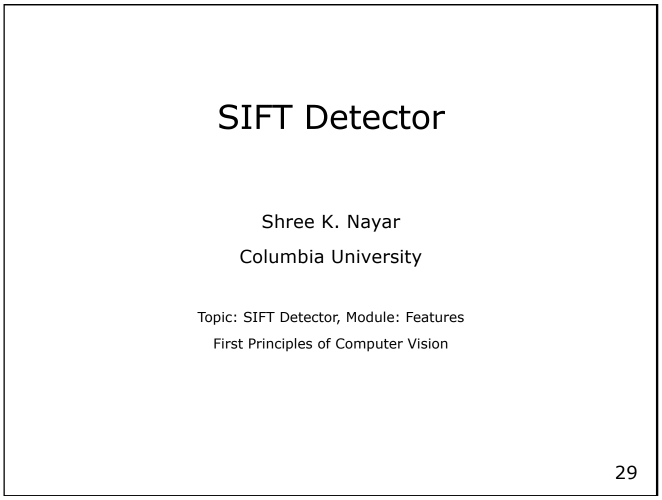
$$\boxed{(x^*,y^*,\sigma^*) = \underset{(x,y,\sigma)}{\arg\max}\ |\sigma^2\nabla^2 n_\sigma * I(x,y)|}$$

(x^*,y^*): Position of the blob

σ^*: Size of the blob

28

We now have all the tools needed to develop the SIFT detector. It uses a few tricks to make it both reliable and efficient.



SIFT Detector

Shree K. Nayar

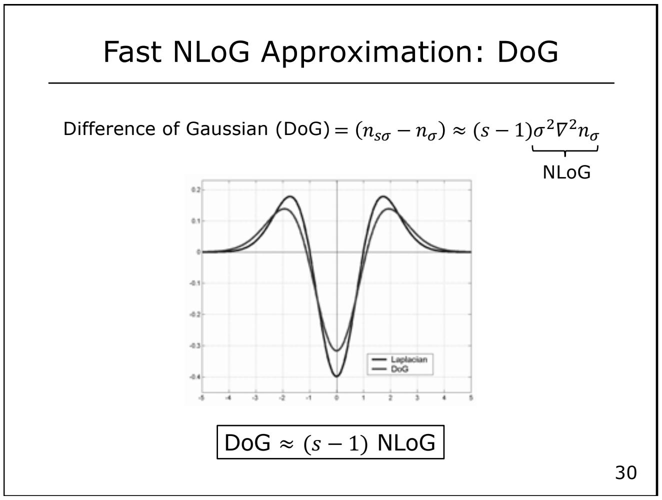
Columbia University

Topic: SIFT Detector, Module: Features

First Principles of Computer Vision

29

Let us take a look at one of these tricks: an approximation to the NLoG operator. If we subtract a Gaussian from a second Gaussian with a sigma that is the sigma of the first one multiplied by a scalar *s*, it turns out that we get an operator, which we can call the Difference of Gaussian (DoG), that is a pretty good approximation of the NLoG operator multiplied by a scale factor (*s*–1). Therefore, given our image stack *S*(*x*,*y*,σ) which represents the scale space of an input image, we can estimate the output of the NLoG operator for one scale by simply finding the difference between two consecutive images in the stack.



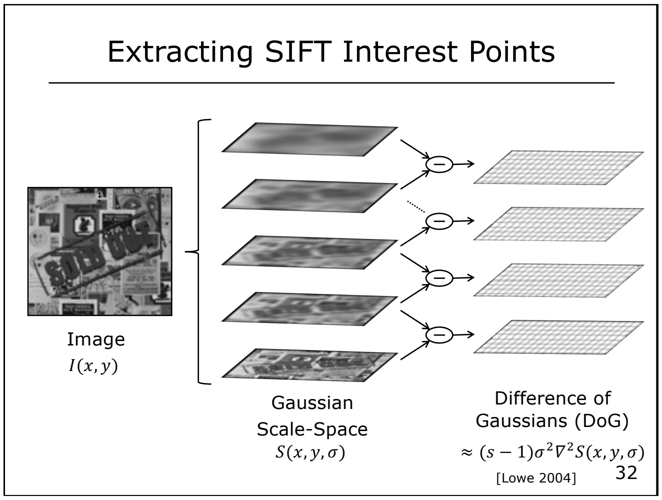
Fast NLoG Approximation: DoG

Difference of Gaussian (DoG) $= (n_{s\sigma} - n_\sigma) \approx (s-1)\underbrace{\sigma^2\nabla^2 n_\sigma}_{\text{NLoG}}$

$$\boxed{\text{DoG} \approx (s-1)\ \text{NLoG}}$$

30

Let’s now see how SIFT is implemented. Given an image, we first create the stack $S(x,y,\sigma)$ of images by convolving the original image with Gaussians of increasing width. In order apply the NLoG operator to $S(x,y,\sigma)$, we will use the trick we just discussed and find the difference between all pairs of consecutive images in $S(x,y,\sigma)$. This results in the stack shown on the right.



Extracting SIFT Interest Points

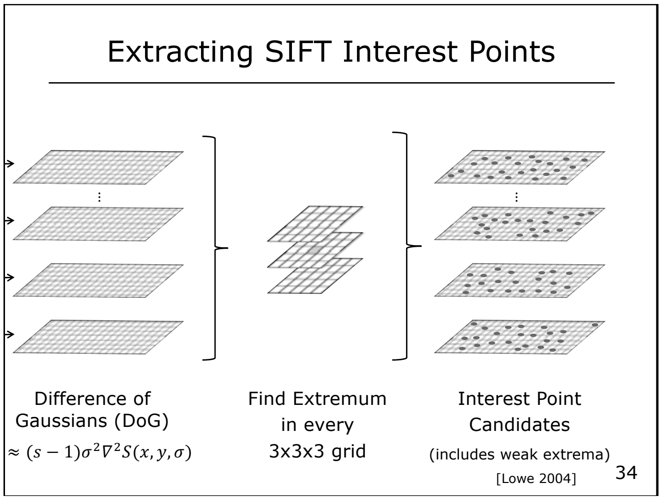
Image $I(x,y)$

Gaussian Scale-Space $S(x,y,\sigma)$

Difference of Gaussians (DoG) $\approx (s-1)\sigma^2\nabla^2 S(x,y,\sigma)$

[Lowe 2004] 32

Next, we need to find extrema in this stack of images. We can do this in many different ways. One way is to treat the stack as a volume and run a small *nxnxn* window over the entire volume. At each location, if the absolute value of center pixel is significantly larger than the absolute values of its neighbors, it is declared to be an extremum. This is similar to the non-maximal suppression algorithm we used for corner detection.



Extracting SIFT Interest Points

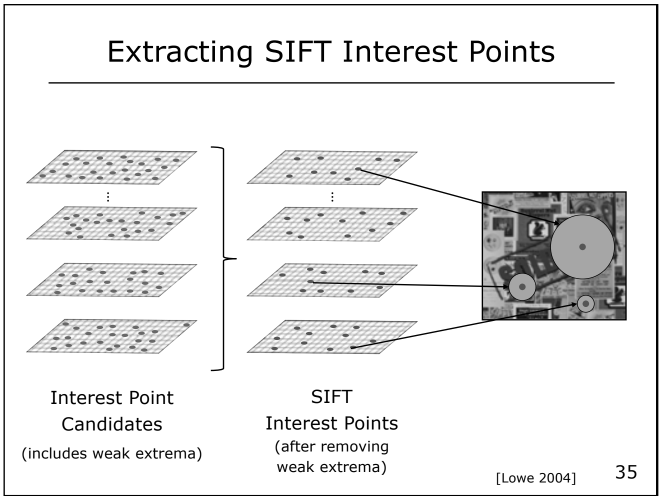
Difference of Gaussians (DoG) $\approx (s-1)\sigma^2\nabla^2 S(x,y,\sigma)$

Find Extremum in every 3x3x3 grid

Interest Point Candidates (includes weak extrema)

[Lowe 2004] 34

To suppress the effects of noise, we can use a threshold to filter out weak extrema. The resulting extrema are the detected interest points. These are our candidates for SIFT features at many different scales. On the right is shown the input image with just three of the features, detected at three different scales, overlaid as circles on the image. The center of each circle corresponds to the location of the feature and the radius of the circle is proportional to its size. It is worth mentioning that while we have highlighted only three features, many more were detected for this image.



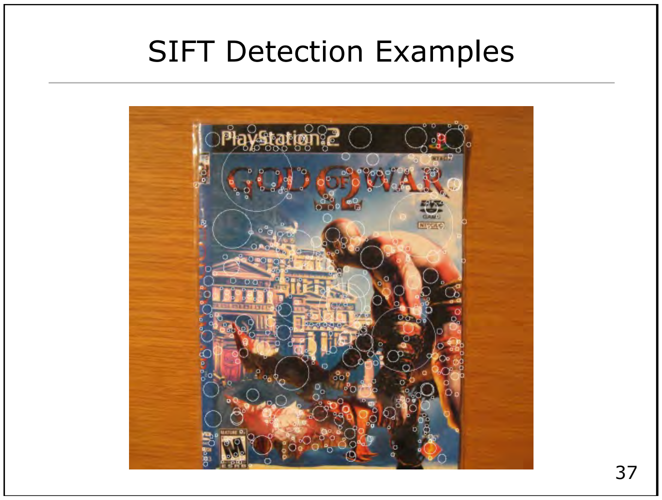
Extracting SIFT Interest Points

Interest Point Candidates (includes weak extrema)

SIFT Interest Points (after removing weak extrema)

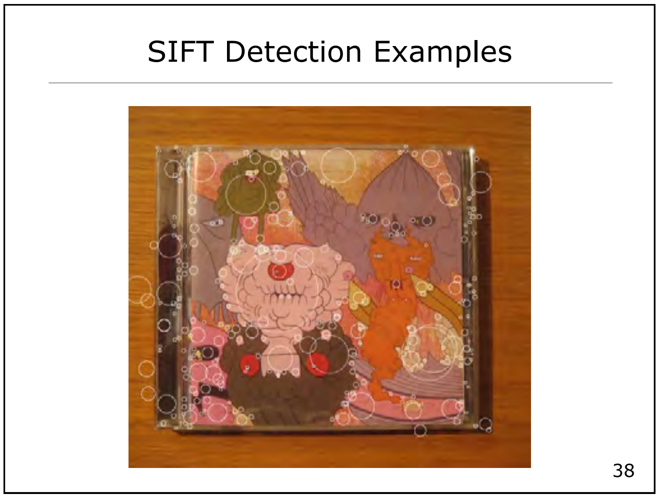
[Lowe 2004] 35

Here are the results of applying the SIFT detector to images of two objects. Again, every circle drawn on the image is a SIFT interest point. Note that not all of these are obviously blobs. That is, the detected blobs do not necessarily correspond to our notion of a blob. This does not matter as long as the features can be robustly detected.



SIFT Detection Examples

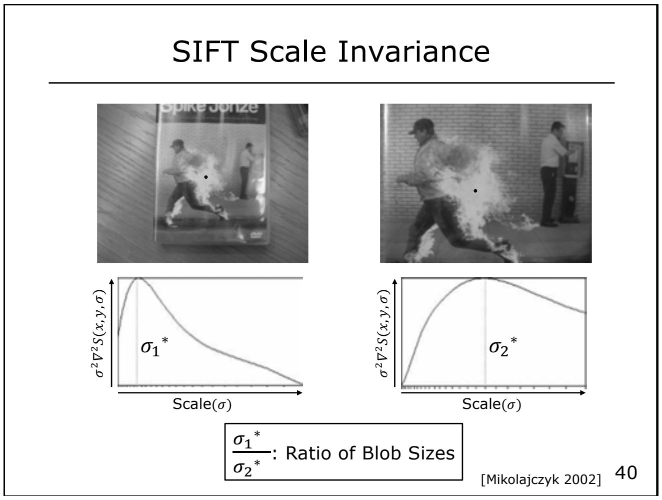
37



SIFT Detection Examples

38

We now know how to find interest points. Let's discuss how to make SIFT scale invariant, so it can be used for matching. On the left is an image of an object, and on the right is a close-up image of the same object. Consider the NLoG operator applied to a single point (the black dot). At the bottom are shown the outputs of the NLoG operator for the points in the two images, plotted as a function of scale. Since the object has different magnifications in the two images, the NLoG operator peaks at different sigma values. Since we know these sigma values, we can normalize for scale, which means, we can compensate for scale before we match two SIFT features.

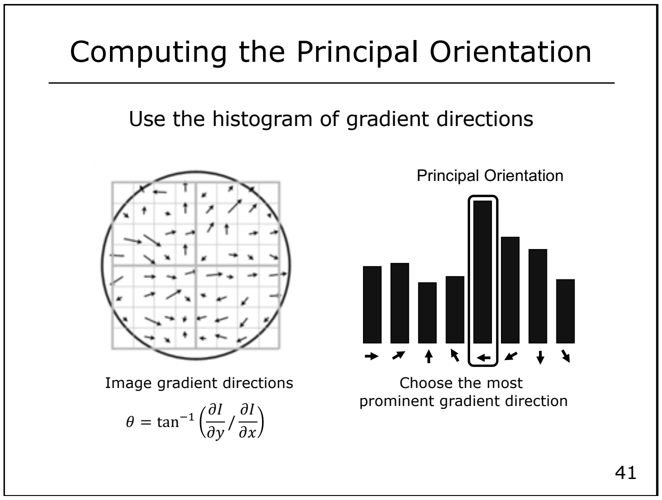


SIFT Scale Invariance

$\frac{\sigma_1{}^*}{\sigma_2{}^*}$: Ratio of Blob Sizes

[Mikolajczyk 2002] 40

The other aspect we need to worry about is orientation, or rotation. Imagine that, after normalizing for scale, the circle shown here is the area that the blob occupies. We consider a square window of pixels that substantially overlaps the circle. We apply the gradient operator to all the pixels within the window, which gives us the magnitude and the orientation of the gradient at each pixel. Since the magnitude also depends on factors such as the camera gain and the illumination of the scene, we ignore the magnitude and only look at the orientation of the gradient. We create a histogram, shown on the right, where the *x*-axis corresponds to the different directions for the gradient and the bar corresponding to each gradient direction is the number of pixels in the window on the left that have that direction. The gradient direction corresponding to the maximum value in the histogram is what we refer to as the principal orientation of the feature.



Computing the Principal Orientation

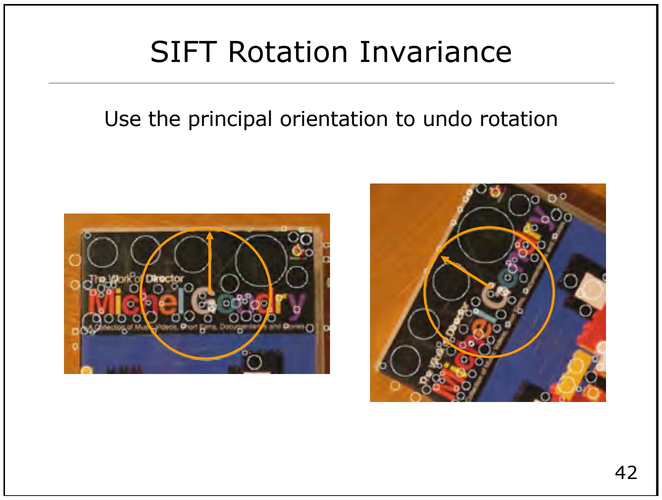
Use the histogram of gradient directions

Principal Orientation
Image gradient directions
Choose the most prominent gradient direction

$\theta = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$

41

The characteristic scale of the feature is used to rescale the image patch that represent the feature, and its principal orientation is used to reorient the feature such that the principal orientation points North. At this point, we are in a position to match the contents within any two SIFT features.



SIFT Rotation Invariance

Use the principal orientation to undo rotation

42

SIFT Descriptor

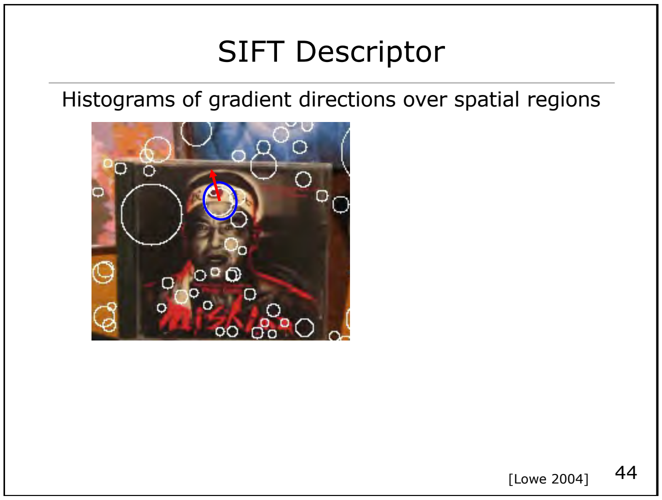
Shree K. Nayar

Columbia University

Topic: SIFT Detector, Module: Features

First Principles of Computer Vision

43

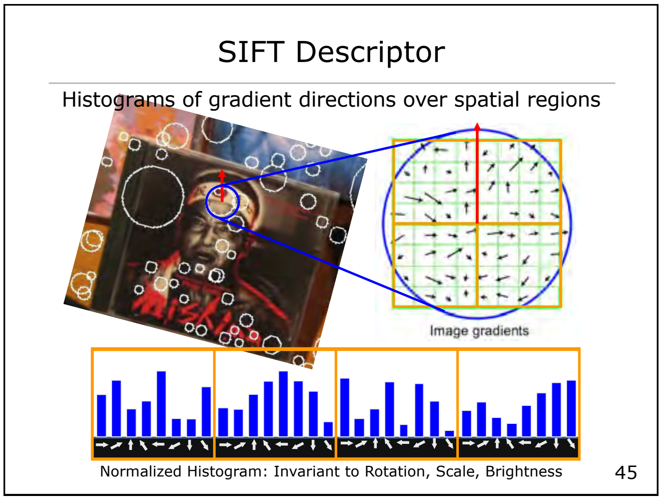


SIFT Descriptor

Histograms of gradient directions over spatial regions

[Lowe 2004] 44

To match a feature with another, we need to extract a concise signature that represents the visual information within the feature. Consider the image shown on the right. We will focus on the feature that is shown as a blue circle.



SIFT Descriptor

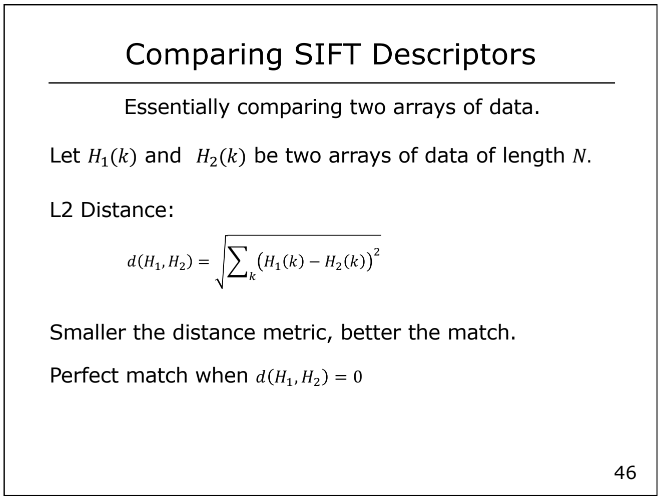
Histograms of gradient directions over spatial regions

Normalized Histogram: Invariant to Rotation, Scale, Brightness

45

On the right is shown a scale and rotation normalized window of pixels that corresponds to the above SIFT feature. We compute the gradient at each pixel and, as we did before, we ignore the magnitude and retain the orientation at each pixel. Next, we compute the gradient orientation histogram of each of the four quadrants of the grid and concatenate them to obtain the histogram shown at the bottom of the slide. We refer to this as the normalized histogram of the feature as it is invariant to scale, rotation and other effects such as illumination. This normalized histogram is called the SIFT descriptor. It is used to match one SIFT feature with another.

There are a few different metrics we can use to match two SIFT descriptors. Let H_1 and H_2 be the normalized histograms of two features. A simple approach would be to compute the L2 distance between two normalized histograms. We subtract the second histogram from the first one, bin for bin, square the differences, sum them, and find the square root of the sum to get the L2 distance. If this distance is zero, then we have a perfect match between two descriptors.



Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N.

L2 Distance:

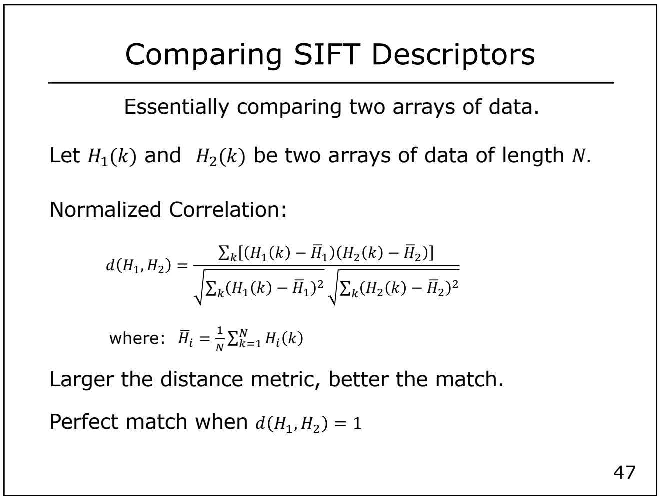
$$d(H_1, H_2) = \sqrt{\sum_k \left(H_1(k) - H_2(k)\right)^2}$$

Smaller the distance metric, better the match.

Perfect match when $d(H_1, H_2) = 0$

46

Another way to compare descriptors is by using normalized correlation, which we discussed in the context of template matching. In this case, we subtract the means from each of the histograms, take the product of the two resulting histograms, find the sum over all the bins, and divide the result by the “energies” of the two histograms. If the normalized correlation is one, we have a perfect match.



Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N.

Normalized Correlation:

$$d(H_1, H_2) = \frac{\sum_k [(H_1(k) - \bar{H}_1)(H_2(k) - \bar{H}_2)]}{\sqrt{\sum_k (H_1(k) - \bar{H}_1)^2}\sqrt{\sum_k (H_2(k) - \bar{H}_2)^2}}$$

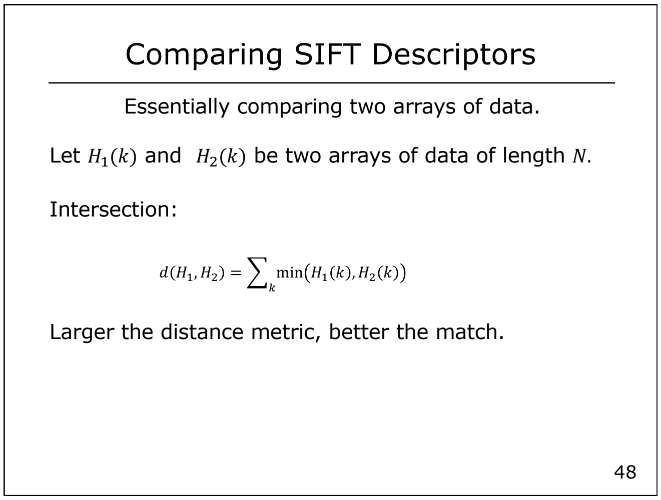
where: $\bar{H}_i = \frac{1}{N}\sum_{k=1}^{N} H_i(k)$

Larger the distance metric, better the match.

Perfect match when $d(H_1, H_2) = 1$

47

Another metric that is useful in many settings is the intersection metric. In this case, we compute for each bin the minimum of the two histograms and take the sum over all bins. In this case, we are, in effect, computing the “overlap” between the two histograms. The larger the value of the result, the better the match.



Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N.

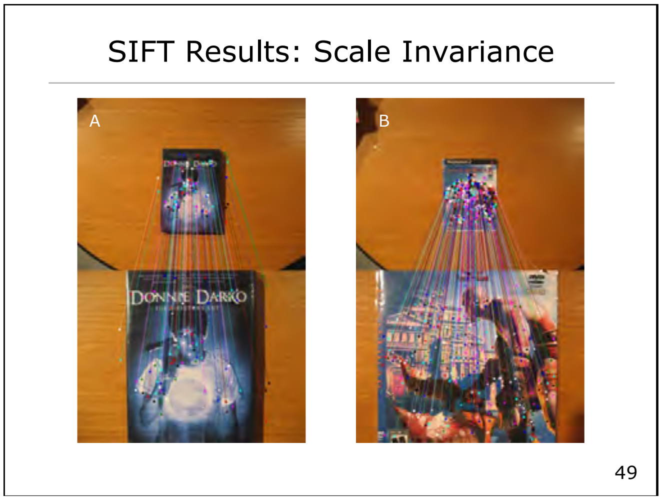
Intersection:

$$d(H_1, H_2) = \sum_k \min\left(H_1(k), H_2(k)\right)$$

Larger the distance metric, better the match.

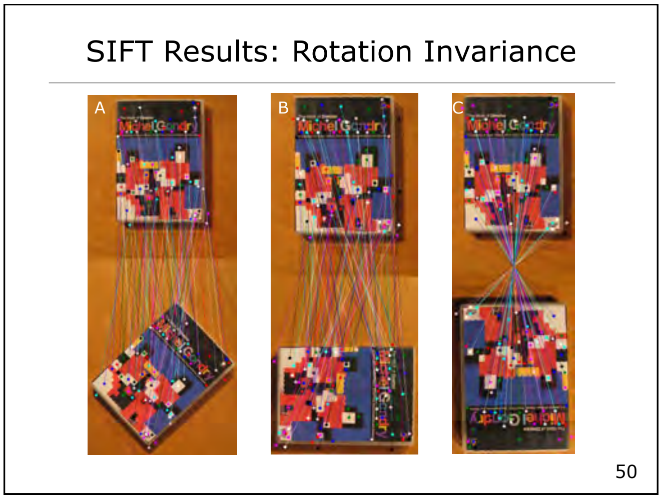
48

Let us take a look at how well the SIFT detector and descriptor work in practice. First, let us explore the scale invariance. On the left are two images of the same object but with different scales. Each one of the lines drawn corresponds to a successful match. On the right is another example. Once again, there are plenty of matches. SIFT is able to find many matches despite the fact that the scale changes between the images are large.



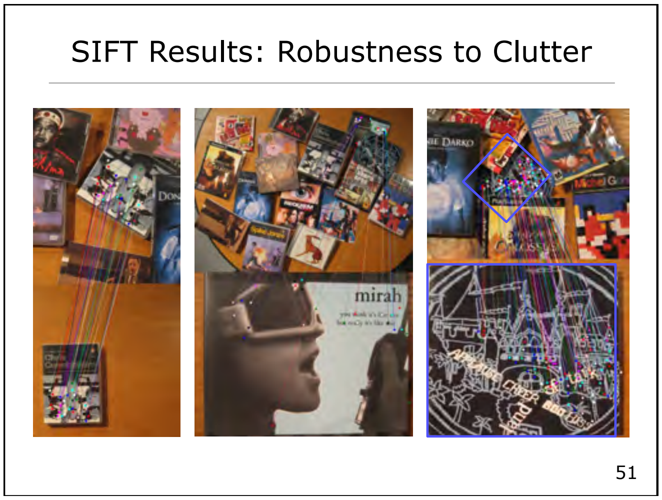
SIFT Results: Scale Invariance

Now let us examine rotation invariance. Here, the object is rotated with respect to the reference image by 45 degrees (left), 90 degrees (middle) and 180 degrees (right). Note that we get a lot of good matches, even when the object is upside down!



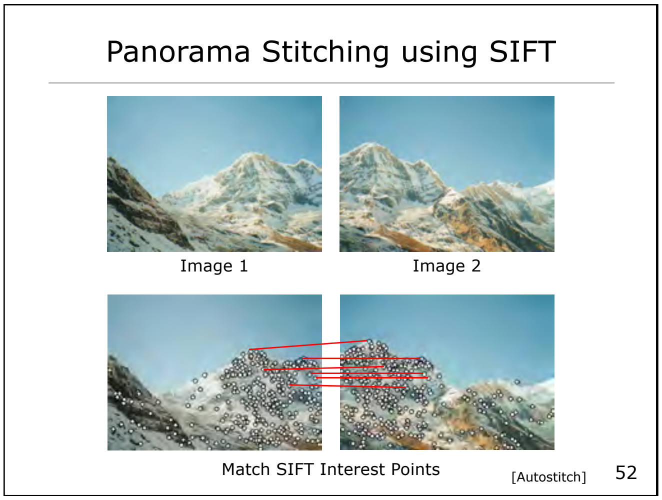
SIFT Results: Rotation Invariance

Here we see the benefit of using SIFT when there is clutter and occlusion in the scene. Occlusion is a difficult problem, especially in the context of object recognition. Shown here are three examples. In each case, the object we are looking for is shown in the bottom and the scene it lies in is shown on the top. Note that, in each case, several features on the object are successfully detected in the scene image. In the example on the right, the object is severely occluded in the scene but since many of its features are detected, we can not only be sure that the object is in the scene but also use the detected features to compute its position, scale and orientation in the scene image.



SIFT Results: Robustness to Clutter

Now let us take a look at some other applications of the SIFT detector. A popular one is image stitching where we are given a set of images of the same scene taken by rotating a camera and merging the images to obtain a single contiguous panorama. We are going to devote an entire lecture to this problem, but let us take a look at a simple case here. Shown here are two images of the same scene with overlapping fields of view. We first find SIFT features in both the images (white dots) and then use the descriptors to match features (red lines). Using these matching features, we can warp one image such that it aligns with the other one. The process of warping an image requires a geometric transformation that we will discuss in detail in the next lecture. For our purposes here, just assume that such a transformation is possible.

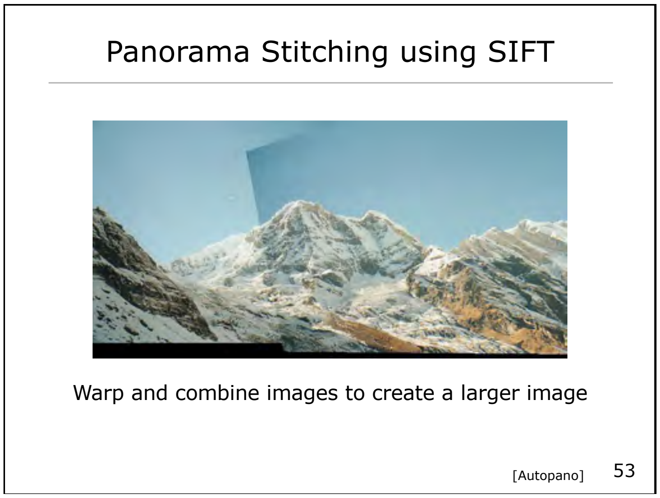


Panorama Stitching using SIFT

Match SIFT Interest Points

[Autostitch] 52

After warping one image such that it lies in the coordinate frame of the other, we overlay them to get the image shown here, which is a panorama with a field of view that is larger than that of the original images. In the next lecture, we will describe how we can remove the visible brightness and color differences between the two images to obtain a seamless panorama.

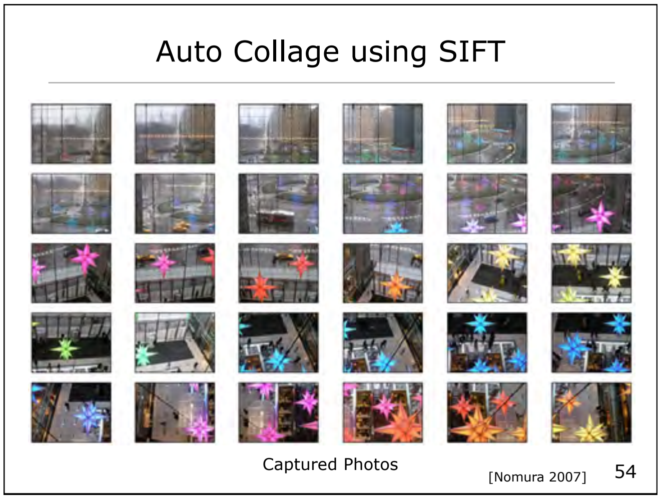


Panorama Stitching using SIFT

Warp and combine images to create a larger image

[Autopano] 53

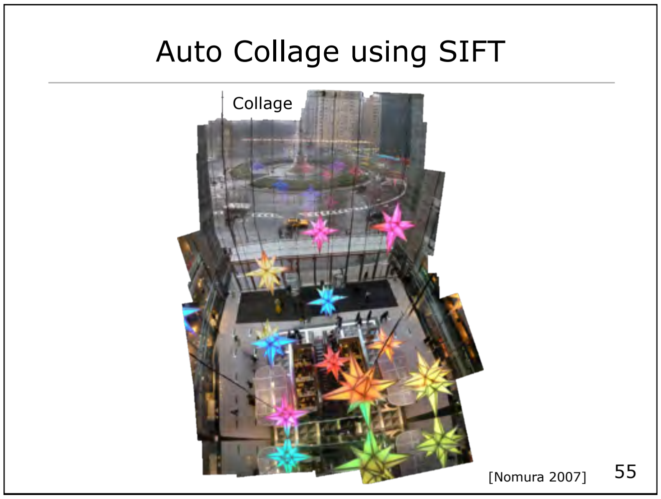
The same approach can be used to construct collages. Here we have a large collection of images. We simply apply SIFT to each one and then match features among the different images. In this case, we overlay the images on top of each other by simply translating, scaling and rotating them, but without geometrically warping them. Although, the end-result is not a single perspective view of the scene, it provides a representation that is useful for visualizing an unusually large field of view.



Auto Collage using SIFT

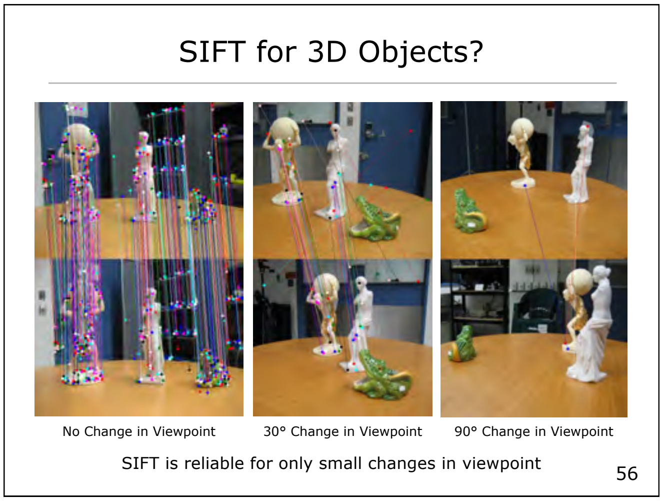
Captured Photos

[Nomura 2007] 54



Auto Collage using SIFT

[Nomura 2007] 55



SIFT for 3D Objects?

No Change in Viewpoint

30° Change in Viewpoint

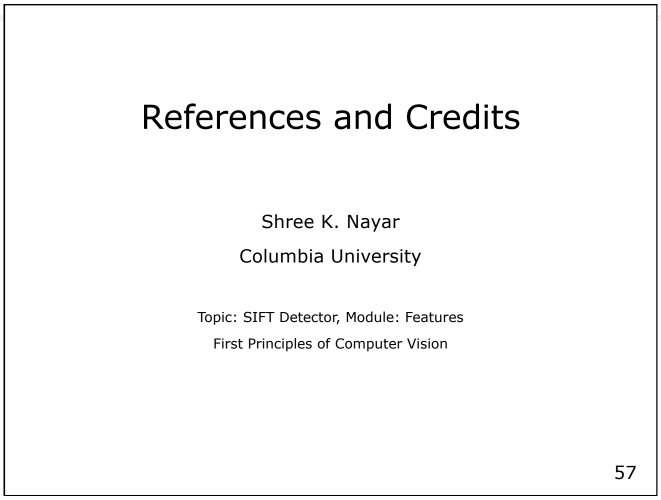
90° Change in Viewpoint

SIFT is reliable for only small changes in viewpoint

56

We have seen that the SIFT detector and descriptor can be used to solve a variety of visual matching and recognition problems. Unfortunately, it does not solve the general problem of recognition of three-dimensional objects. The kinds of objects we have examined thus far are more or less flat (planar) objects. When it comes to 3D objects, the problem is more complex. If we look at an object from different viewpoints, the local appearance, and hence the SIFT descriptor, of any given feature is going to vary with the viewpoint. So, if we applied SIFT to two images of a 3D object captured from different viewpoints, we can expect to get lots of features in each of the images, but less matches between the images as the difference in the viewpoints increases.

This is illustrated by the example shown here. On the left are two images of a scene taken from roughly the same viewpoint. As expected, we get lots of matches, with each one of the lines representing a successful match. In the center, we see two images taken with a difference in viewpoint of about 30 degrees along the horizontal direction. We see that there is dramatic drop in the number of matches between the images. If we further increase the difference in viewpoint to about 90 degrees (right), we hardly get any matches at all. Thus, in the context of recognition, SIFT is only reliable for small changes in viewpoint. That said, it still is an extremely useful detector that has found many real-world applications.



References and Credits

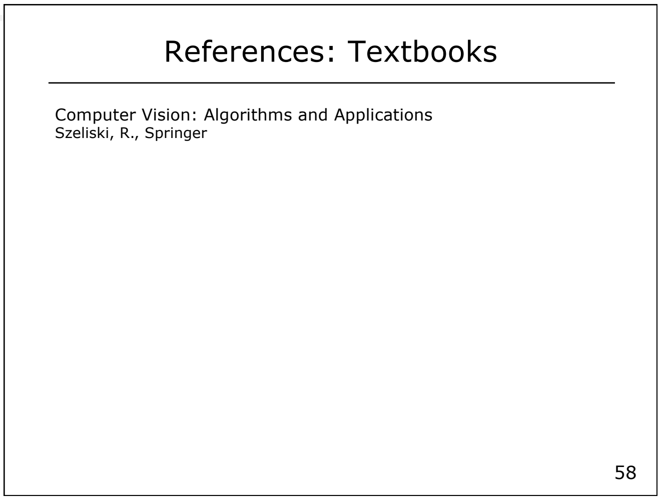
Shree K. Nayar

Columbia University

Topic: SIFT Detector, Module: Features

First Principles of Computer Vision

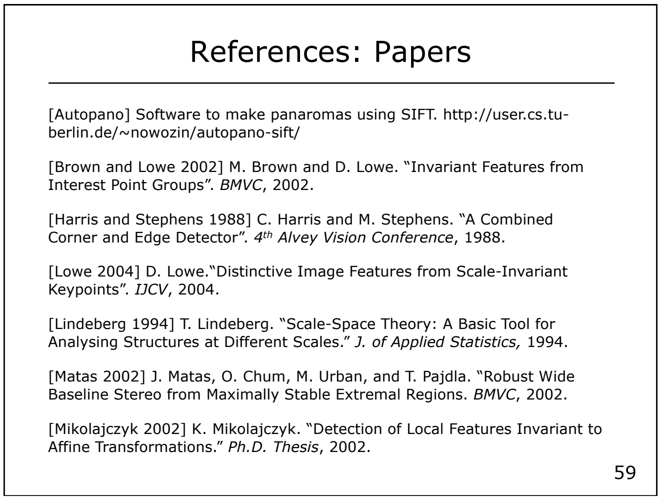
57



References: Textbooks

Computer Vision: Algorithms and Applications
Szeliski, R., Springer

58



References: Papers

[Autopano] Software to make panaromas using SIFT. http://user.cs.tu-berlin.de/~nowozin/autopano-sift/

[Brown and Lowe 2002] M. Brown and D. Lowe. "Invariant Features from Interest Point Groups". *BMVC*, 2002.

[Harris and Stephens 1988] C. Harris and M. Stephens. "A Combined Corner and Edge Detector". *4th Alvey Vision Conference*, 1988.

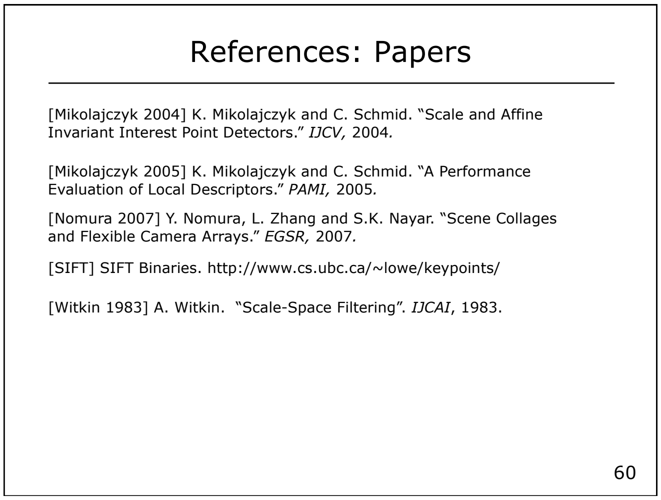
[Lowe 2004] D. Lowe."Distinctive Image Features from Scale-Invariant Keypoints". *IJCV*, 2004.

[Lindeberg 1994] T. Lindeberg. "Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales." *J. of Applied Statistics,* 1994.

[Matas 2002] J. Matas, O. Chum, M. Urban, and T. Pajdla. "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. *BMVC*, 2002.

[Mikolajczyk 2002] K. Mikolajczyk. "Detection of Local Features Invariant to Affine Transformations." *Ph.D. Thesis*, 2002.

59



References: Papers

[Mikolajczyk 2004] K. Mikolajczyk and C. Schmid. "Scale and Affine Invariant Interest Point Detectors." *IJCV, 2004.*

[Mikolajczyk 2005] K. Mikolajczyk and C. Schmid. "A Performance Evaluation of Local Descriptors." *PAMI,* 2005.

[Nomura 2007] Y. Nomura, L. Zhang and S.K. Nayar. "Scene Collages and Flexible Camera Arrays." *EGSR,* 2007.

[SIFT] SIFT Binaries. http://www.cs.ubc.ca/~lowe/keypoints/

[Witkin 1983] A. Witkin. "Scale-Space Filtering". *IJCAI*, 1983.

60

Acknowledgements: Thanks to Nisha Aggarwal and Jenna Everard for their help with transcription, editing and proofreading.

References

[Szeliski 2022] Computer Vision: Algorithms and Applications, Szeliski, R., Springer, 2022.

[Autopano] Software to make panaromas using SIFT. http://user.cs.tu-berlin.de/~nowozin/autopano-sift/

[Brown and Lowe 2002] M. Brown and D. Lowe. “Invariant Features from Interest Point Groups”. BMVC, 2002.

[Harris and Stephens 1988] C. Harris and M. Stephens. “A Combined Corner and Edge Detector”. 4th Alvey Vision Conference, 1988.

[Lowe 2004] D. Lowe.“Distinctive Image Features from Scale-Invariant Keypoints”. IJCV, 2004.

[Lindeberg 1994] T. Lindeberg. “Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales.” J. of Applied Statistics, 1994.

[Matas 2002] J. Matas, O. Chum, M. Urban, and T. Pajdla. “Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. BMVC, 2002.

[Mikolajczyk 2002] K. Mikolajczyk. “Detection of Local Features Invariant to Affine Transformations.” Ph.D. Thesis, 2002.

[Mikolajczyk 2004] K. Mikolajczyk and C. Schmid. “Scale and Affine Invariant Interest Point Detectors.” IJCV, 2004.

[Mikolajczyk 2005] K. Mikolajczyk and C. Schmid. “A Performance Evaluation of Local Descriptors.” PAMI, 2005.

[Nomura 2007] Y. Nomura, L. Zhang and S.K. Nayar. “Scene Collages and Flexible Camera Arrays.” EGSR, 2007.

[SIFT] SIFT Binaries. http://www.cs.ubc.ca/~lowe/keypoints/

[Witkin 1983] A. Witkin. “Scale-Space Filtering”. IJCAI, 1983.

[Nayar 2022E] Image Processing I, Nayar, S. K., Monograph FPCV-1-4, First Principles of Computer Vision, Columbia University, New York, March 2022.

[Nayar 2022F] Image Processing II, Nayar, S. K., Monograph FPCV-1-5, First Principles of Computer Vision, Columbia University, New York, March 2022.

[Nayar 2022G] Edge Detection, Nayar, S. K., Monograph FPCV-2-1, First Principles of Computer Vision, Columbia University, New York, May 2022.

[Nayar 2022H] Boundary Detection, Nayar, S. K., Monograph FPCV-2-2, First Principles of Computer Vision, Columbia University, New York, June 2022.

[Nayar 2025A] Image Stitching, Nayar, S. K., Monograph FPCV-2-4, First Principles of Computer Vision, Columbia University, New York, January 2025.

[Nayar 2025H] Camera Calibration, Nayar, S. K., Monograph FPCV-4-1, First Principles of Computer Vision, Columbia University, New York, April 2025.

[Nayar 2025I] Uncalibrated Stereo, Nayar, S. K., Monograph FPCV-4-2, First Principles of Computer Vision, Columbia University, New York, April 2025.

[Nayar 2025J] Optical Flow, Nayar, S. K., Monograph FPCV-4-3, First Principles of Computer Vision, Columbia University, New York, April 2025.

[Nayar 2025K] Structure from Motion, Nayar, S. K., Monograph FPCV-4-4, First Principles of Computer Vision, Columbia University, New York, April 2025.

[Nayar 2025L] Object Tracking, Nayar, S. K., Monograph FPCV-5-1, First Principles of Computer Vision, Columbia University, New York, May 2025.