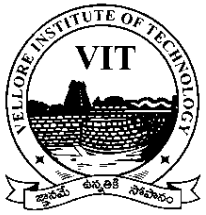


VIT-AP UNIVERSITY

Computer Vision

(Course Code: 4047)

Module-2:Lecture-4: SIFT

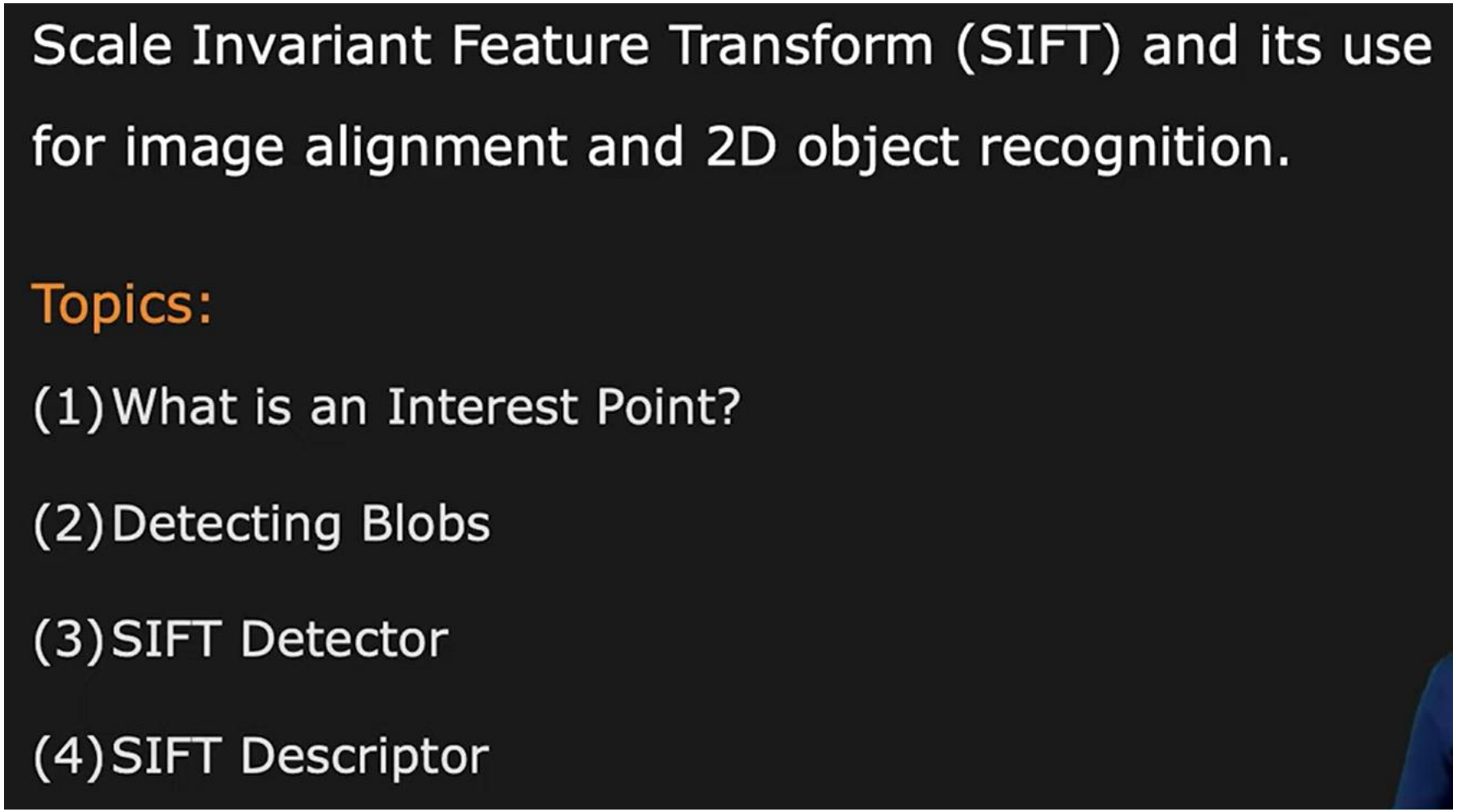
Gundimeda Venugopal, Professor of Practice, SCOPE

How would you recognize the following types of objects?

Need to account for:
Scale
Rotation
Occlusion

To address this, Extract descriptive unique features in Template and the Target Image and do the matching

Scale Invariant Feature Transform(SIFT) Detector

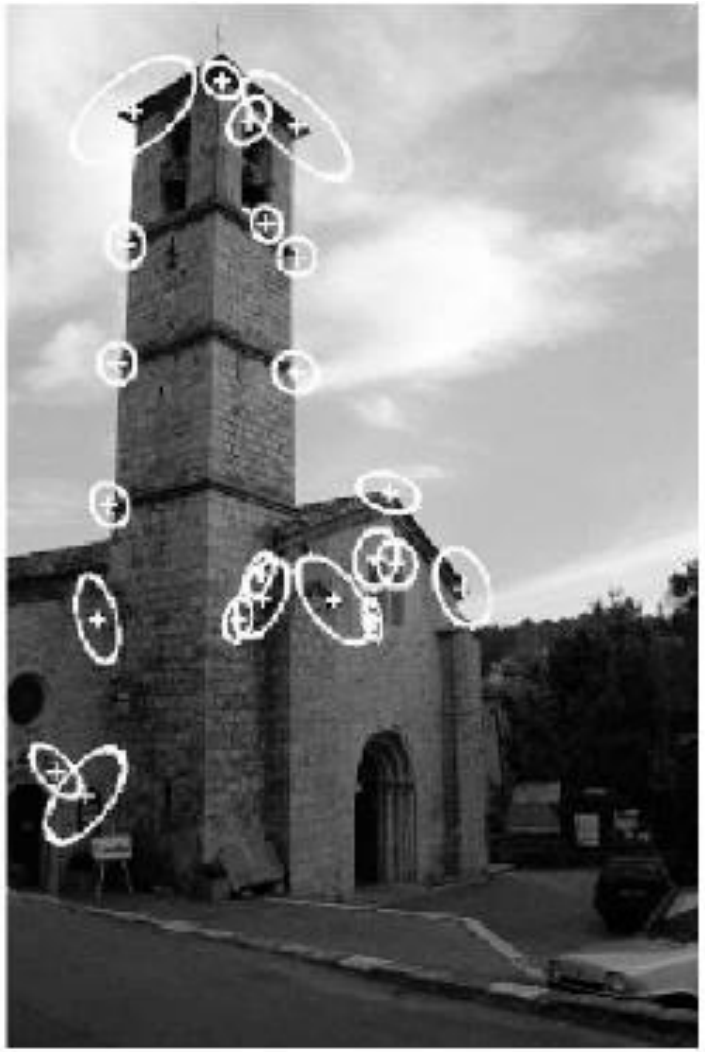


Interest Points

- A point in an image which has a well-defined position and can be robustly detected and is relevant for higher level processing (a general term in computer vision).
- Interest points are commonly used by image stabilization and structure from motion applications to track how the image changes from frame to frame
- Typically associated with a significant change of one or more image properties simultaneously (e.g., intensity, color, texture).

Motivation: Matching Problem

❖Vision tasks such as stereo, panorama stitching and motion estimation require finding corresponding features across two or more views.

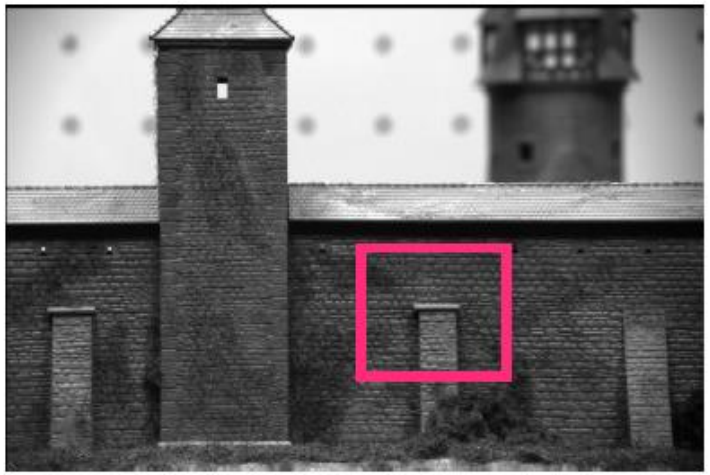


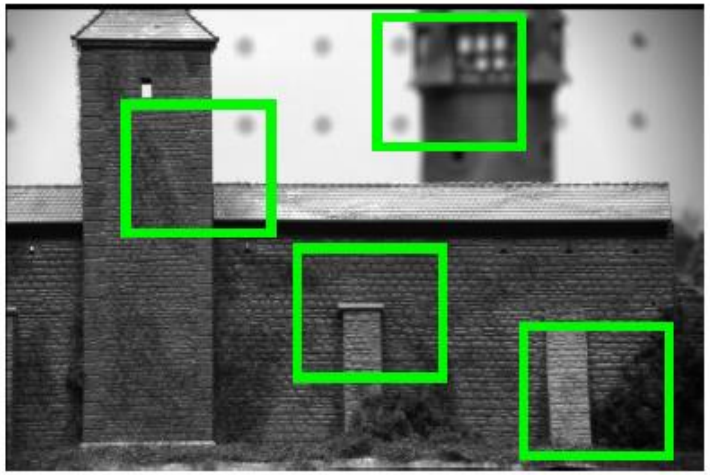
Look at cross marks in those two images , the position of these points(cross marks) haven't changed even the position of camera changed.

Take a small window around each of these points in both the images.

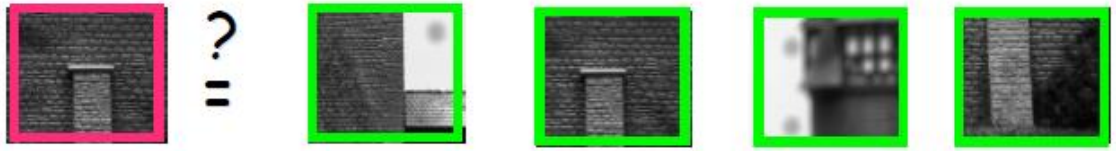
Consider you have a mechanism to map the window in one image to another window in the second image according to their features(feature extraction), then we completed our task of recognizing the objects.

Not all patches are Created Equal





Inituition: this would be a good patch for matching, since it is very distinctive (there is only one patch in the second frame that looks similar).

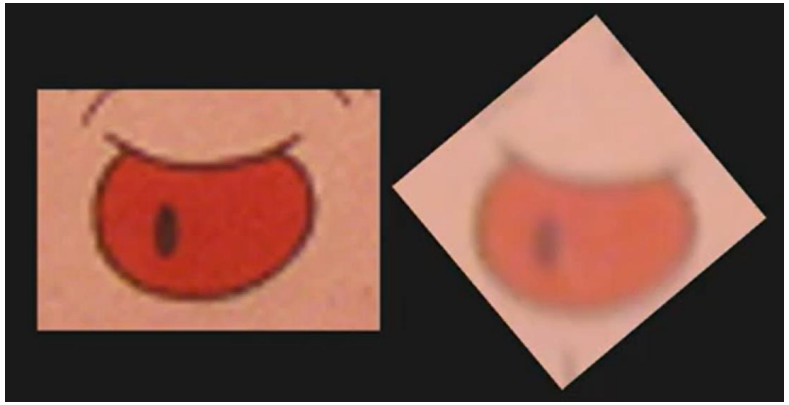


Raw Images are hard to match

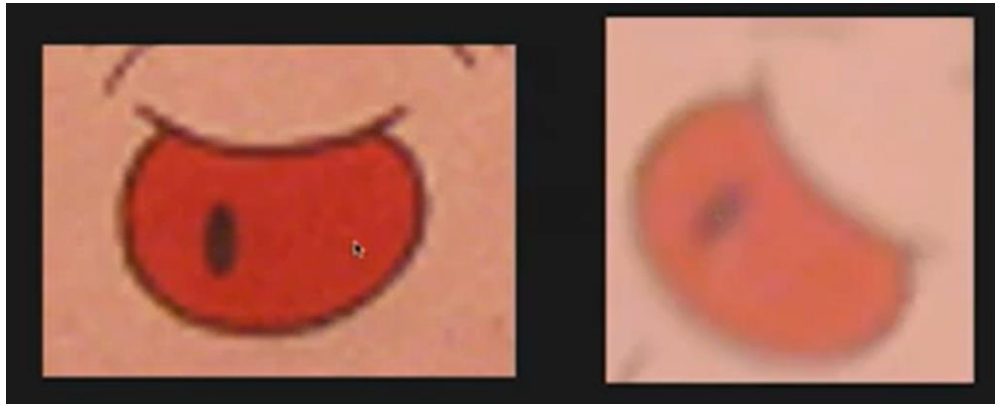
Removing Sources of variations

Scale differences

Reorient one to match the orientation of other

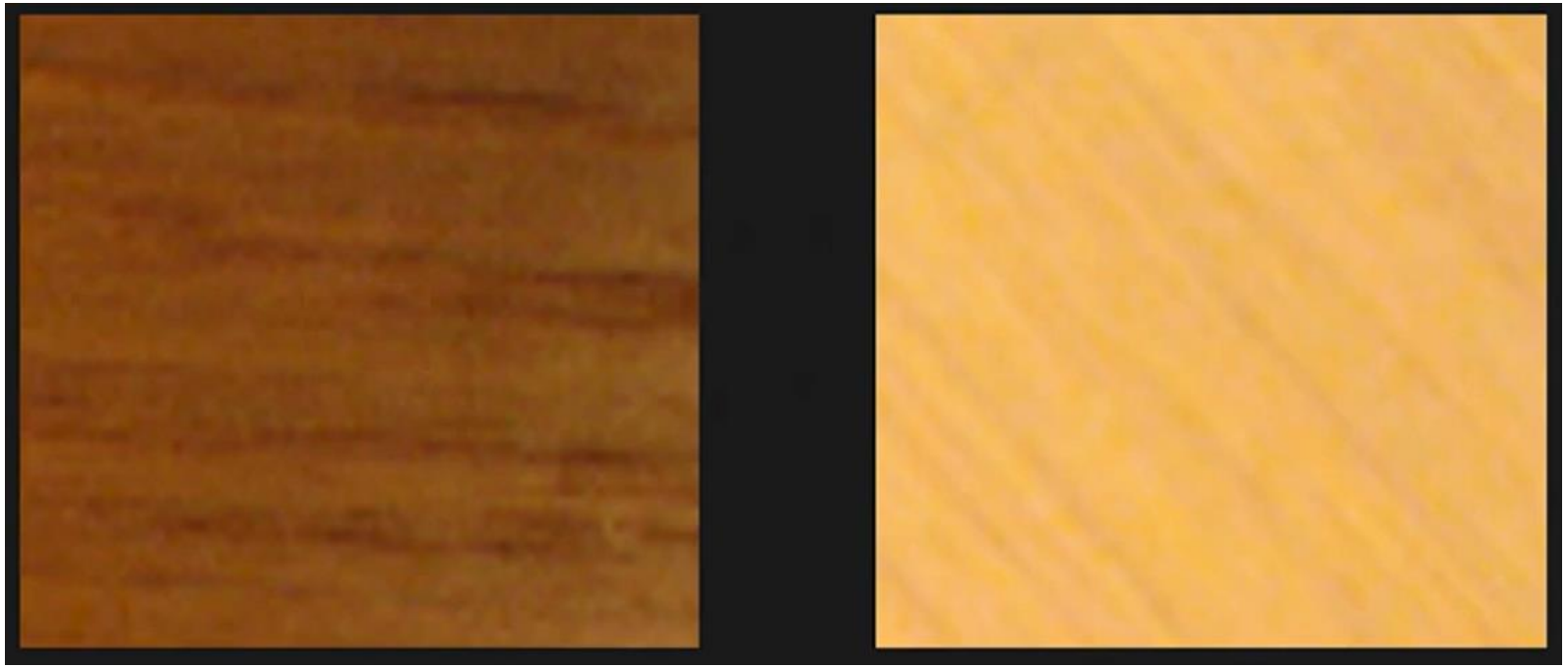


Insensitive to lighting changes



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

Some patches are not interesting

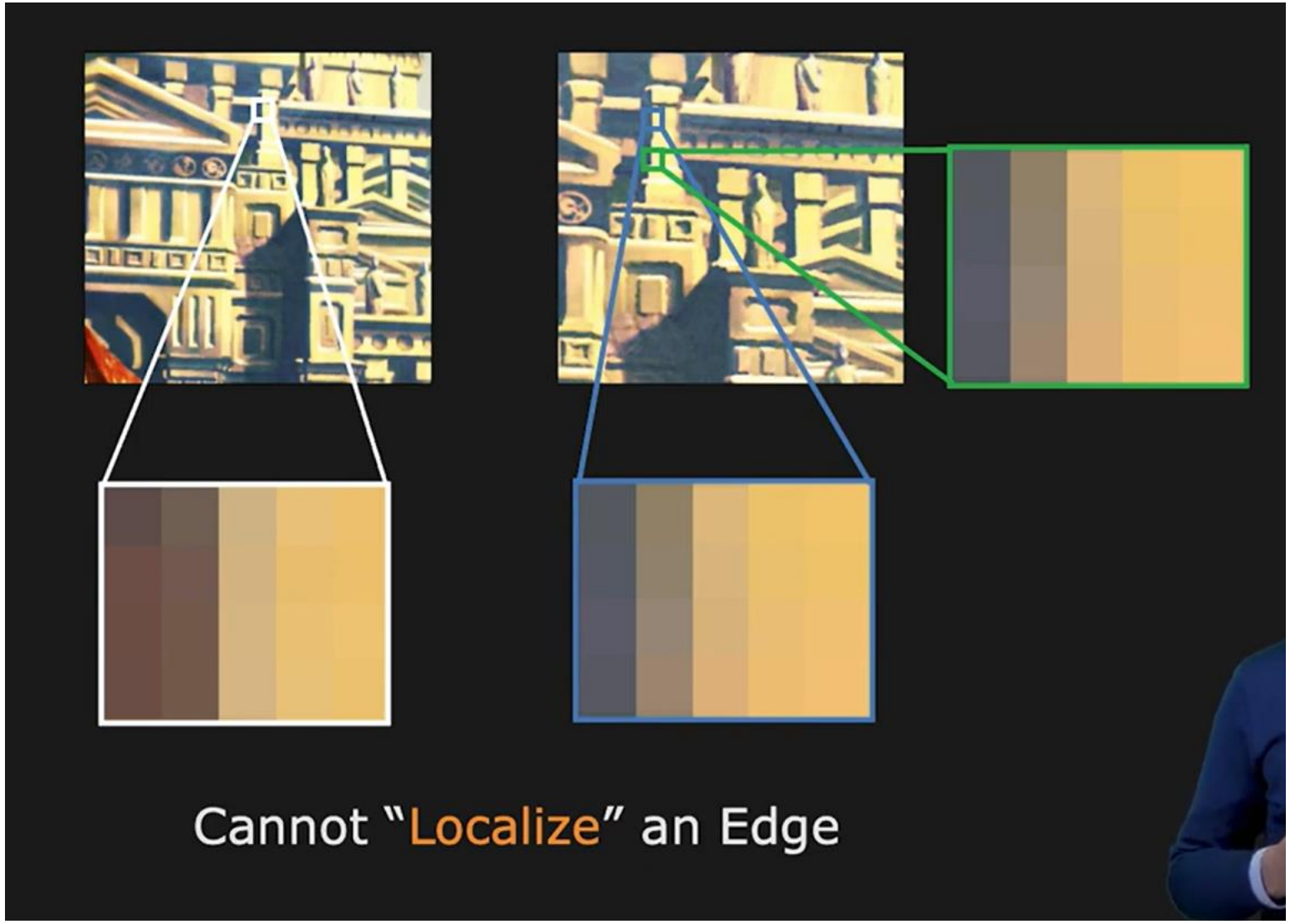


Not unique enough w.r.t brightness variation, color variation etc

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

Are Lines/Edges Interesting?

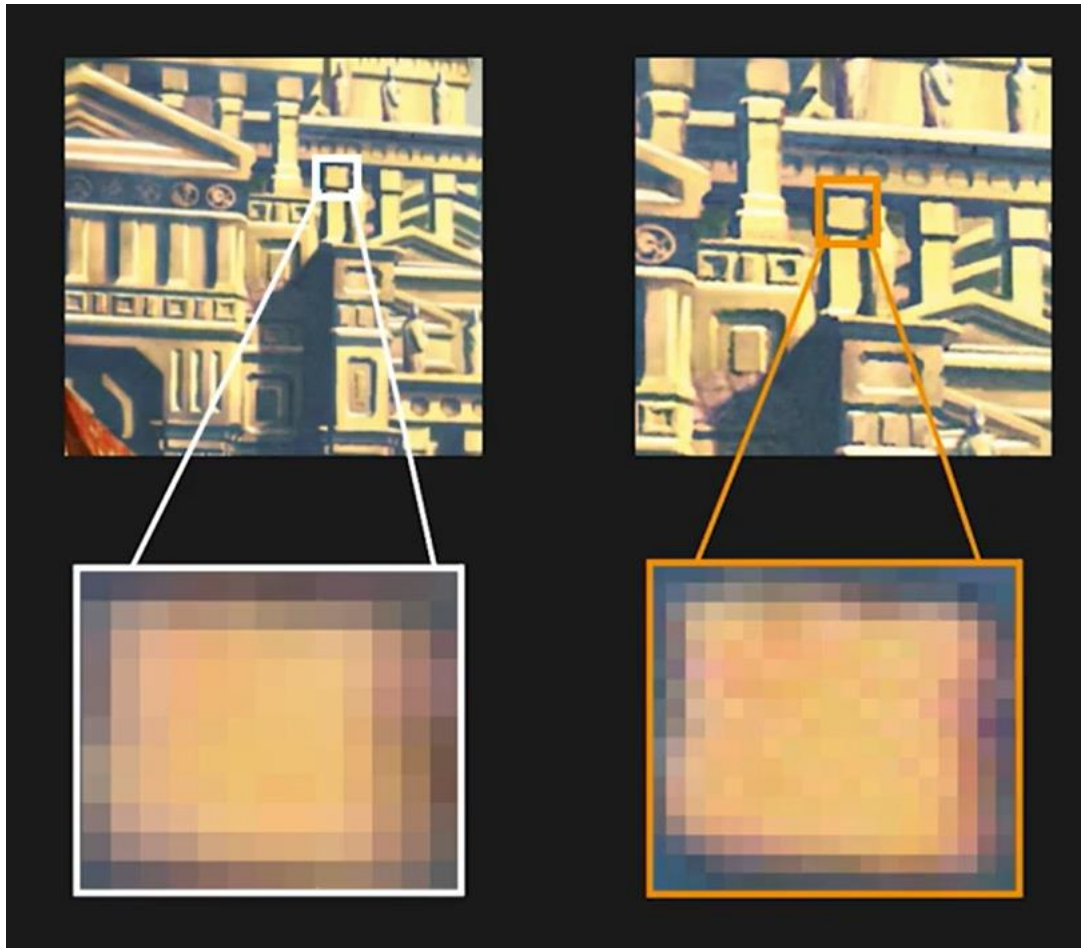


Same kind of brightness variations

Not descriptive enough or unique enough

Corners are interesting points and they are useful for simple applications and not for recognizing complex objects.

Are Blobs interesting?



After you normalize for scale
(put in windows of similar size):

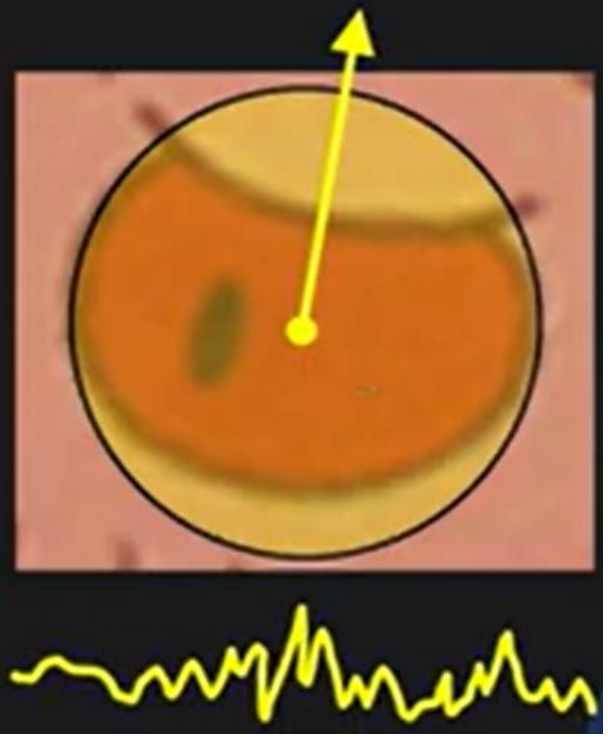
Local Appearance: Brightness variation
Position of blob well-defined

Potentially good interest points

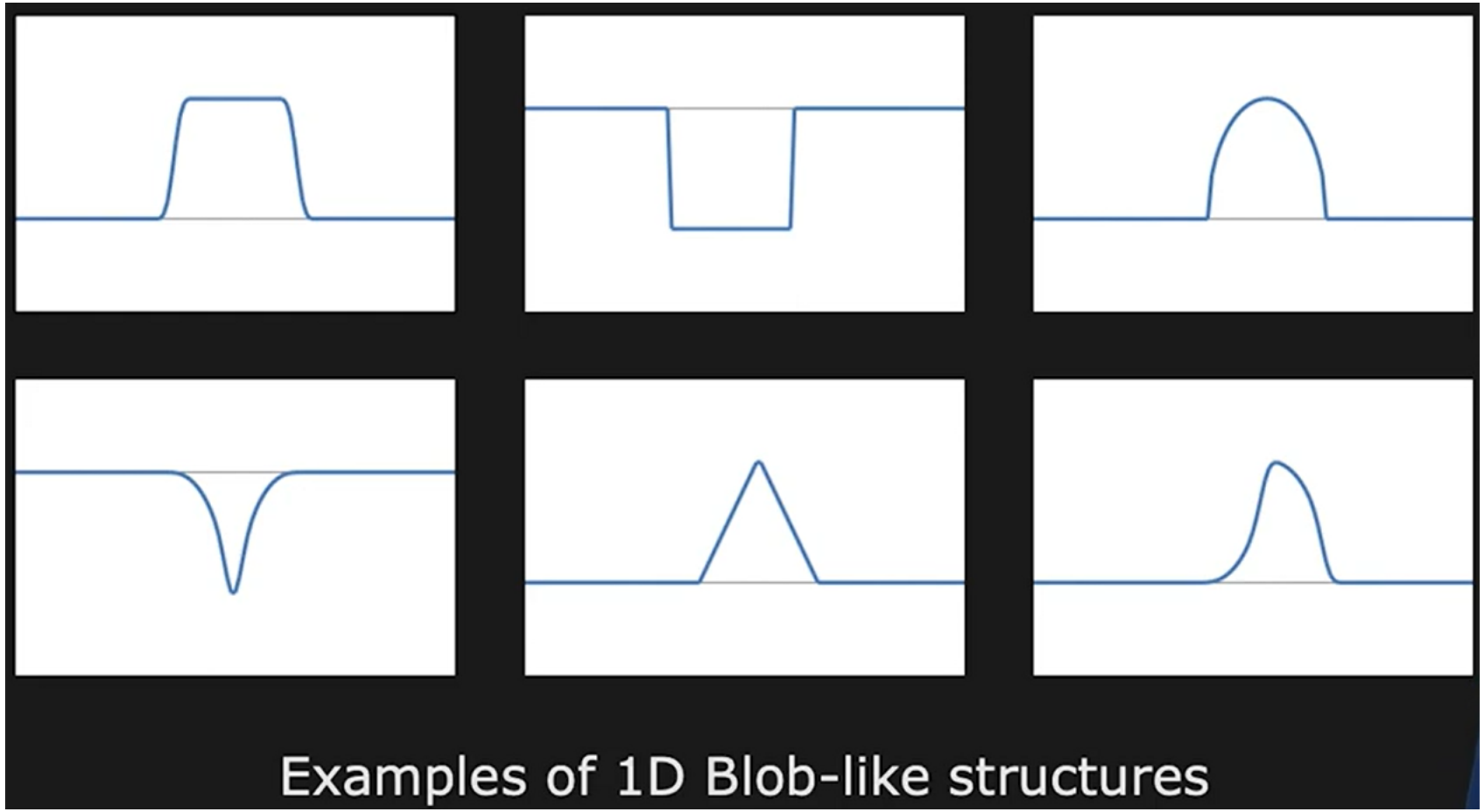
Blobs as Interest Points

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

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

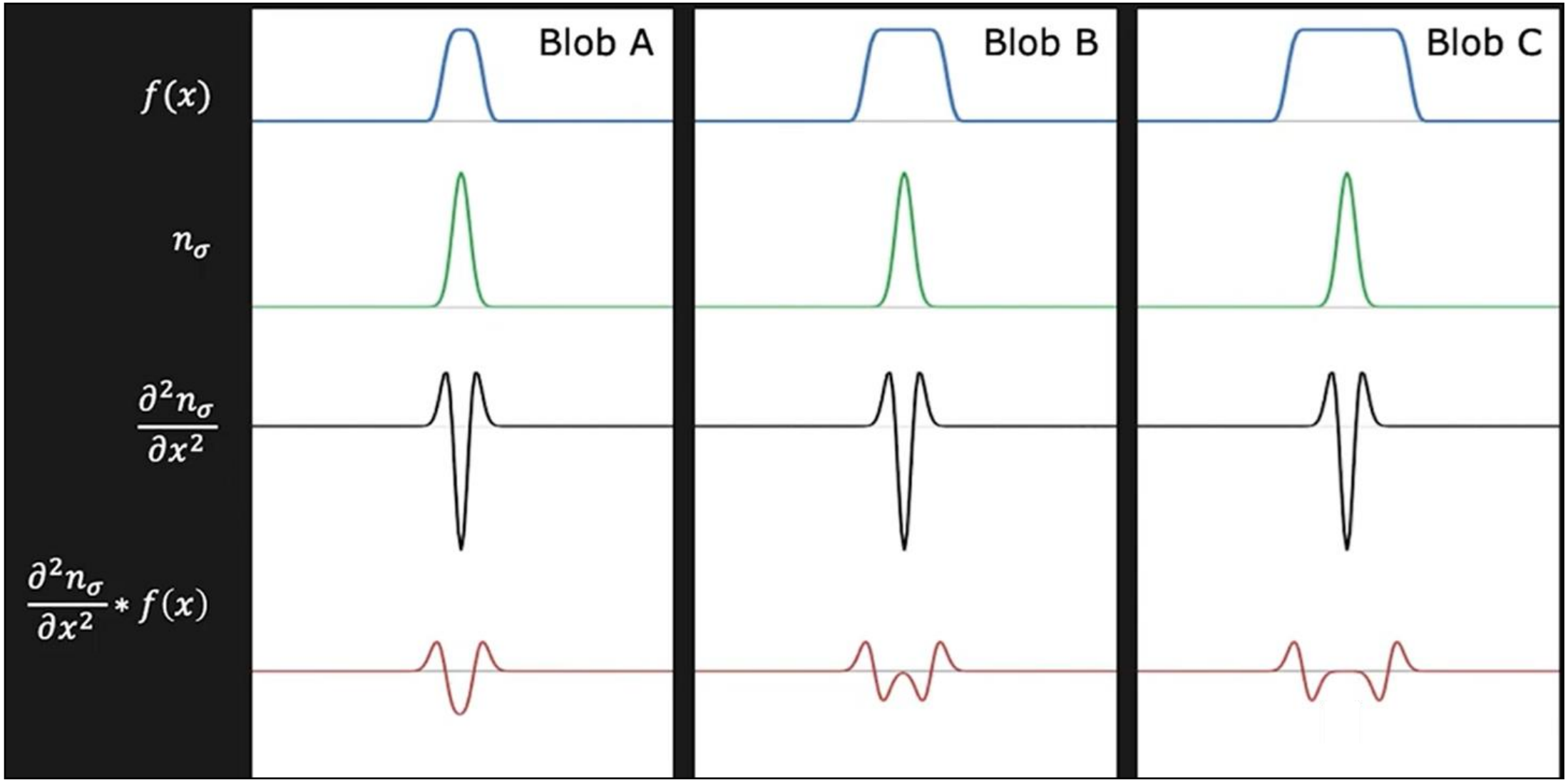


1D Blobs

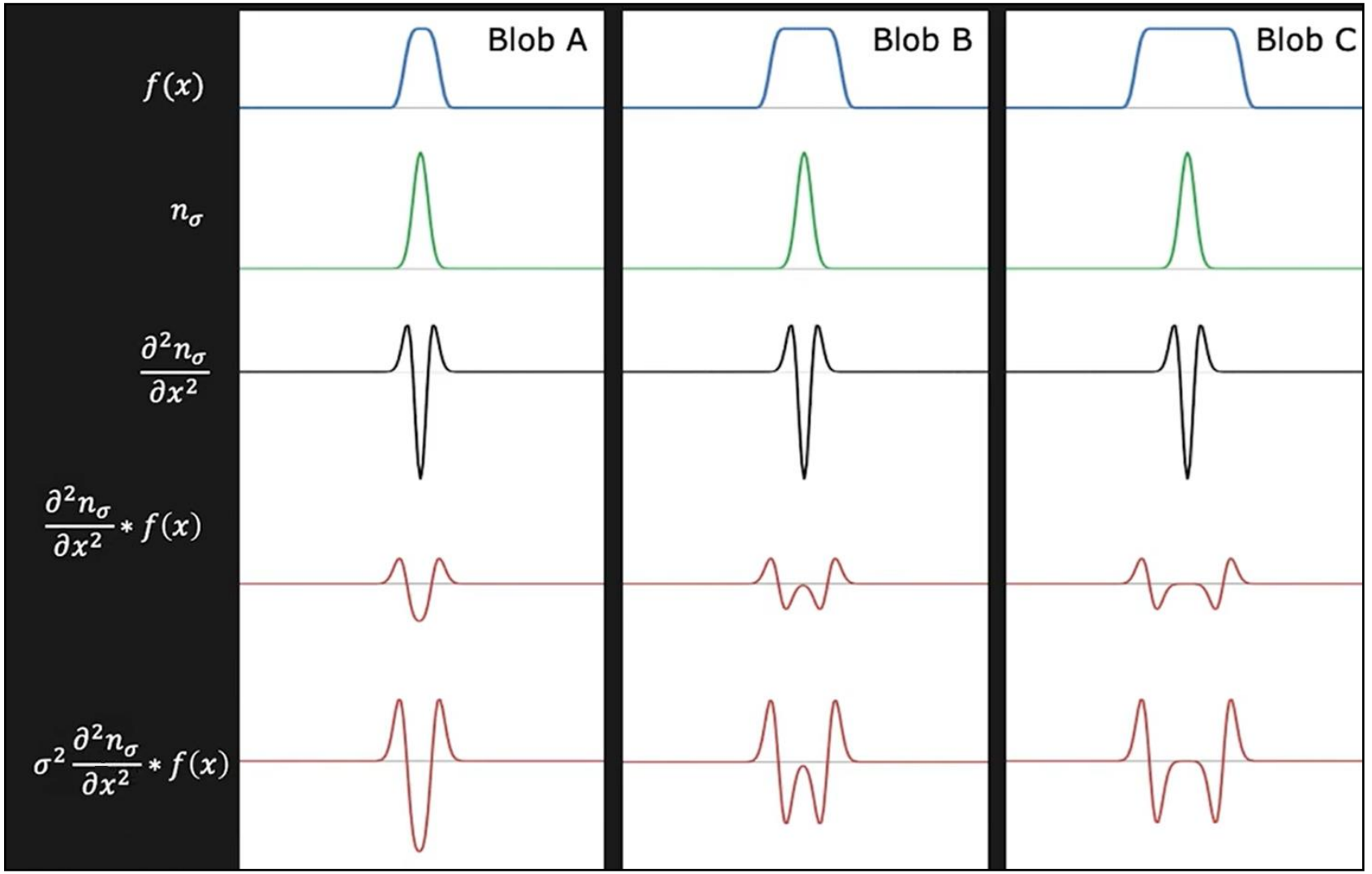


1D Blob 2nd Derivative of Gaussian

Need to detect Blobs of different widths:

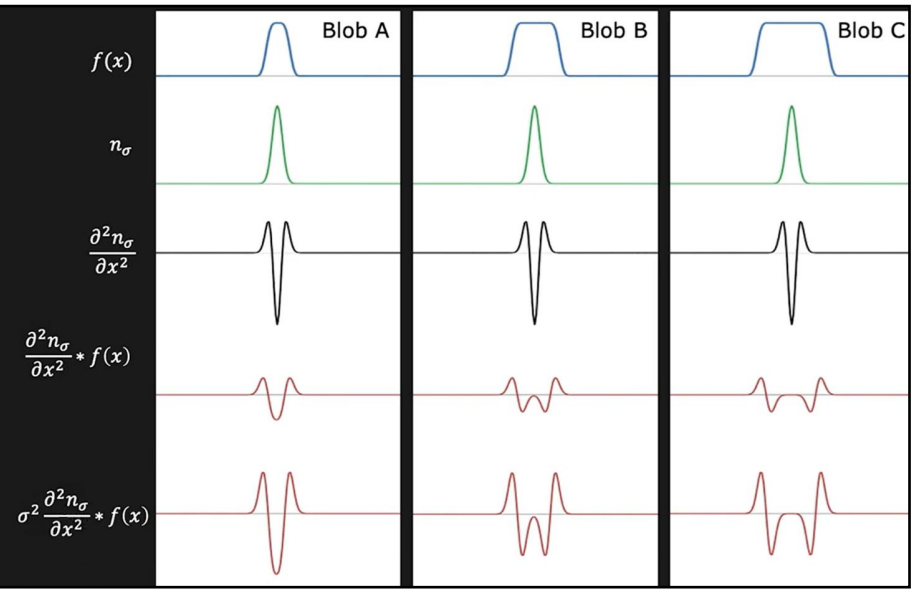


1D Blob σ-normalized 2nd Derivative of Gaussian

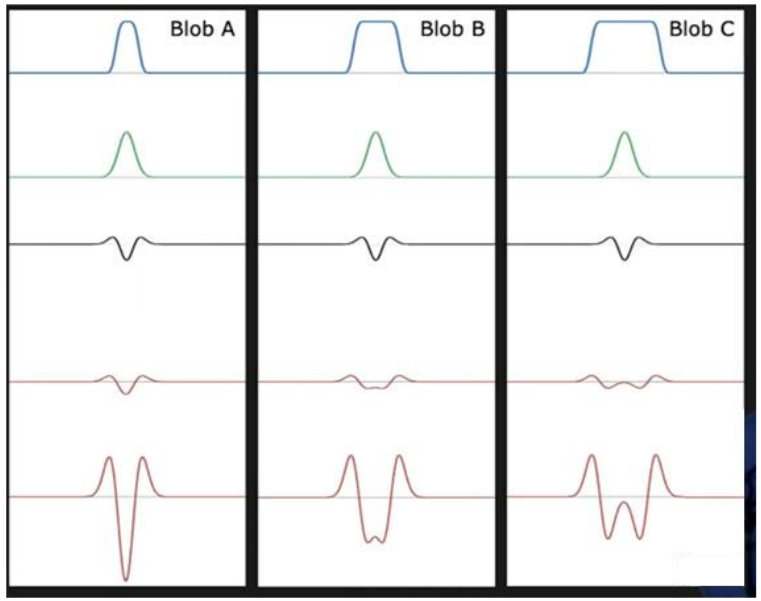


Characteristic Scale change (σ) to detect blobs of different size

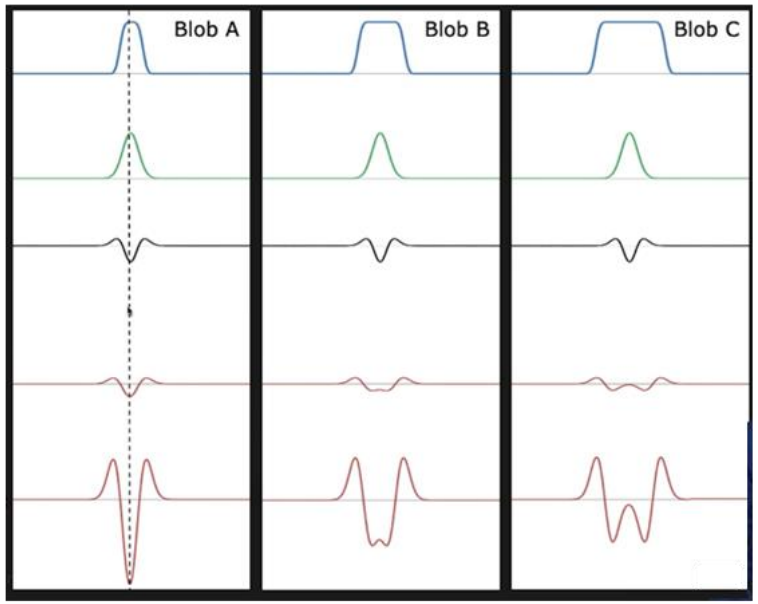
σ-normalized 2nd Gaussian for better σ changes

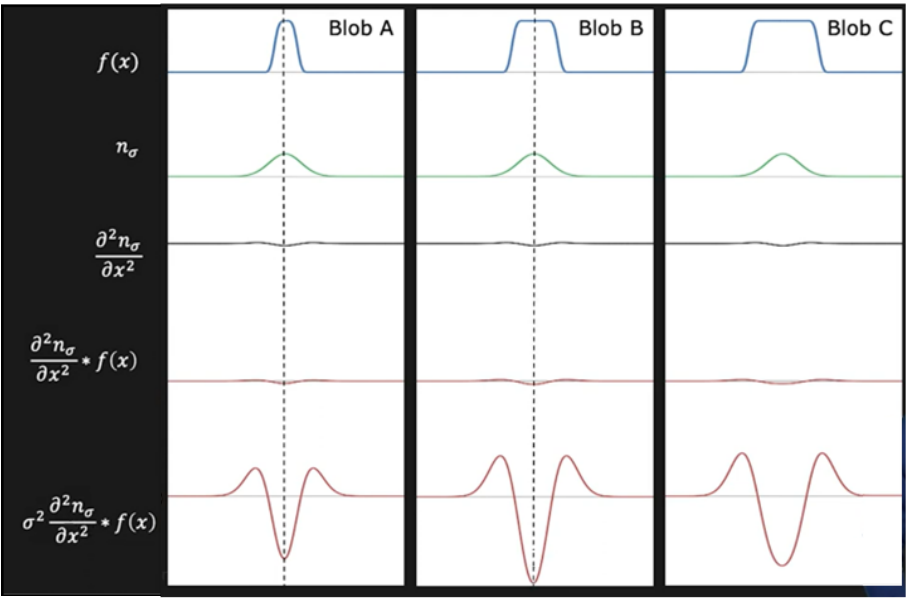


Peak Response for Blob A at σ= σ_A

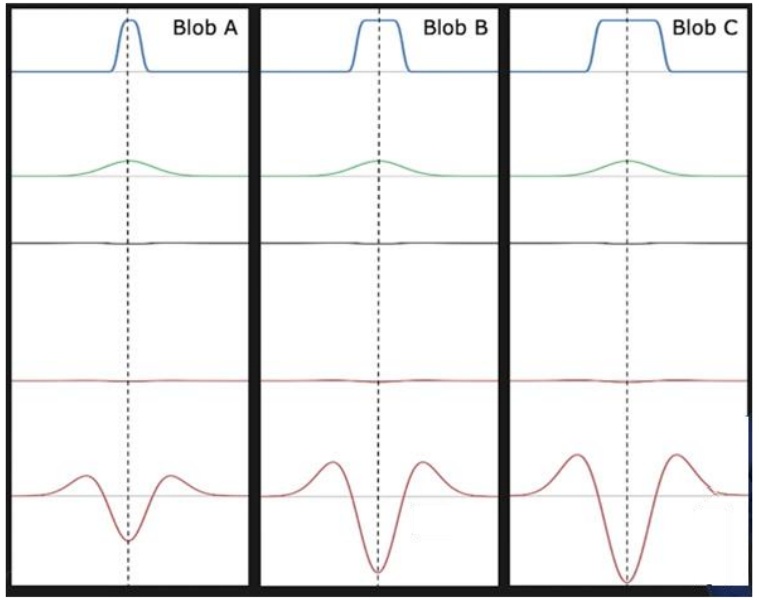


Blob A detection at σ= σ_A

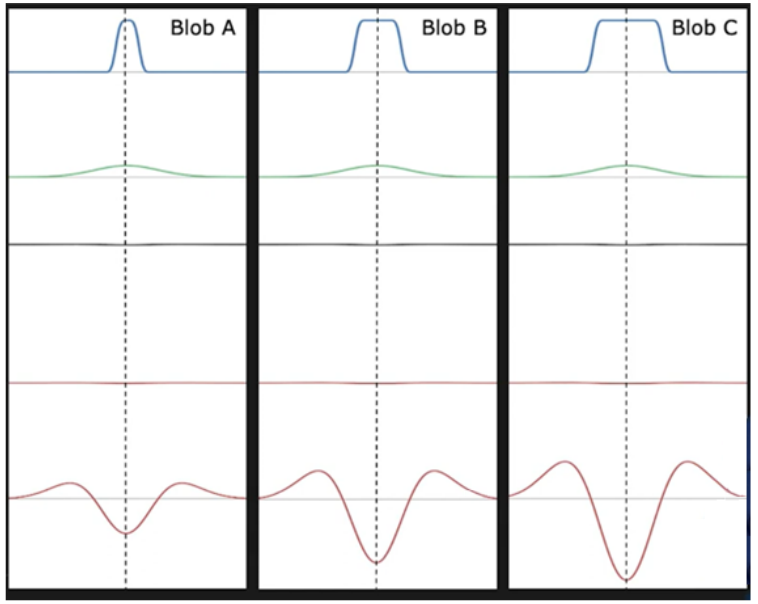




Blob B detection at σ= σ_B

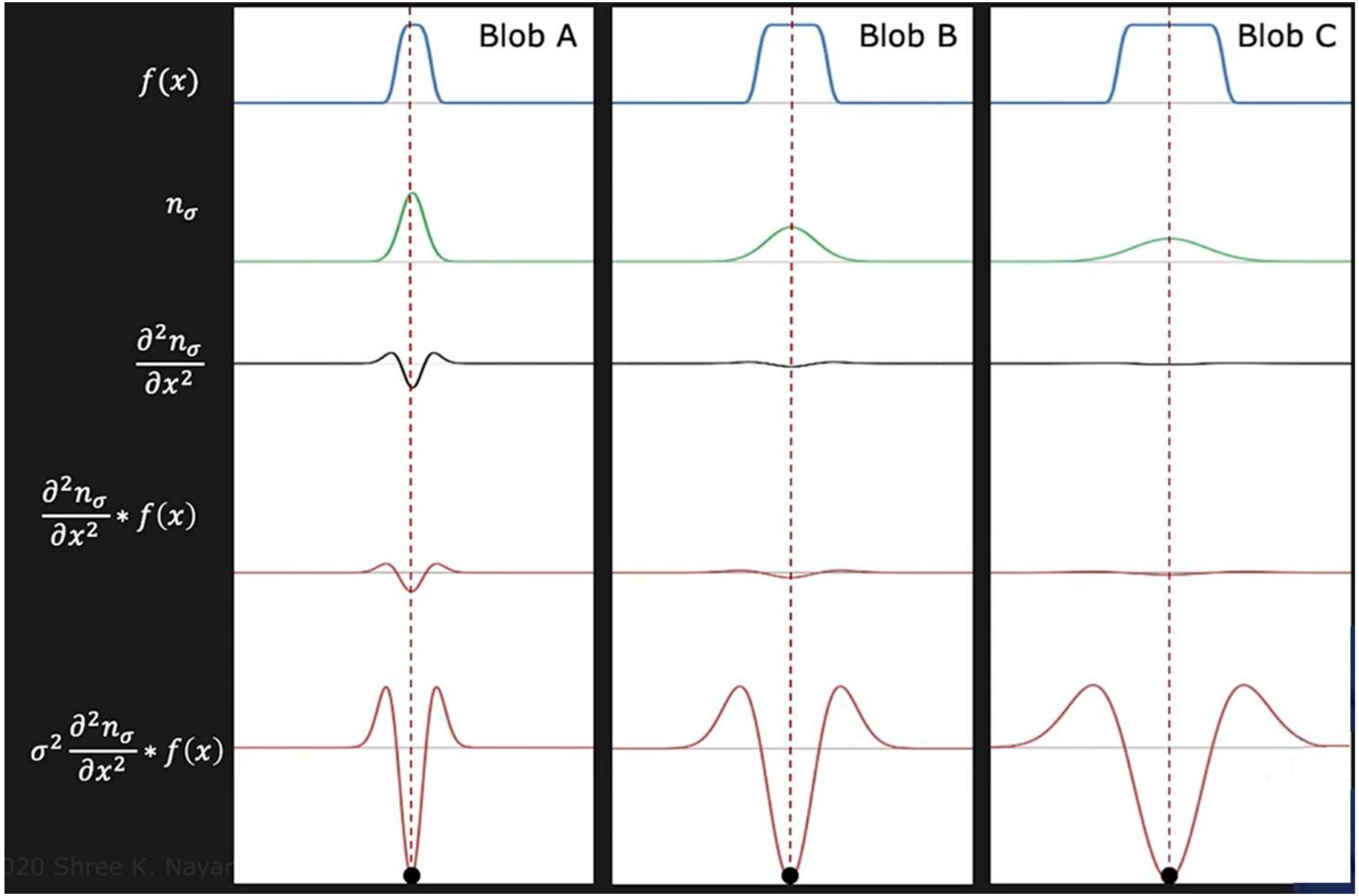


Blob C detection at σ= σ_C

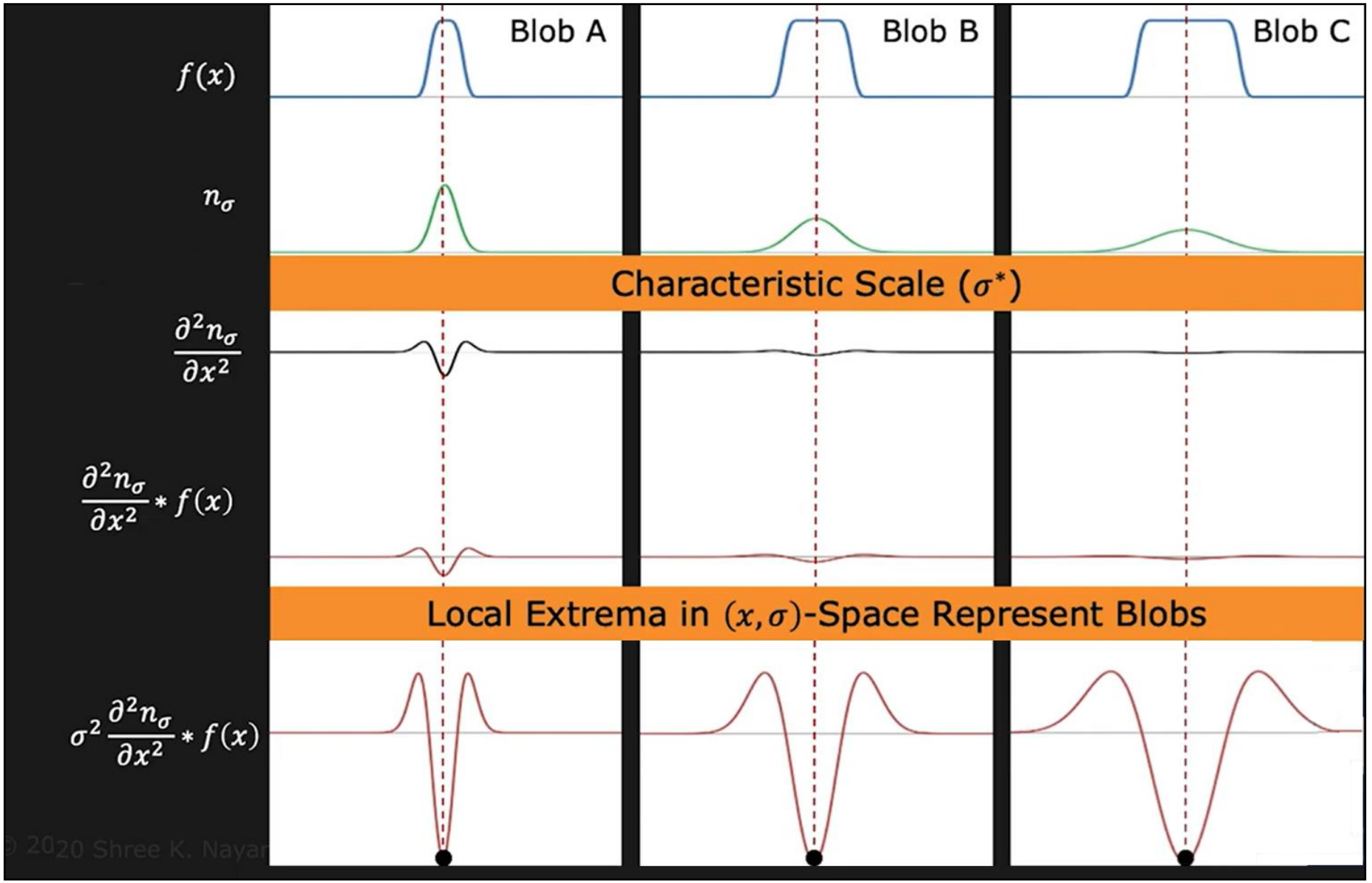


As σ increases more, Response decreases for A, B and C

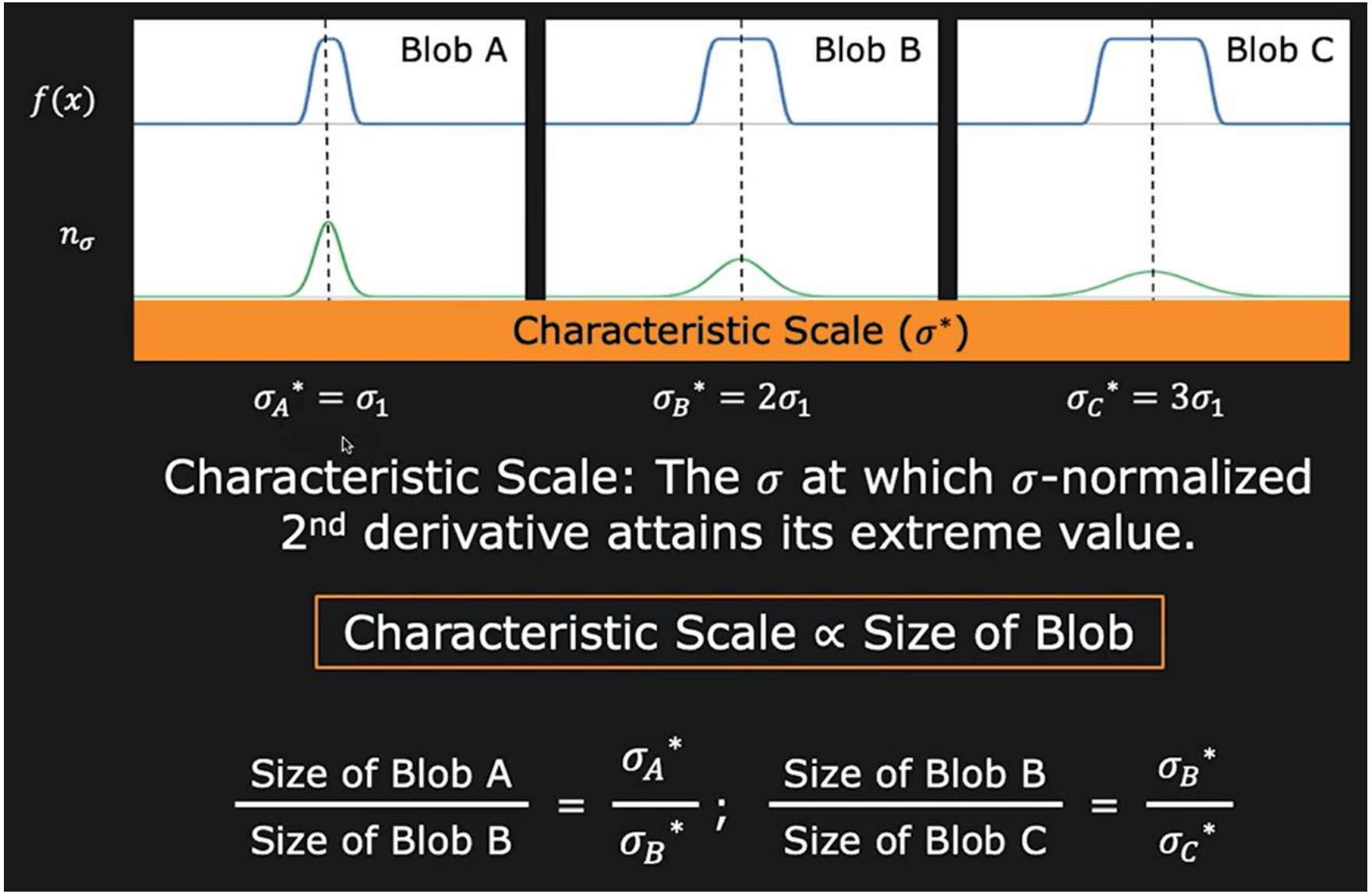
Detecting 1D Blobs of different sizes with respective σ-values for Gaussian



1D Blob σ-normalized 2nd Derivative of Gaussian



Characteristic Scale and Size



1D Blob Detection Summary

Detect a stack of 1D Blobs of different scales (corresponding to different σ values) at different positions

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 x, σ pairs where you have you have the local extremum in this space

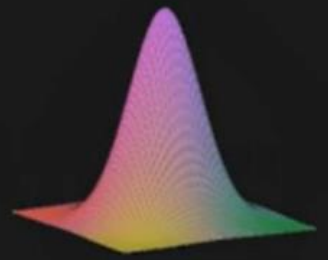
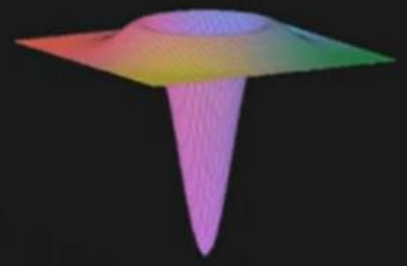
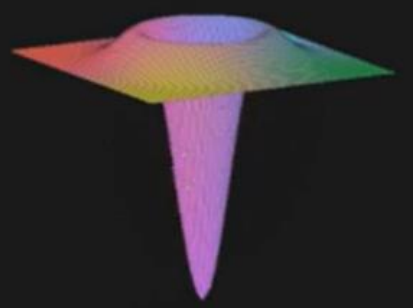
Find: $(x^*, \sigma^*) = \arg\max_{(x,\sigma)} \left| \sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x) \right|$

x^*: Blob Position

σ^*: Characteristic Scale (Blob Size)

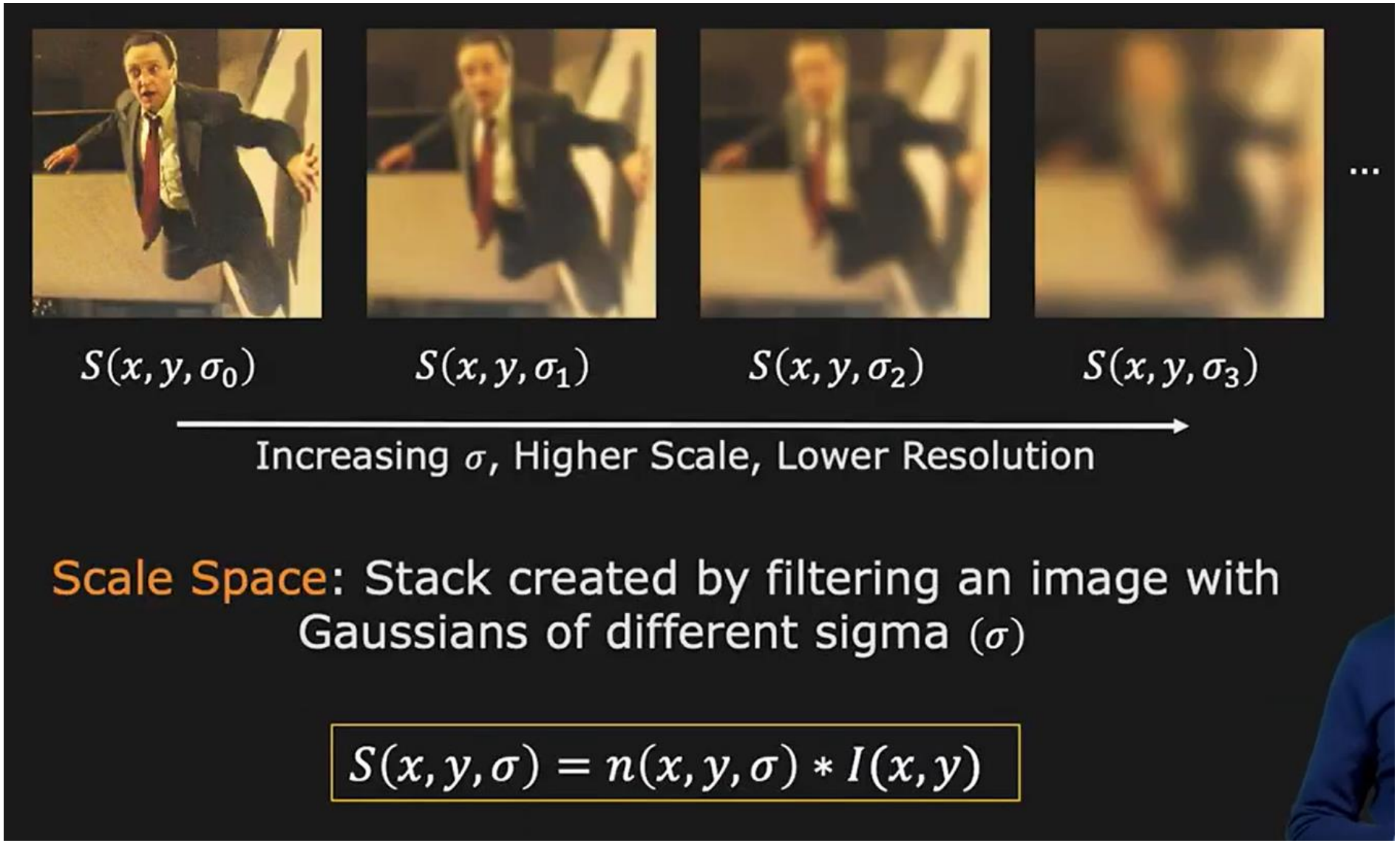
2D Blob Detector

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

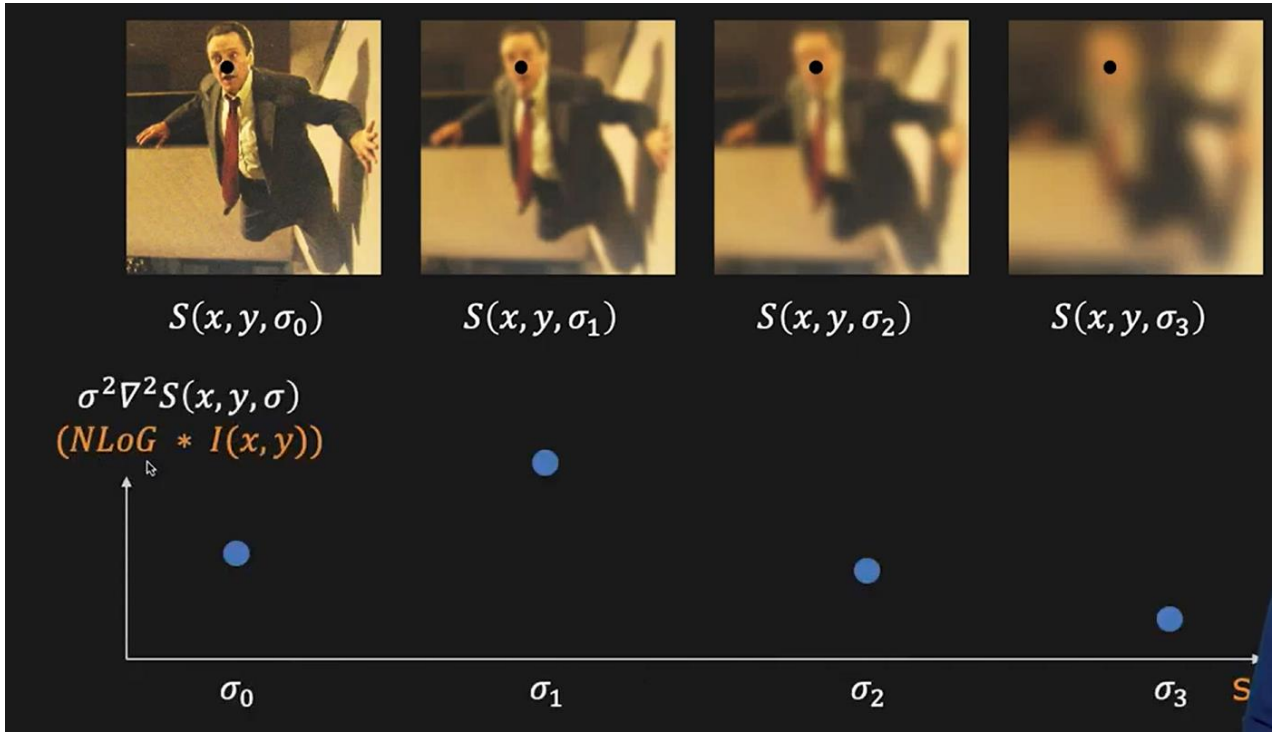
Laplacian	Gaussian	LoG	NLoG
$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$			
	n_σ	$\nabla^2 n_\sigma$	$\sigma^2 \nabla^2 n_\sigma$

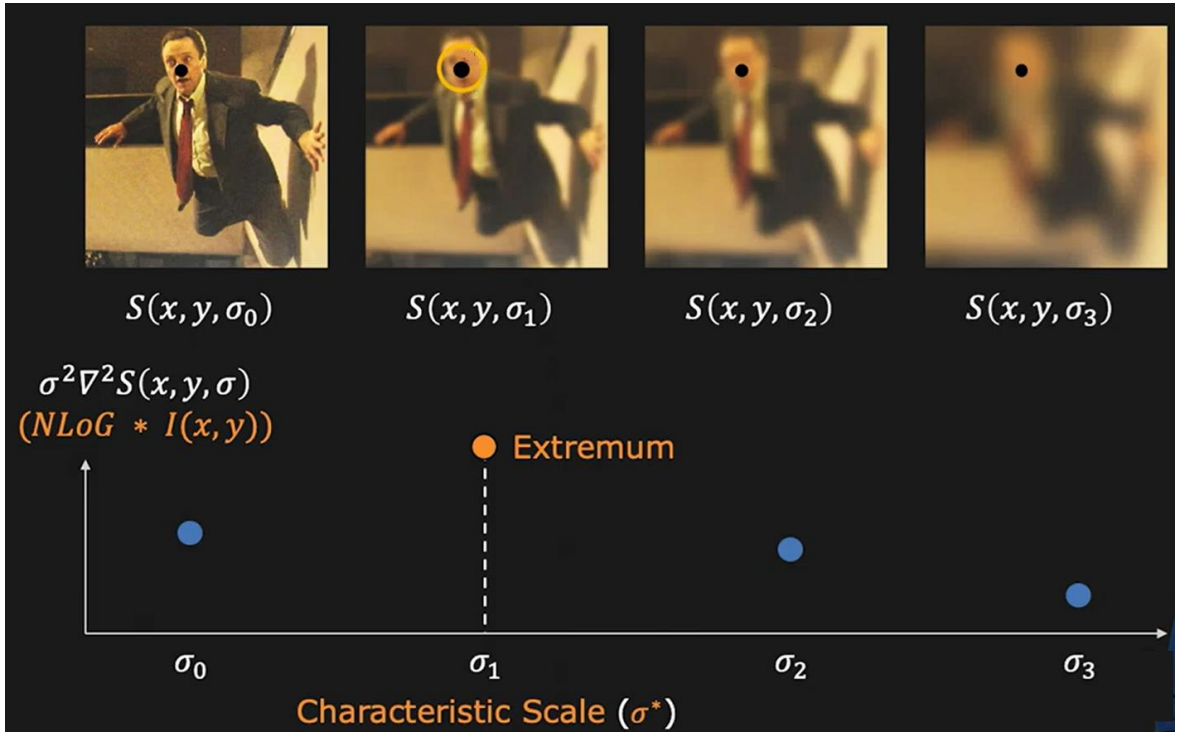
Location of Blobs given by Local Extrema after applying Normalized Laplacian of Gaussian at many scales.

Scale Space

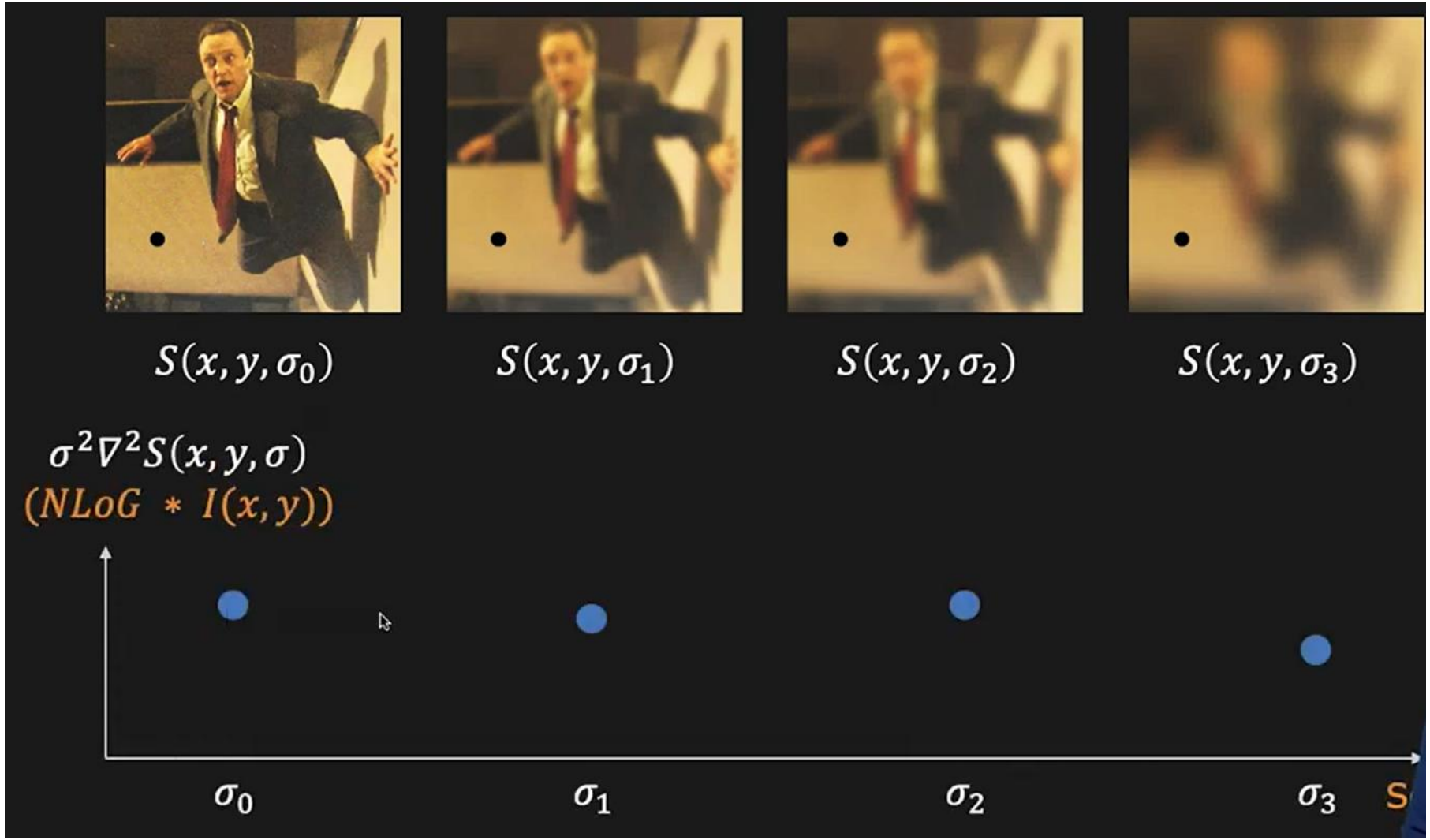

...
$S(x, y, \sigma_0)$
$S(x, y, \sigma_1)$
$S(x, y, \sigma_2)$
$S(x, y, \sigma_3)$
Increasing σ, Higher Scale, Lower Resolution
Scale Space: Stack created by filtering an image with Gaussians of different sigma (σ)
$S(x, y, \sigma) = n(x, y, \sigma) * I(x, y)$

Blob Detection Example using local Extrema





Blob Detection using local Extrema (e.g., Flat region)



There is no extrema (at different σ values) and there is no blob around this point

2D Blob Detection Summary

Detect a 3D stack of 2D Blobs of different scales (corresponding to different σ values) at different positions

Given an image $I(x,y)$

Convolve the image using NLoG at many scales σ

Find:

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

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

σ^*: Size of the blob

SIFT Detector

SIFT Detector: Proposed by David G. Lowe

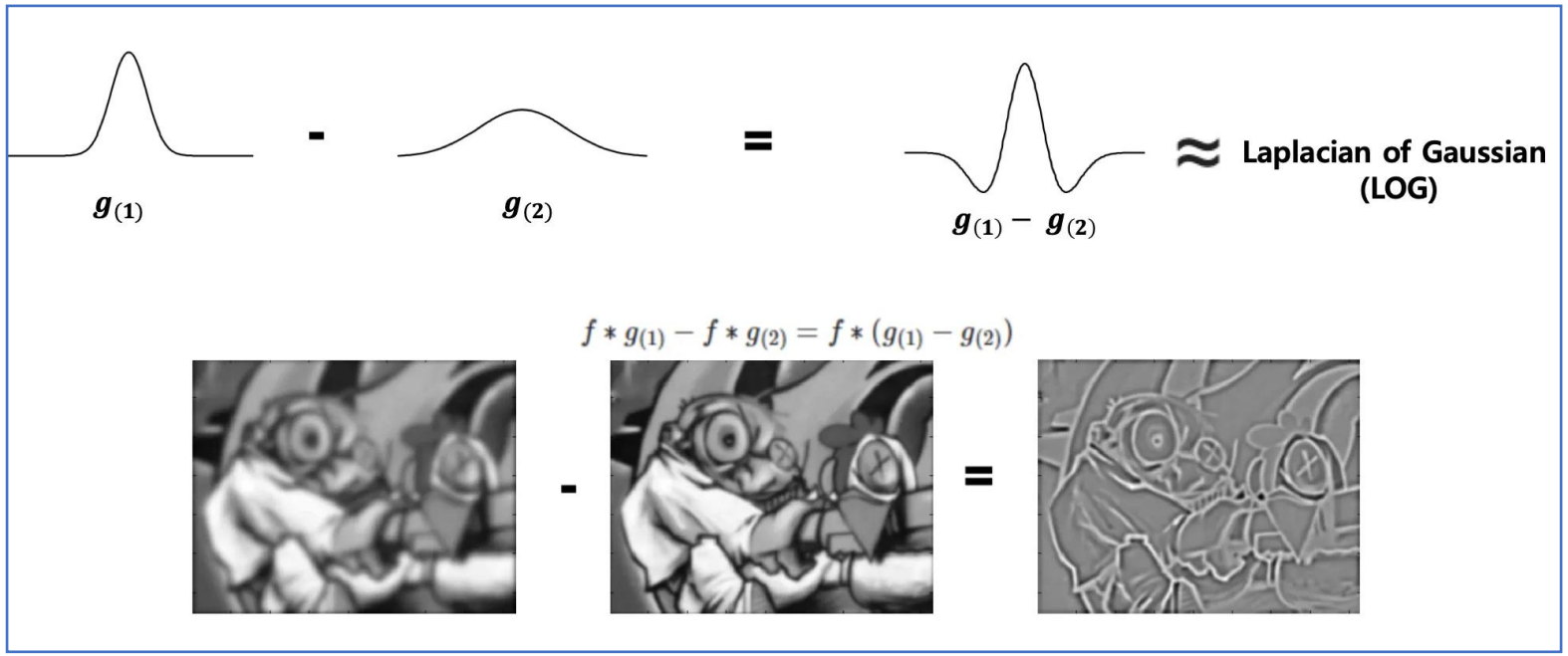
Very widely used in Computer Vision

Uses multiple tricks to make it reliable and efficient

Distinctive Image Features from Scale-Invariant Keypoints:
https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

Difference of Gaussian (DoG)

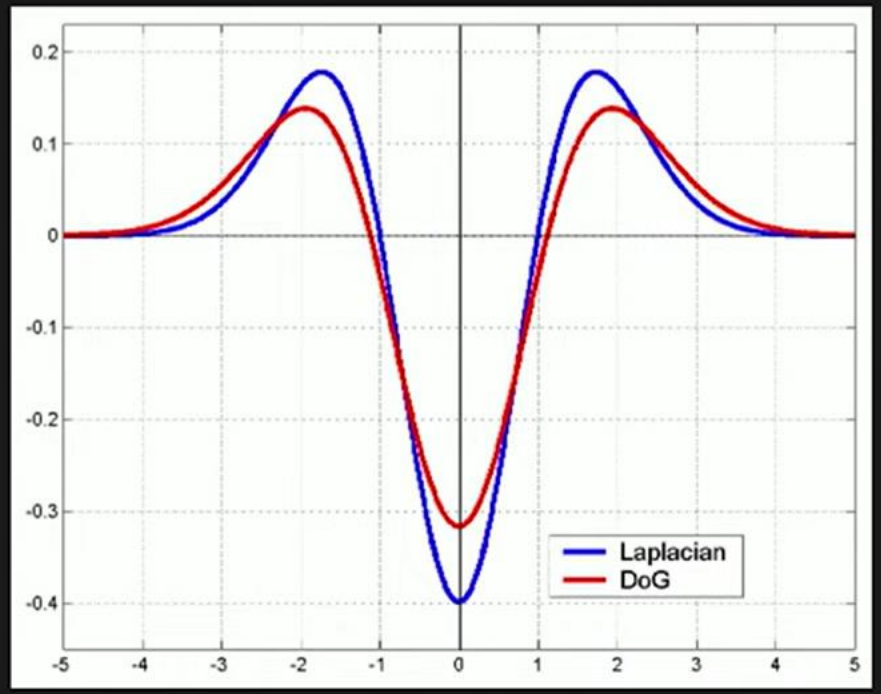
The Difference of Gaussian is a scaled version of Normalized Laplacian of Gaussian



- Subtracting one Gaussian by another approximates the Laplacian of Gaussian. For approximating the LoG, there is no actual derivative computation needed.
- Take the input image and Smooth the image with different σ values. Take consecutive σ-smoothed images and find the difference between them. The result you would get will be very close to the output you get if apply the Normalized Laplacian of Gaussian on the input image for a given σ value.
- The reason why DoG is beneficial is that it is common in computer vision tasks that an image is filtered by Gaussian filters/smoothing at many scales. While storing the Gaussian-filtering results of an image, we can easily make use of them to extract edges by computing the difference between subsequent filtering outputs, without actual computation of image derivatives.

Fast NLOG Approximation: DoG

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



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

SIFT Scale-Space Interest Points

❖ Scale-space function *L*

➢ Gaussian convolution

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$ where σ is the width of the Gaussian.

➢ Difference of Gaussian kernel is a close approximate to scale-normalized Laplacian of Gaussian

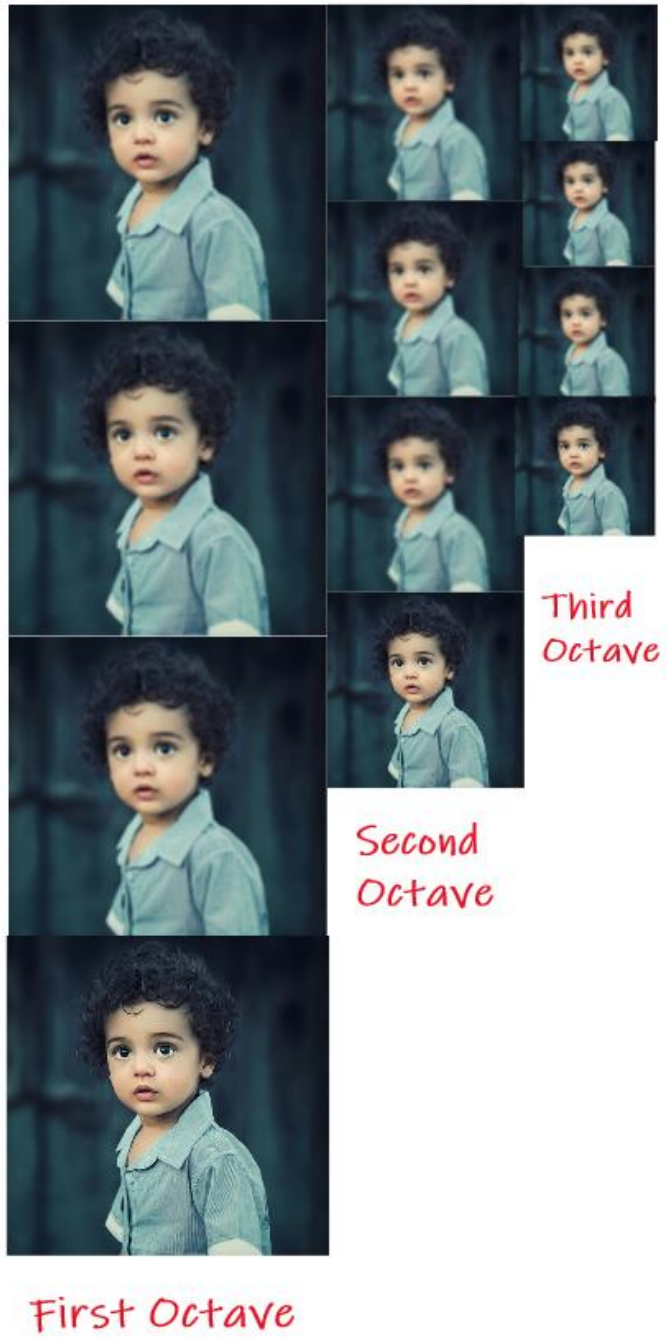
$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

2 scales:
σ *and* $k\sigma$

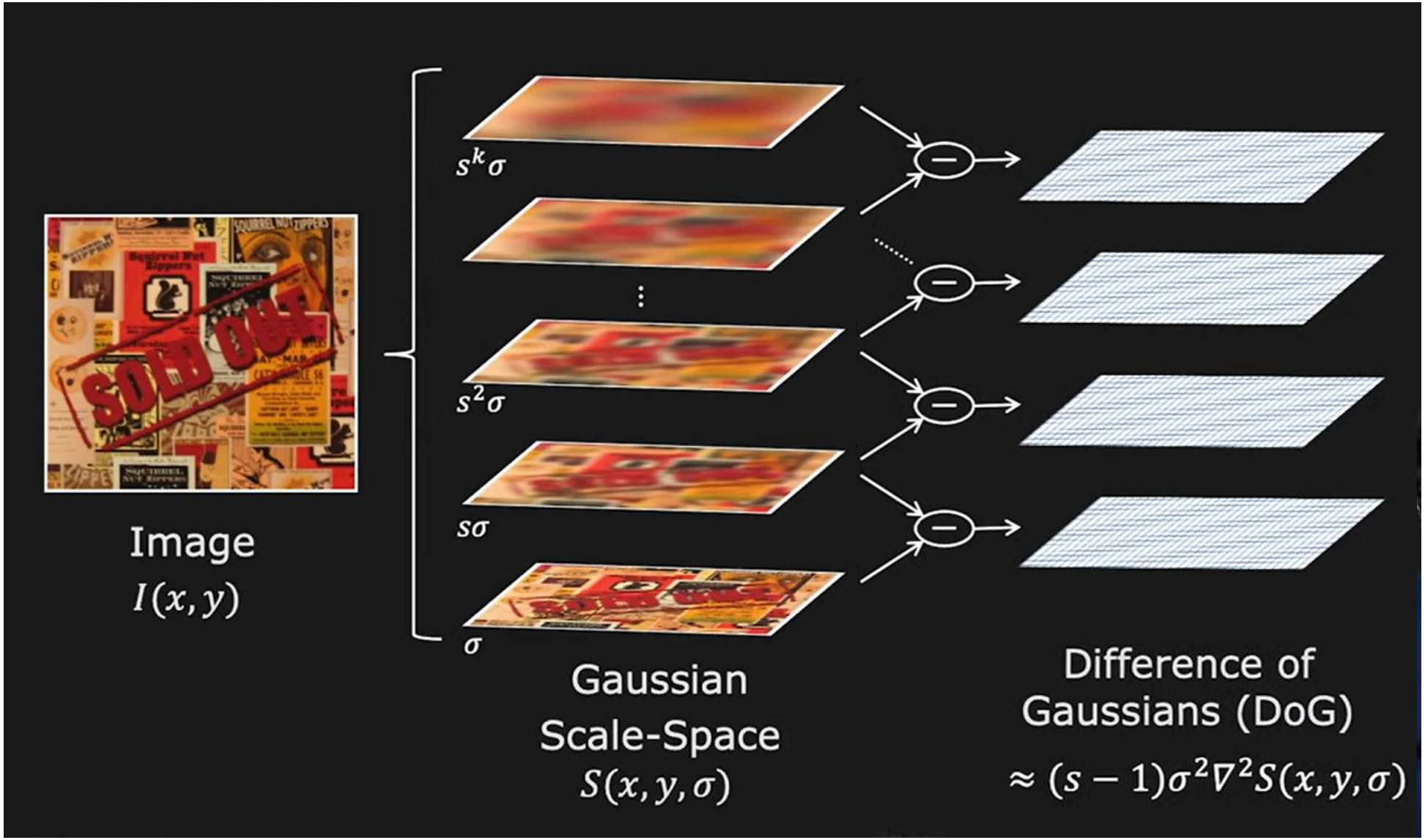
➢ Can approximate the Laplacian of Gaussian kernel with a difference of **separable** convolutions

SIFT Scale Space

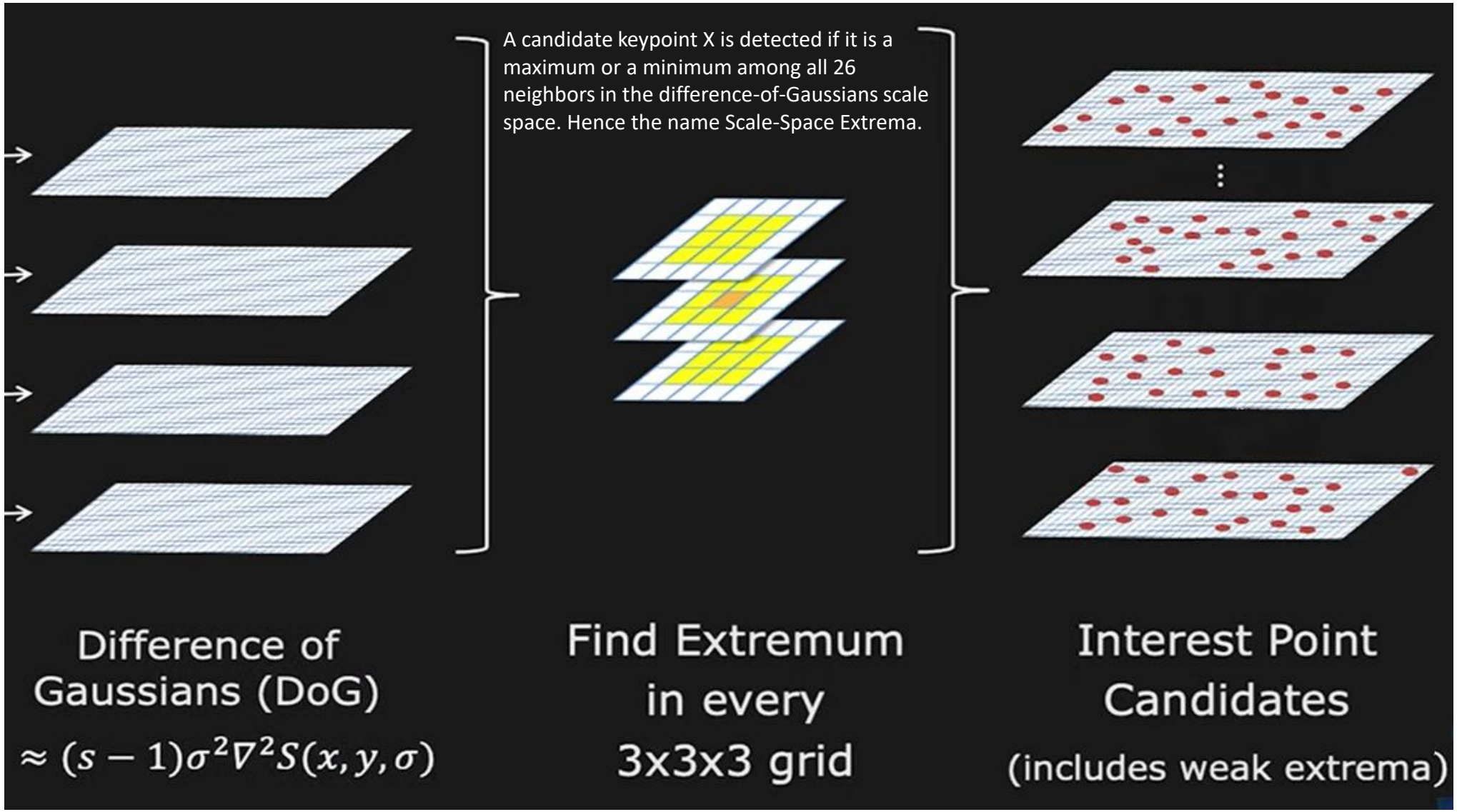
- In the SIFT paper, the authors modified the scale-space representation. Instead of creating the scale-space representation for the original image only, they created the scale-space representations for different image sizes. This helps in increasing the number of keypoints detected.
- The idea is:
  - Take the original image, and generate progressively blurred out images.
  - Then, resize the original image to half size. And generate blurred out images again. And keep repeating. This is shown below
  - The term octave to denote the scale-space representation for a particular image size. For instance, all the same size images in vertical line forms one octave. Here, we have 3 octaves and all the octaves contain 4 images at different scales (blurred using Gaussian filter).
  - Within an octave, the adjacent scales differ by a constant factor k. If an octave contains s+1 images, then k = 2(1/s). The first image has scale σ0, the second image has scale kσ0, the third image has scale k2σ0, and the last image has scale ksσ0. In the paper, they have used the values as number of octaves = 4, number of scale levels = 5, initial σ0 =1.6, k=√2 etc.



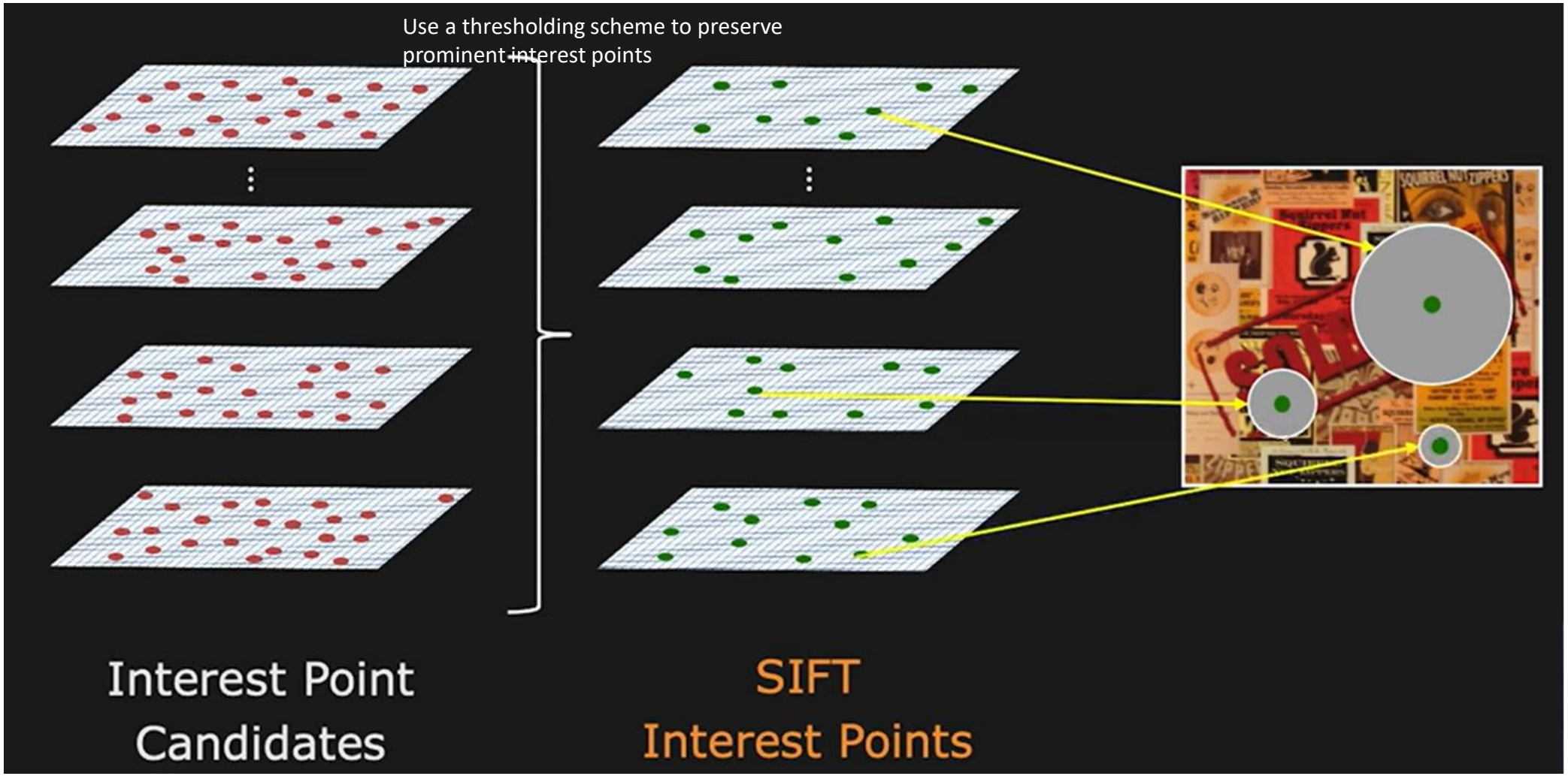
Extracting SIFT Interest Points (1)



Extracting SIFT Interest Points (2)



Extracting SIFT Interest Points (3)



Extracting SIFT Interest Points: Keypoint Localization and Filtering (3)

- Keypoints generated in the previous step produce a lot of keypoints. Some of them lie along an edge, or they don't have enough contrast. In both cases, they are not as useful as features and get rid of them. The approach is similar to the one used in the Harris Corner Detector for removing edge features. For low contrast features, simply check their intensities..
- **Eliminate Low Contrast Points**: Use Taylor series expansion of scale space to get a more accurate location of extrema, and if the intensity at this extrema is less than a threshold value (0.03 as per the paper), it is rejected.

  $|D(\hat{\mathbf{x}})|$ < 0.03

- **Eliminate Edges:** Edge points have high contrast in one direction, low in the other. compute principal curvatures from eigenvalues of 2x2 Hessian matrix (H), and limit ratio to eliminate Edges.

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

α : largest eigenvalue(λ_{max})

β : smallest eigenvalue(λ_{min})

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

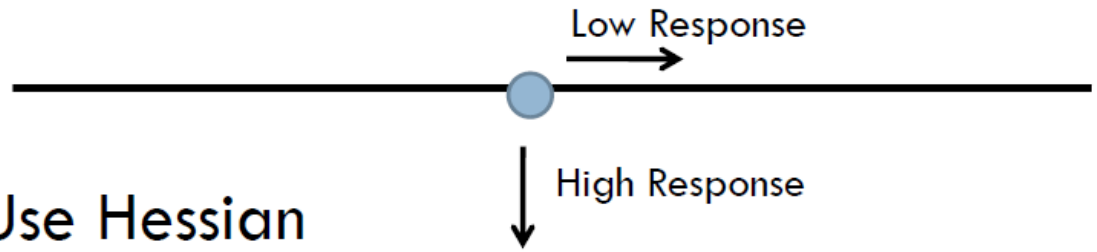
Evaluate Ratio: $\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$ where $r = \frac{\alpha}{\beta}$

$(r+1)^2/r$ is at a min when the two eigenvalues are equal (when r = 1) and it increases with r.

Reject Keypoints, if the ratio $\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})}$ > Threshold (Original SIFT uses r = 10)

Edge Response Elimination

- Peak has high response along edge, poor other direction



- Use Hessian
  - Eigenvalues Proportional to principle Curvatures
  - Use Trace and Determinant

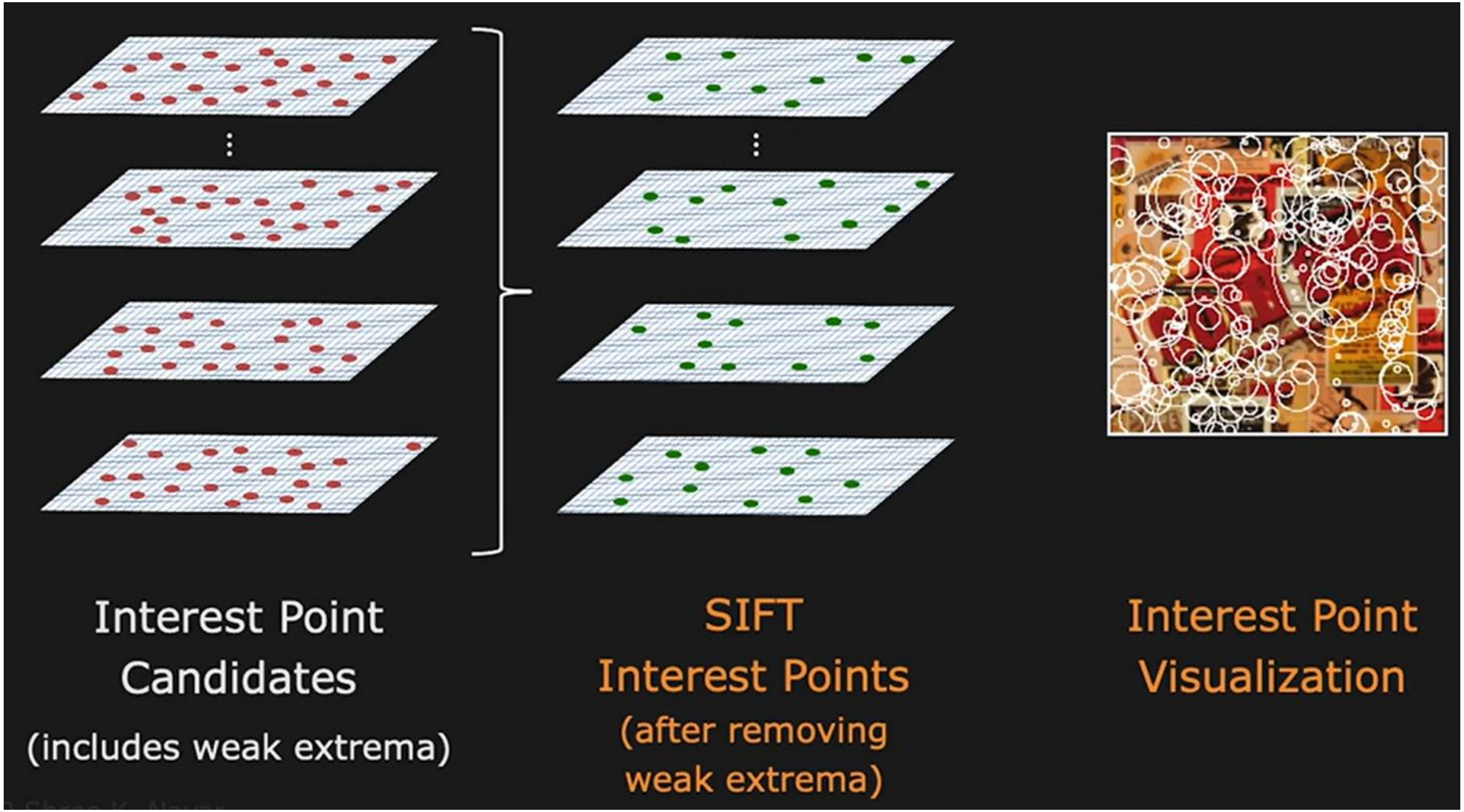
$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta, Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

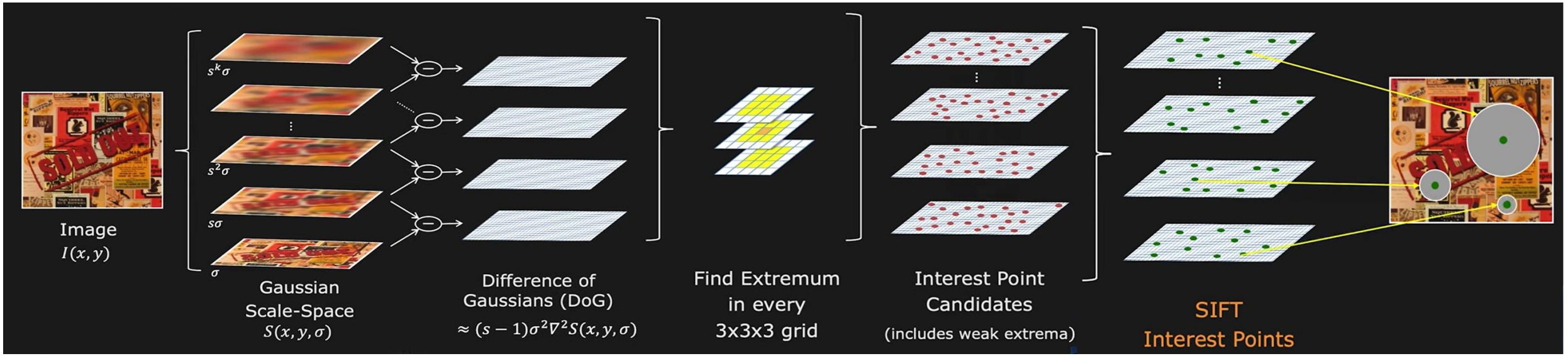
$(r+1)^2/r$ is at a min when the two eigenvalues are equal (when r = 1) and it increases with r.

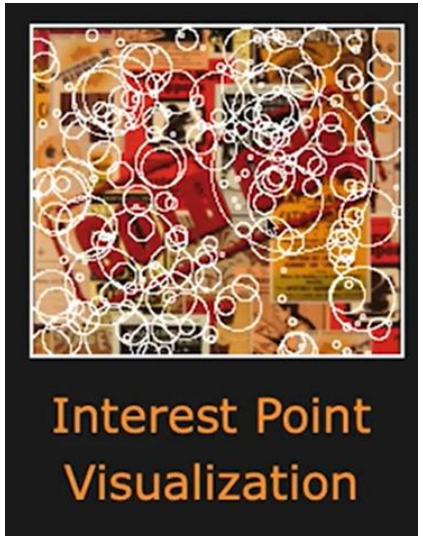
Reject Keypoints, if the ratio $\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})}$ > Threshold (Original SIFT uses r = 10)

Extracting SIFT Interest Points (4)



Extracting SIFT Interest Points Summary





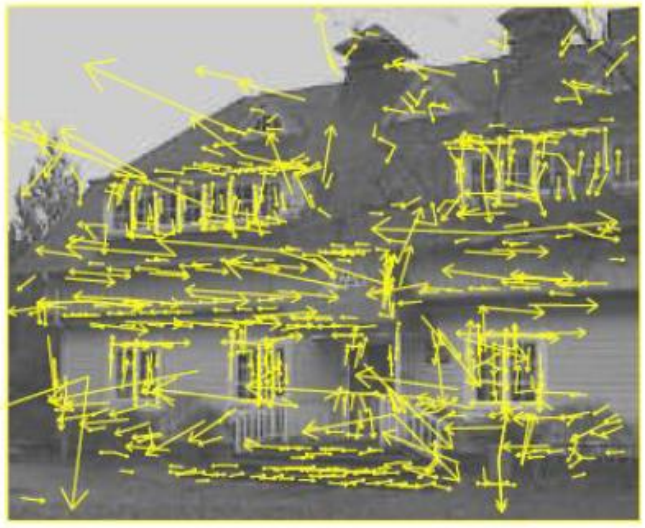
SIFT Interest Points

Input Image
(233X189 pixel image)

a)

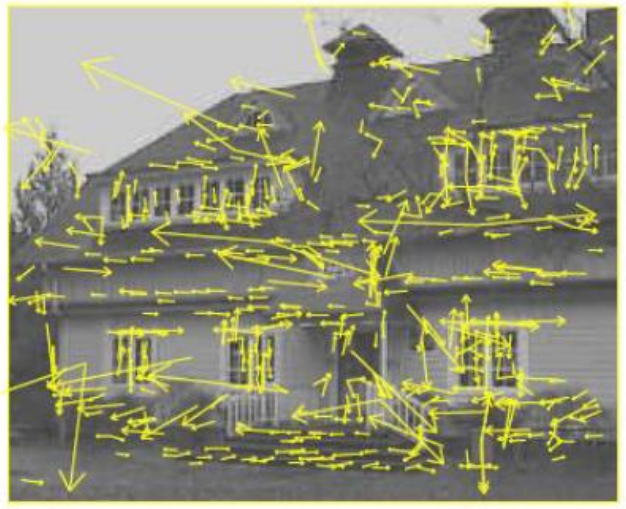
After DOG Extrema Localization
(832 key points)

b)



(c)

Apply Contrast Limit
(729 keypoints)

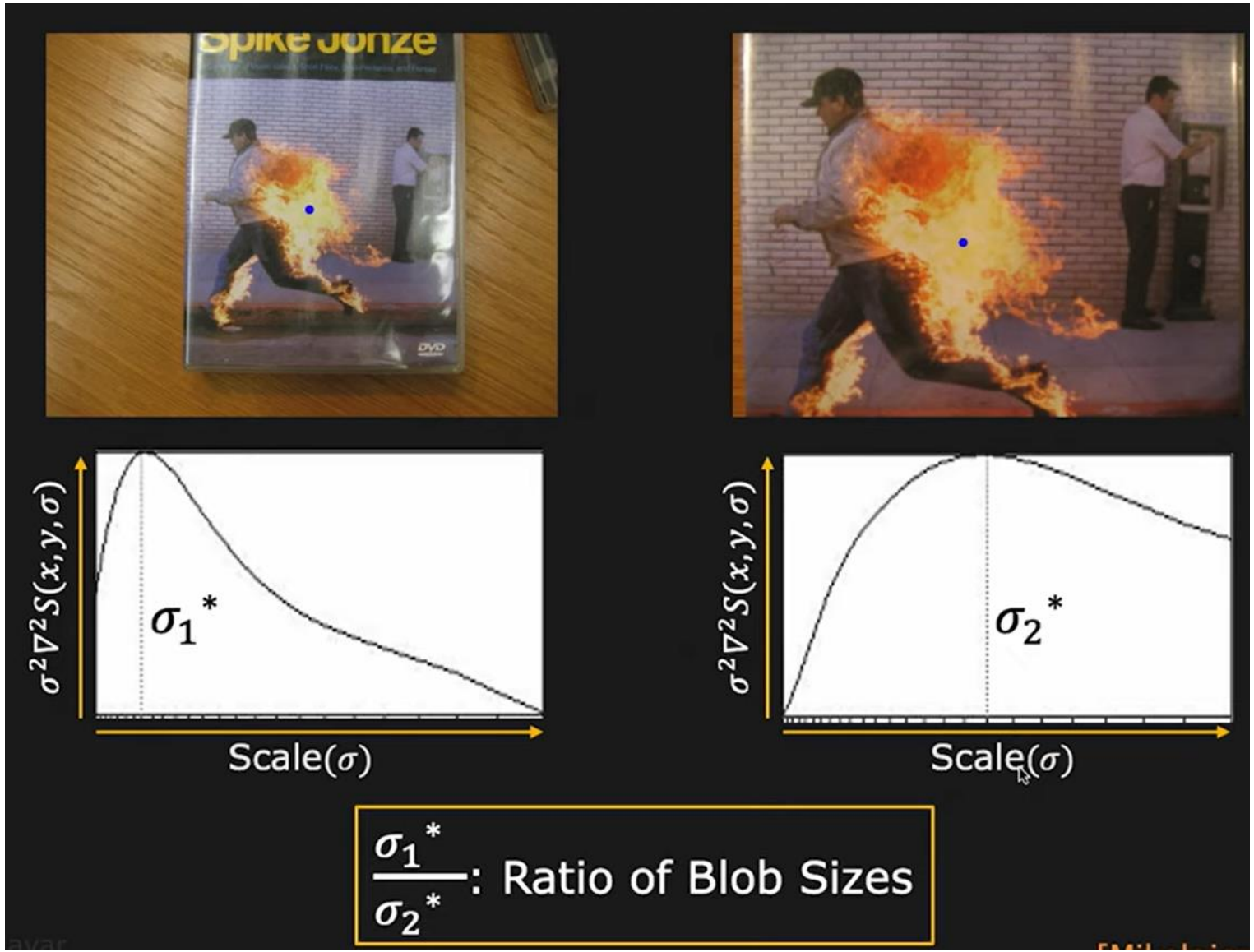


(d)

Apply Contrast Limit and Edge response Elimination
(536 key points)

SIFT Detection Examples

SIFT Scale Invariance



Find SIFT Feature in both images with same appearance.

NLOG peaks at two different σ values in the two images

One we know the σ values, we can adjust the scale of one to match the other.

Computing the Principal Orientation

Need to remove the affect of Orientation

Use the histogram of gradient directions

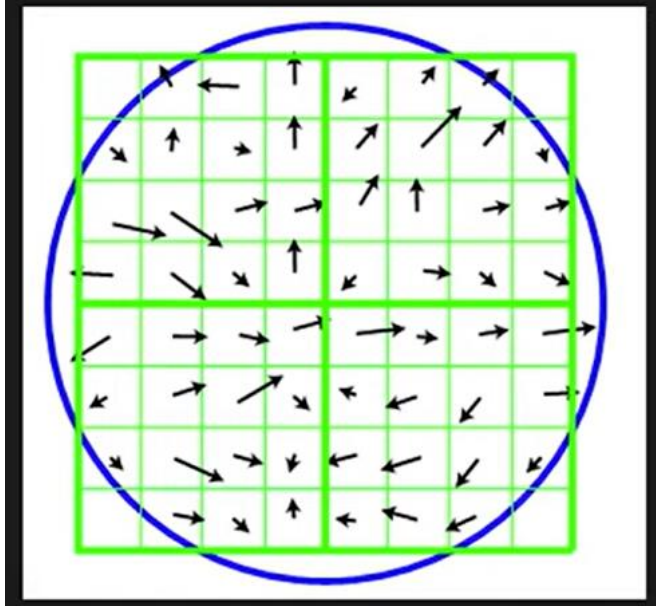


Image gradient directions

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

Image gradient direction is only considered as it is unaffected by Lighting conditions.

Gradient Magnitude is affected by

- Lighting conditions
- Gain of camera

….

Computing the Principal Orientation

Use the histogram of gradient directions

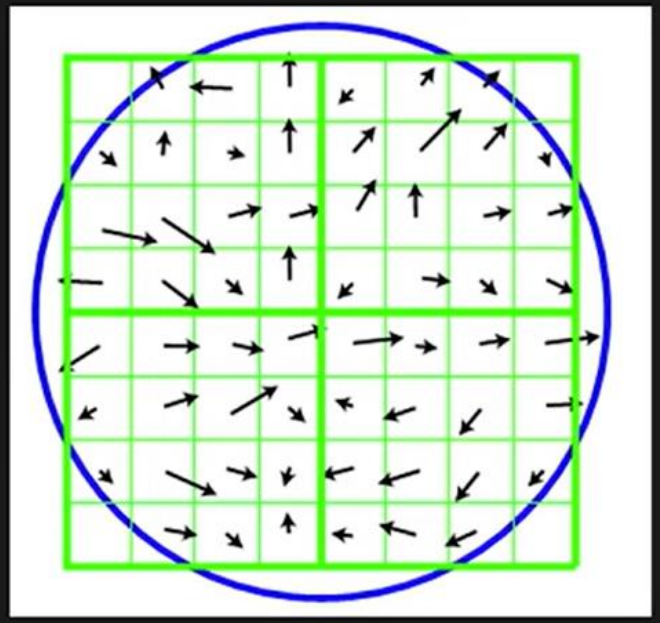
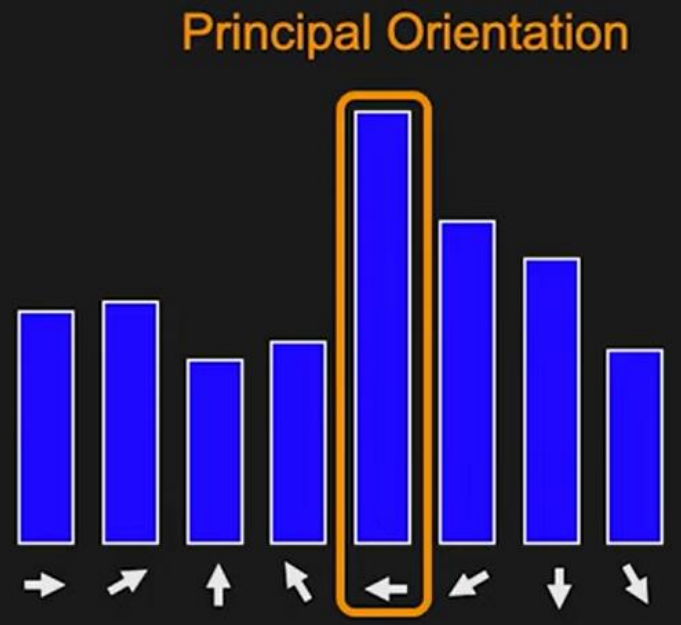


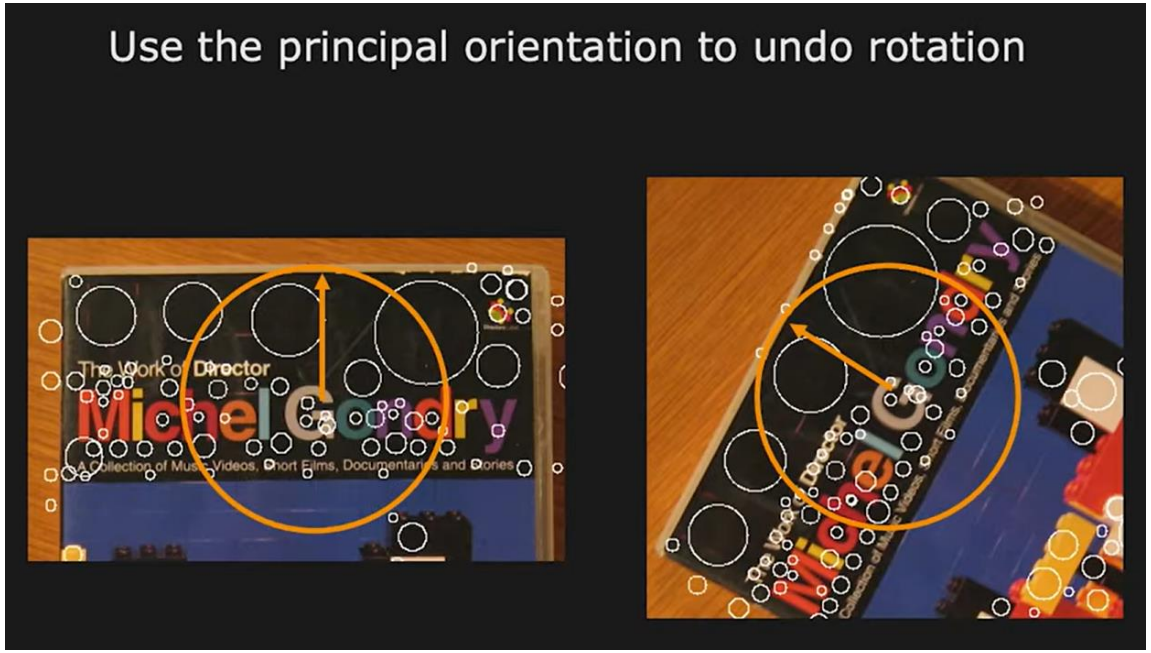
Image gradient directions

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



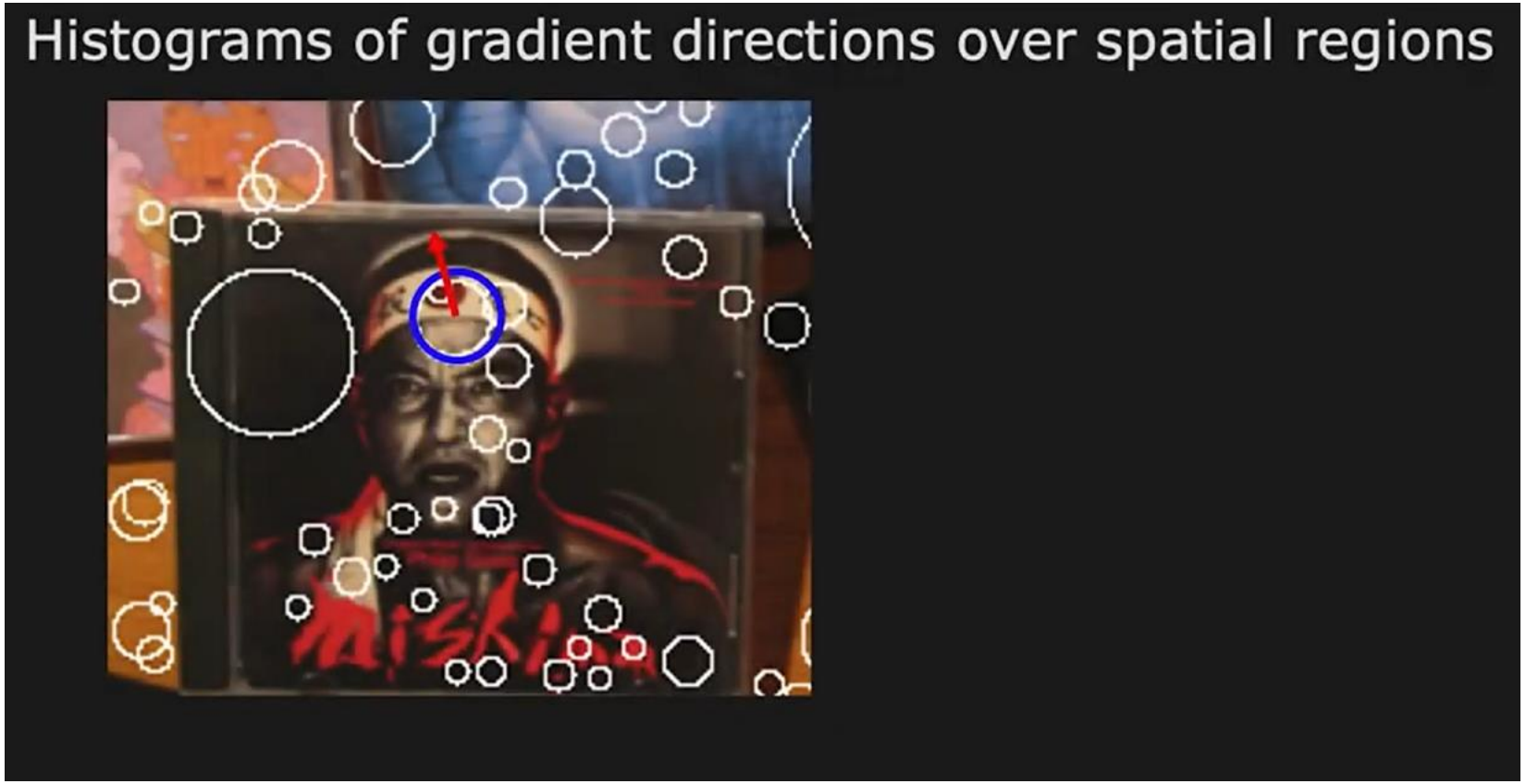
Choose the most prominent gradient direction

SIFT Rotation Invariance

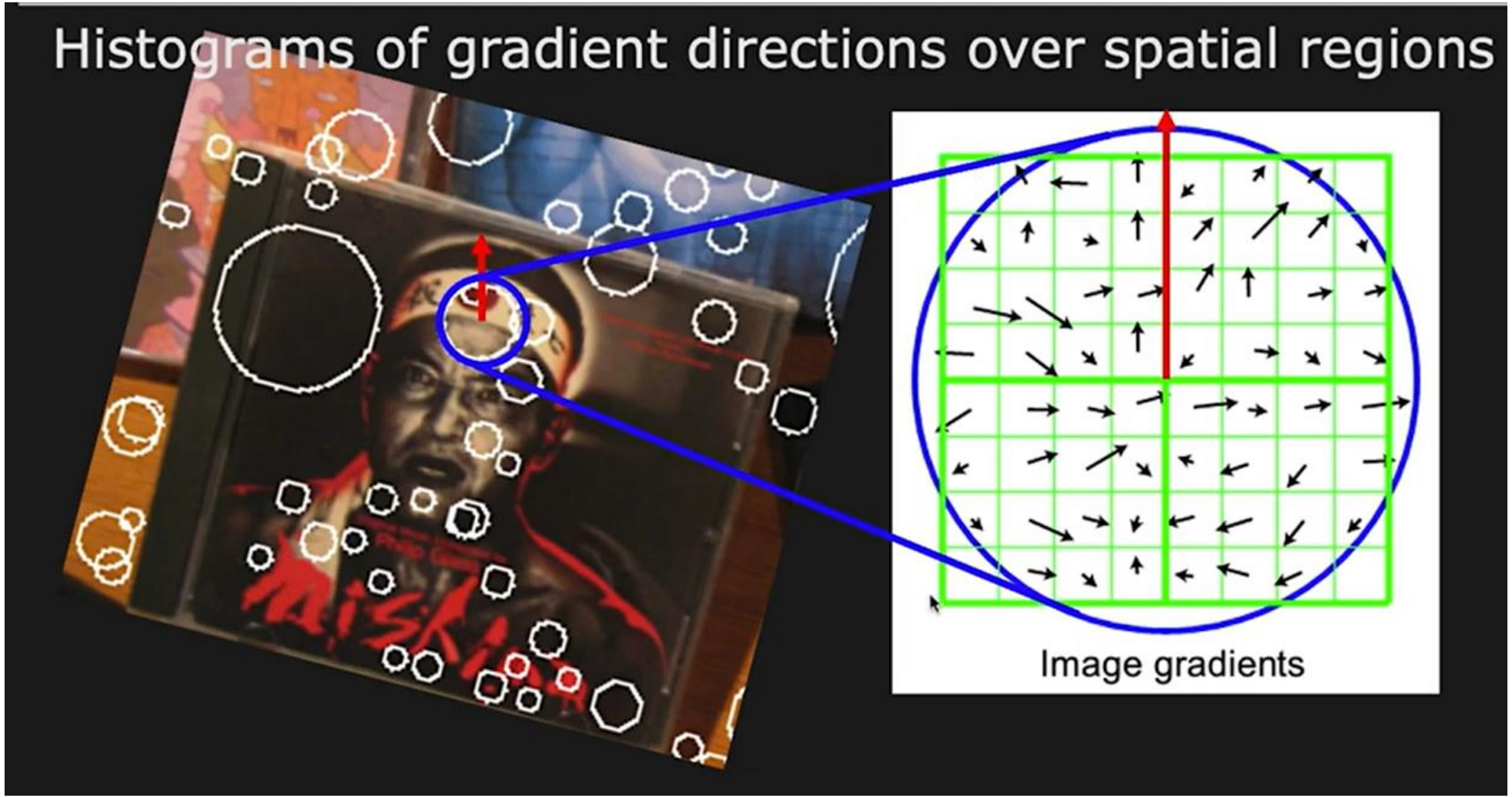


SIFT Descriptor

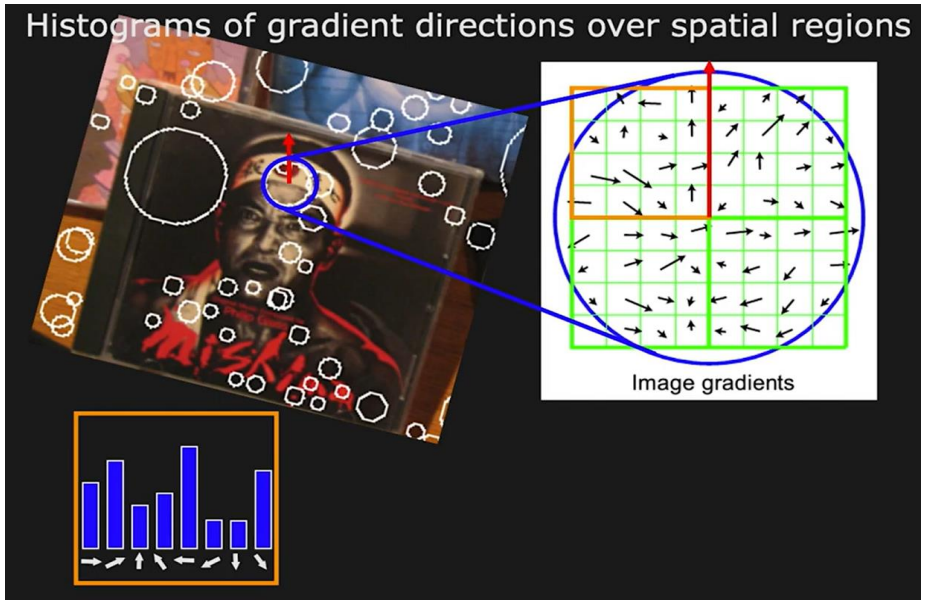
SIFT Descriptor

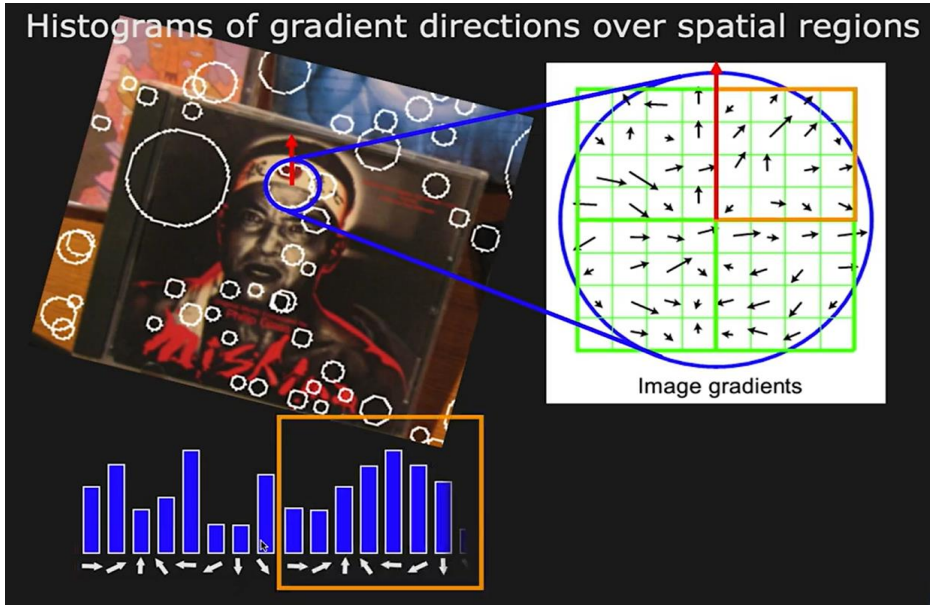


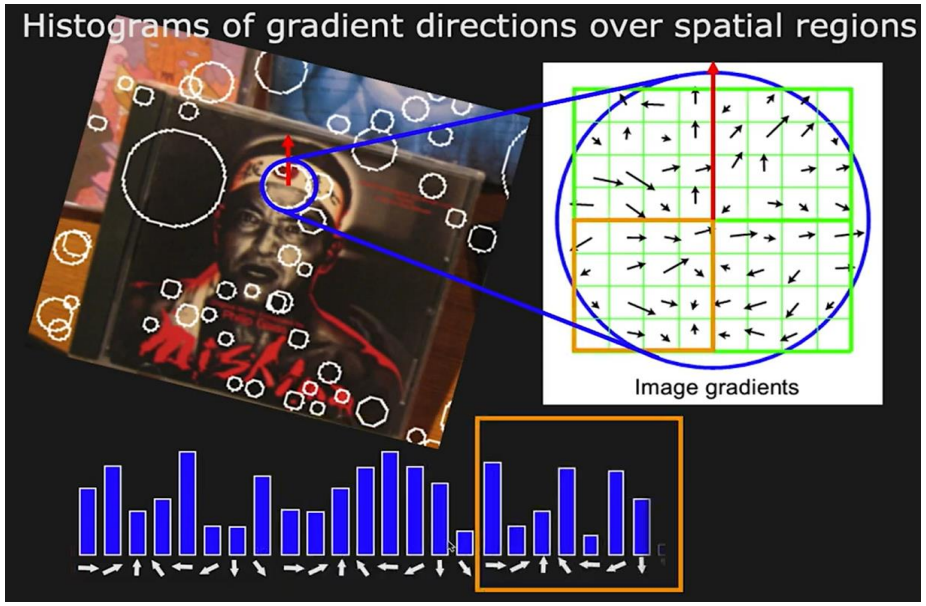
SIFT Descriptor

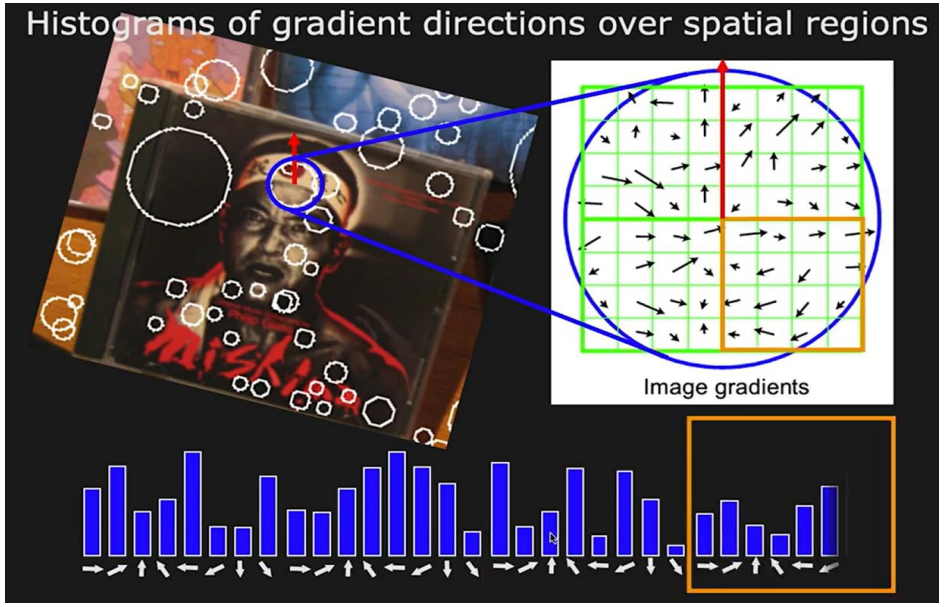


SIFT Descriptor

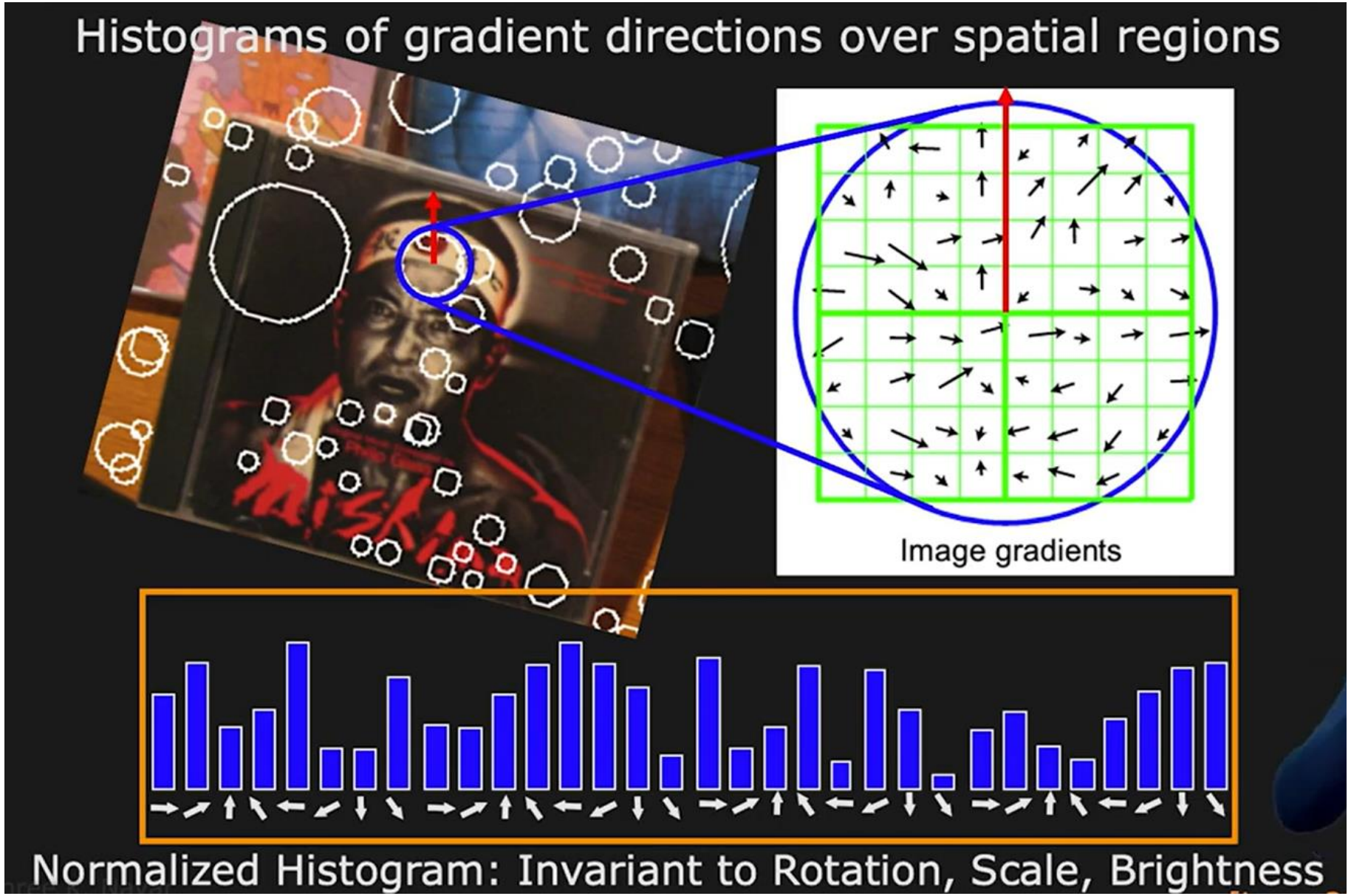






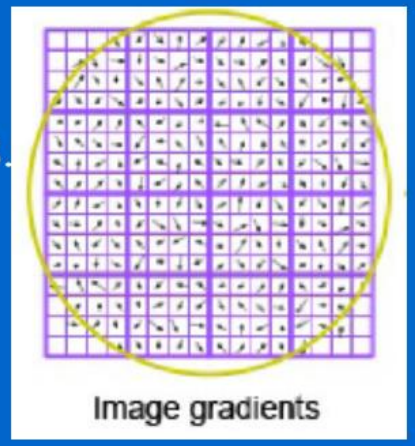


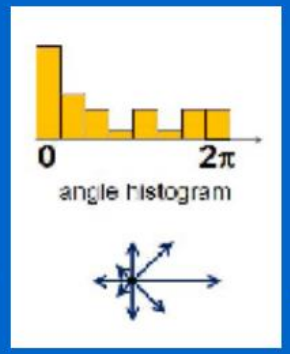
SIFT Descriptor



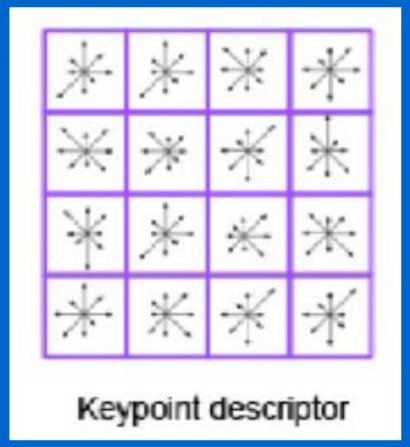
SIFT Keypoint Descriptor

1. Take a 16 x16 window around detected interest point.
2. Divide into a 4x4 grid of cells.
3. Compute histogram in each cell.





(8 bins)



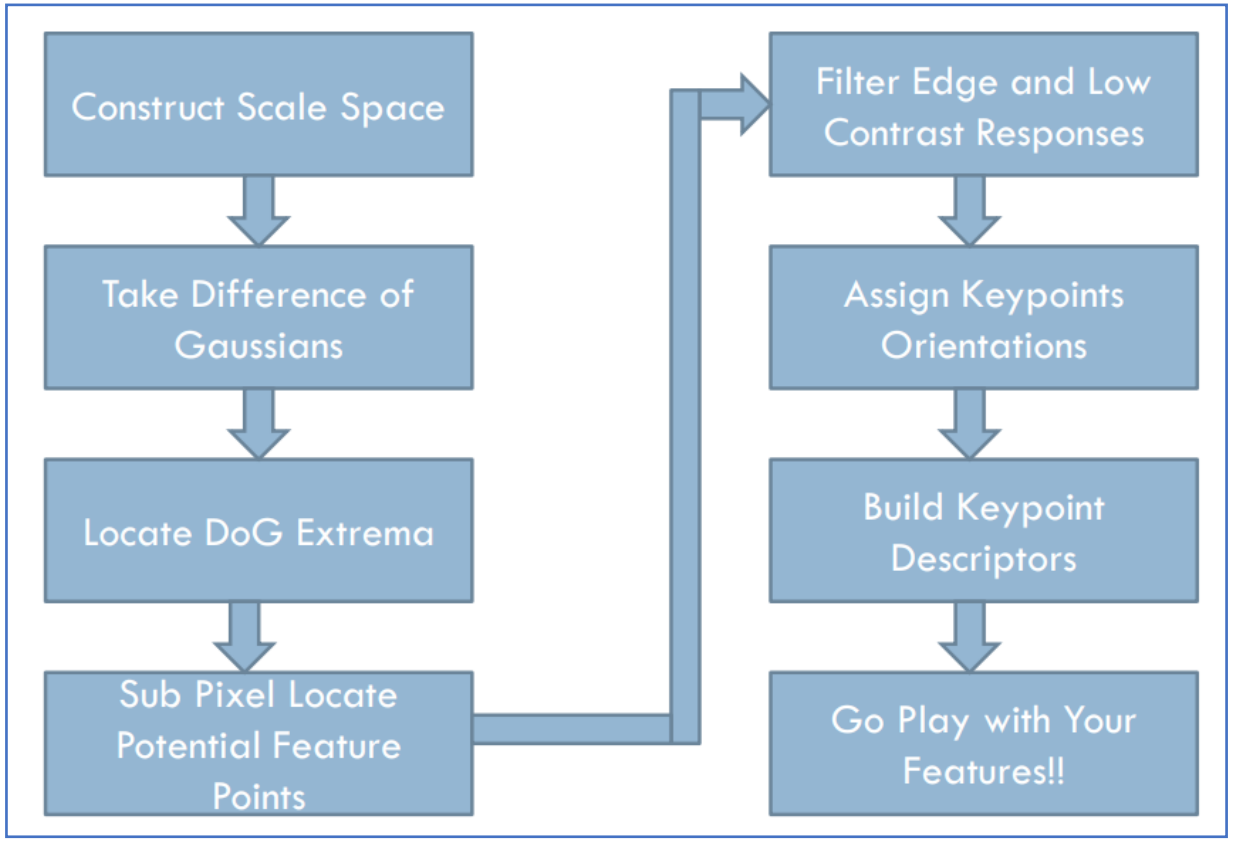
16 histograms x 8 orientations
= 128 features

SIFT Feature Descriptor for a keypoint:

- For each keypoint, a descriptor is created using the keypoints neighborhood. These descriptors are used for matching keypoints across images.
- A 16×16 neighborhood of the keypoint is used for defining the descriptor of that key-point.
- This 16×16 neighborhood is divided into sub-block. Each such sub-block is a non-overlapping, contiguous, 4×4 neighborhood.
- Subsequently, for each sub-block, an 8 bin orientation is created similarly as discussed in Orientation Assignment.
- These 128 bin values (16 sub-blocks * 8 bins per block) are represented as a vector to generate the keypoint descriptor.

SIFT Algorithm

- **Step 1. Scale-space Extrema Detection**: Search over all scales and image locations. It uses a Laplacian pyramid (difference-of-Gaussians) to identify potential interest points that are invariant to scale and orientation. These are the points where we are going to try to compute the SIFT keypoint.
- **Step 2. Keypoint Localization and filtering**: At each candidate location, a detailed model is fitted to determine location and scale. Unstable keypoints are eliminated from further processing.
- **Step 3. Orientation assignment**: Use local image gradients to assign orientation to each localized keypoint. Preserve orientation, scale and location for each feature.
- **Step 4. Creation of SIFT Keypoint Descriptor**: Extract local image gradients at selected scale around keypoint and form a representation invariant to local shape distortion and change in illumination.



Distinctive Image Features from Scale-Invariant Keypoints by David G. Lowe (2004) : https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

SIFT Advantages

- Major advantages of SIFT are:
  - Extraordinarily Robust feature detection
  - Changes in view point: up to about 60 degree out of plane rotation
  - Changes in illumination: sometimes even day vs night
  - Locality: features are local, so robust to occlusion and clutter (no prior segmentation)
  - Distinctiveness: individual features can be matched to a large database of objects
  - Quantity: many features can be generated for even small objects
  - Efficiency: Fast and Efficient. Can run in real-time
  - Extensibility: can easily be extended to a wide range of different feature types, with each adding robustness

SIFT Challenges

- SIFT is expensive (i.e., long) to compute.
  - Large amount of time in computing the Gaussian scale-space (in the detector part)
  - Computing histograms of the gradient direction (for the descriptor part).

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 $d(H_1, H_2) = 0$

Note: Subtract bin for bin, Square them up and add them up and find square root

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: $\bar{H}_i = \frac{1}{N} \sum_{k=1}^{N} H_i(k)$ Mean value of the histogram

Larger the distance metric, better the match.

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

Note: Divide by the Energies of the two histograms

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.

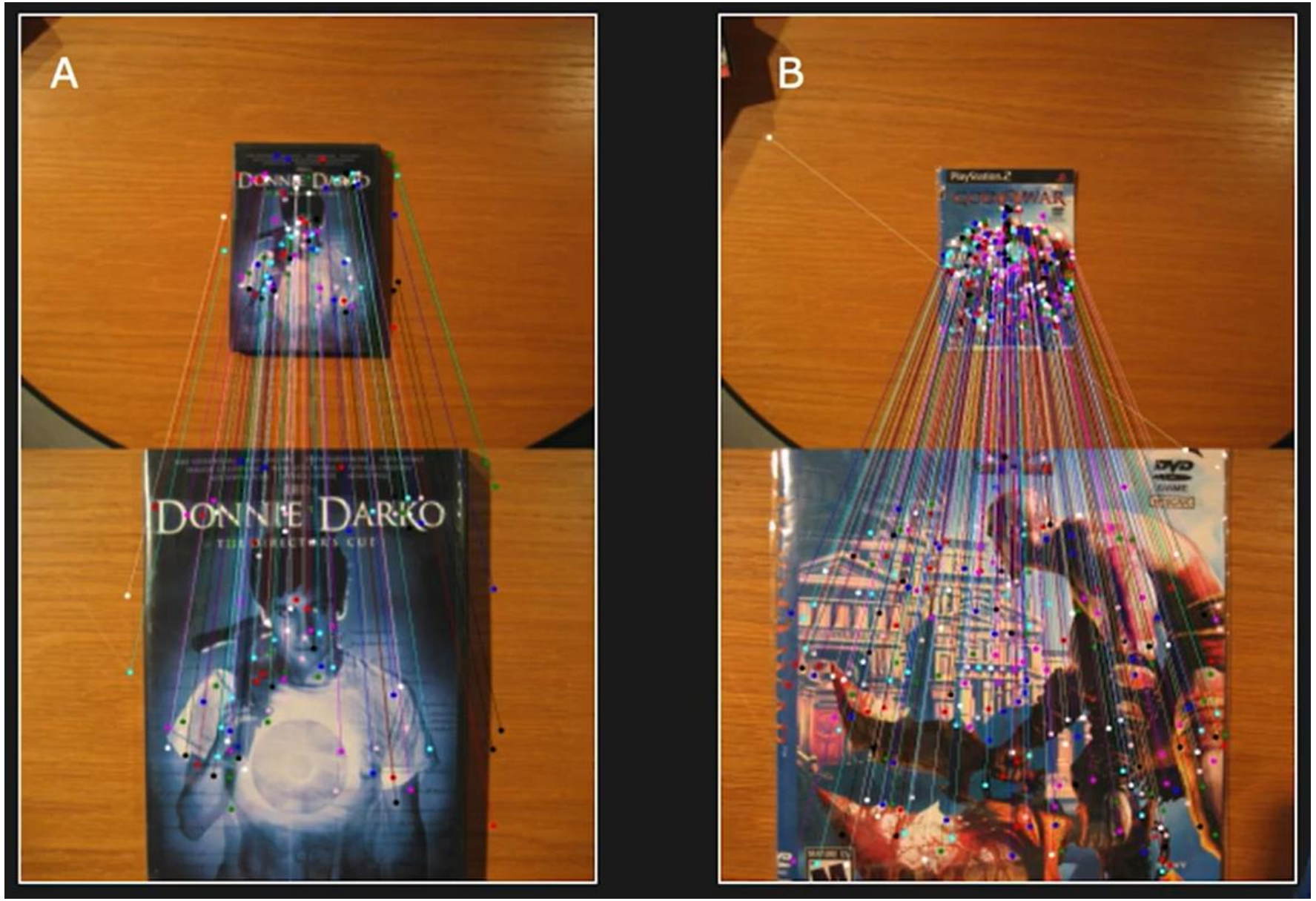
Intersection:

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

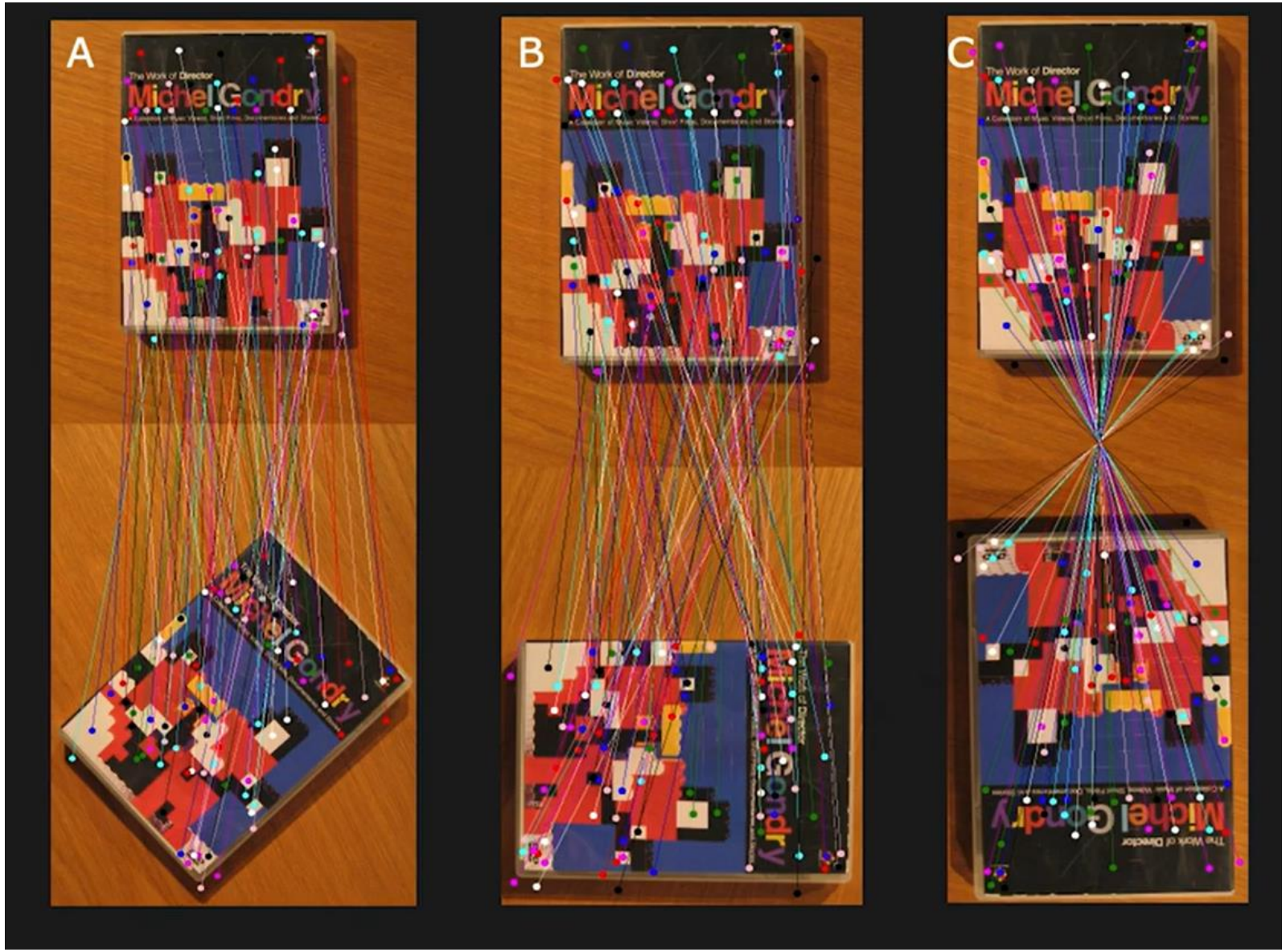
Larger the distance metric, better the match.

Note: Find Min value bin for bin, add them up.

SIFT Results: Scale Invariance

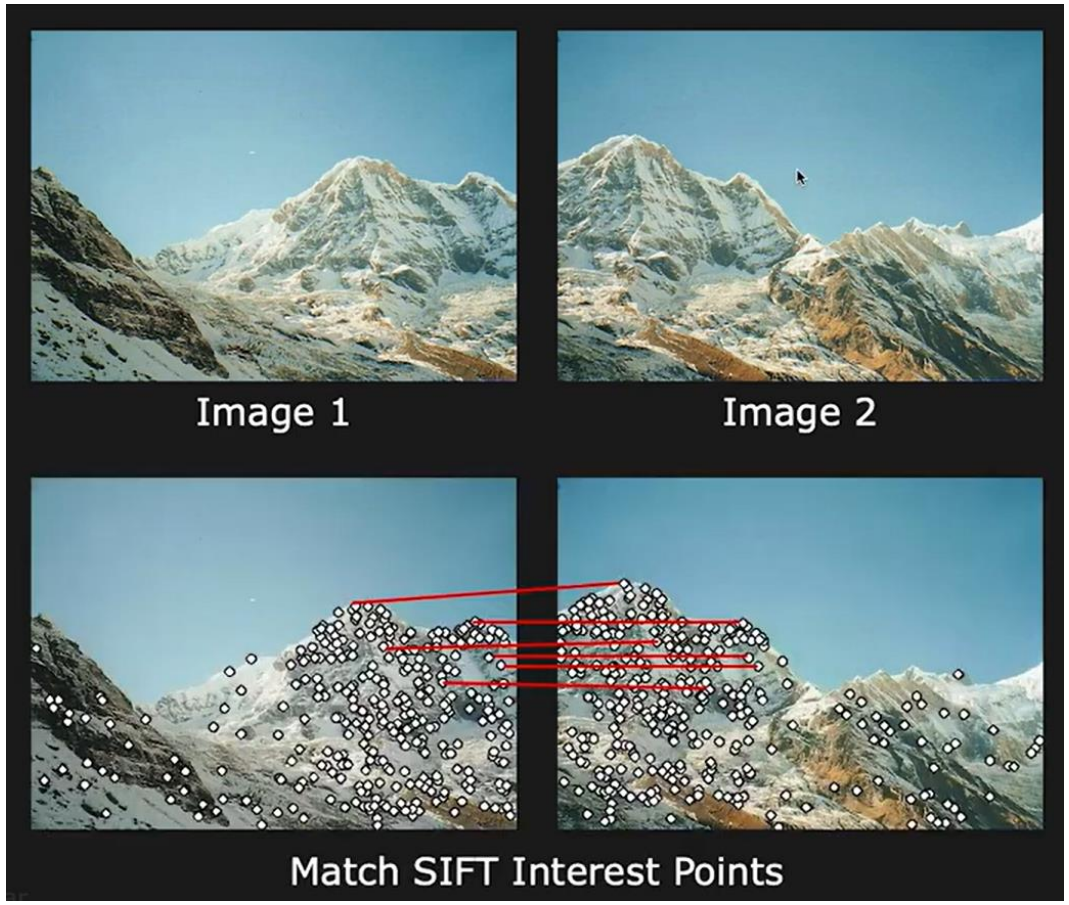


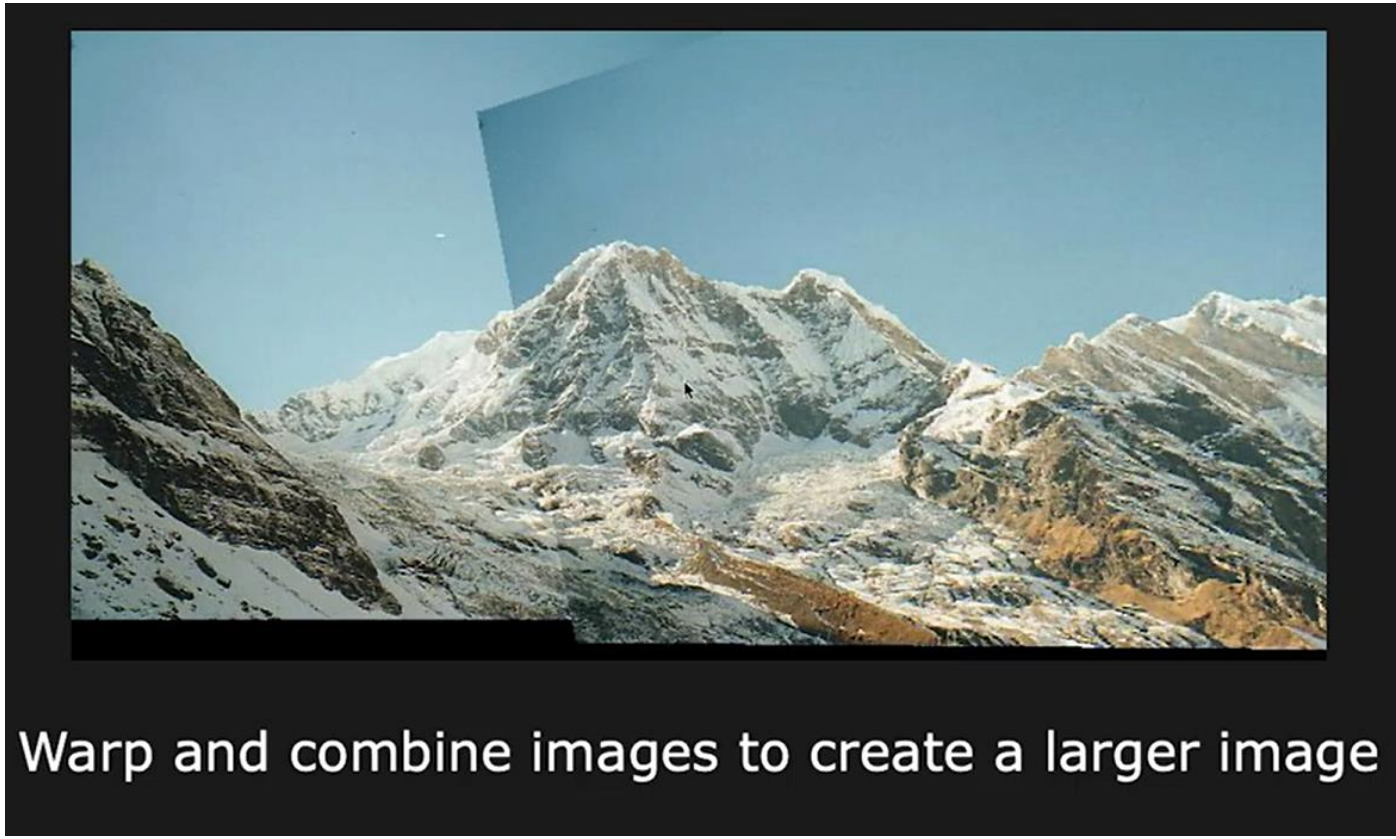
SIFT Results: Rotation Invariance



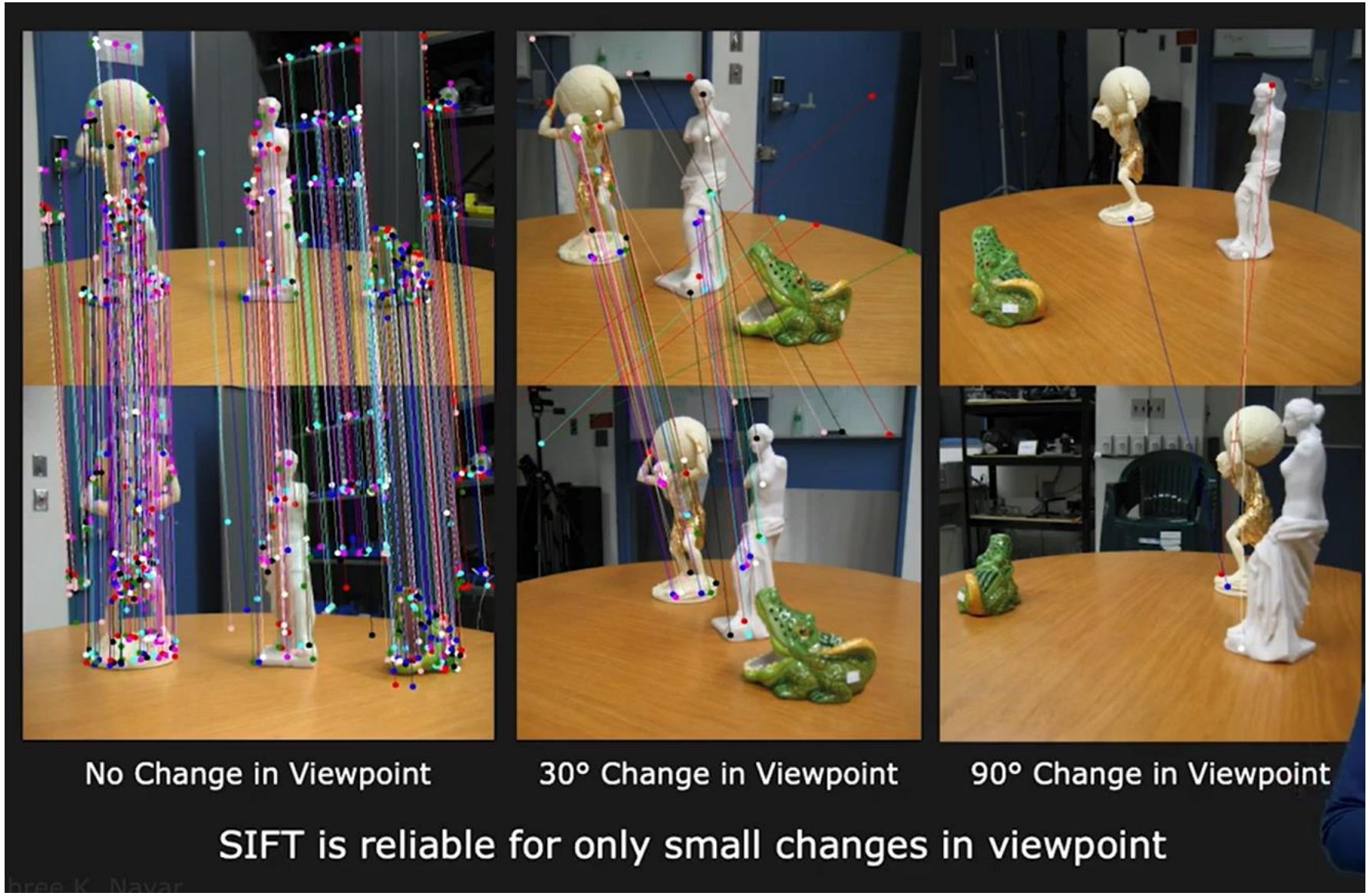
SIFT Results: Robustness to Clutter

Panorama Stitching using SIFT





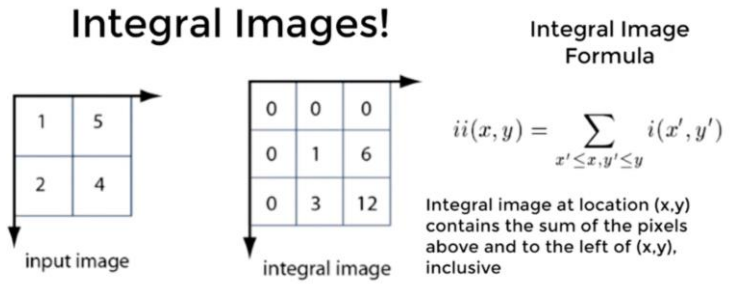
SIFT for 3D Objects

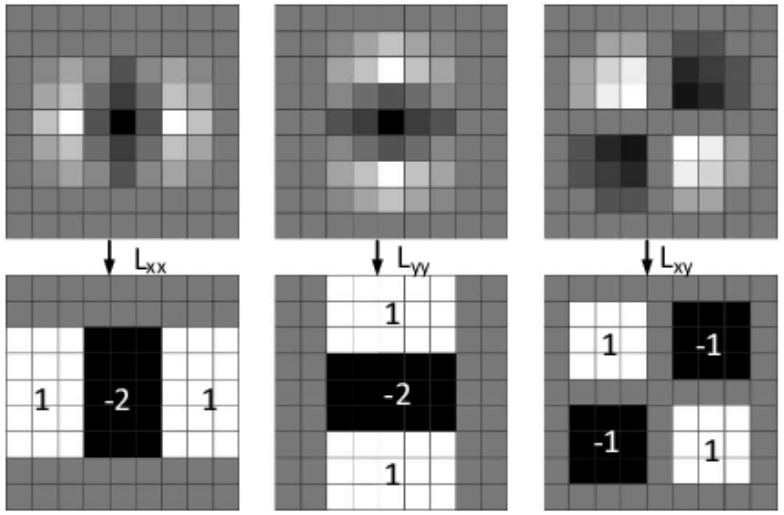


SIFT Variants

SURF: Speedup Robust Features

- Create an Integral image (to speeded up the calculations and save time)
- Uses box filters instead of Difference of Gaussians to approximate Laplacians
- Use Haar wavelets to get key point orientations
- SURF is good at Blur and rotation variations
- SURF is not as good as SIFT invariance to illumination and viewpoint changes
- Surf is ~3 times faster than SIFT





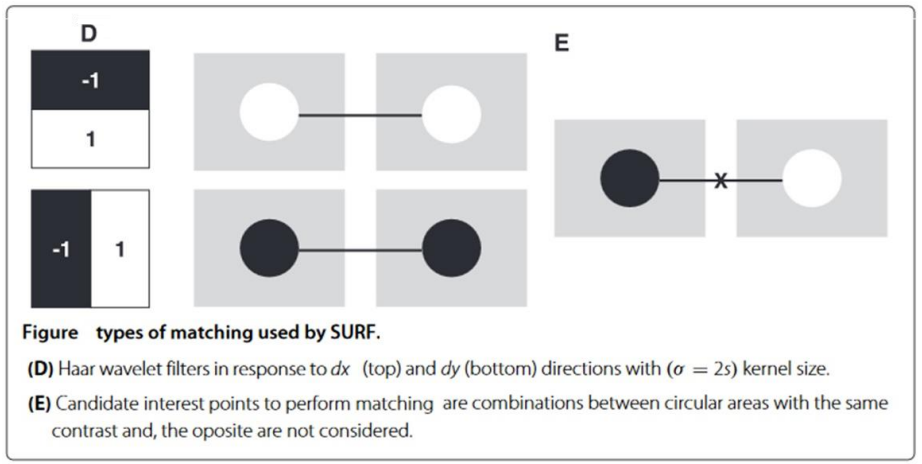


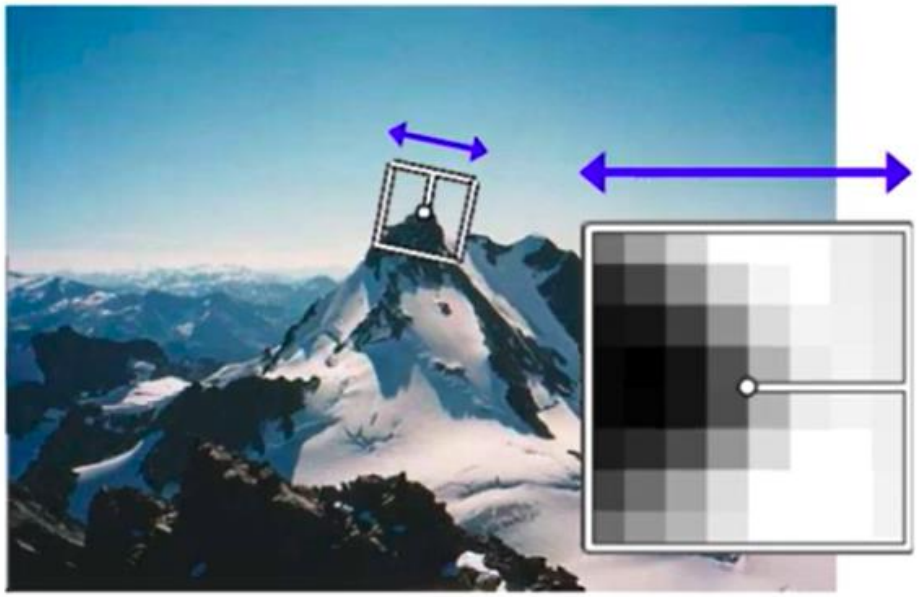
Figure types of matching used by SURF.

(D) Haar wavelet filters in response to *dx* (top) and *dy* (bottom) directions with ($\sigma = 2s$) kernel size.

(E) Candidate interest points to perform matching are combinations between circular areas with the same contrast and, the oposite are not considered.

MOPS: Making Descriptor Rotation-invariant

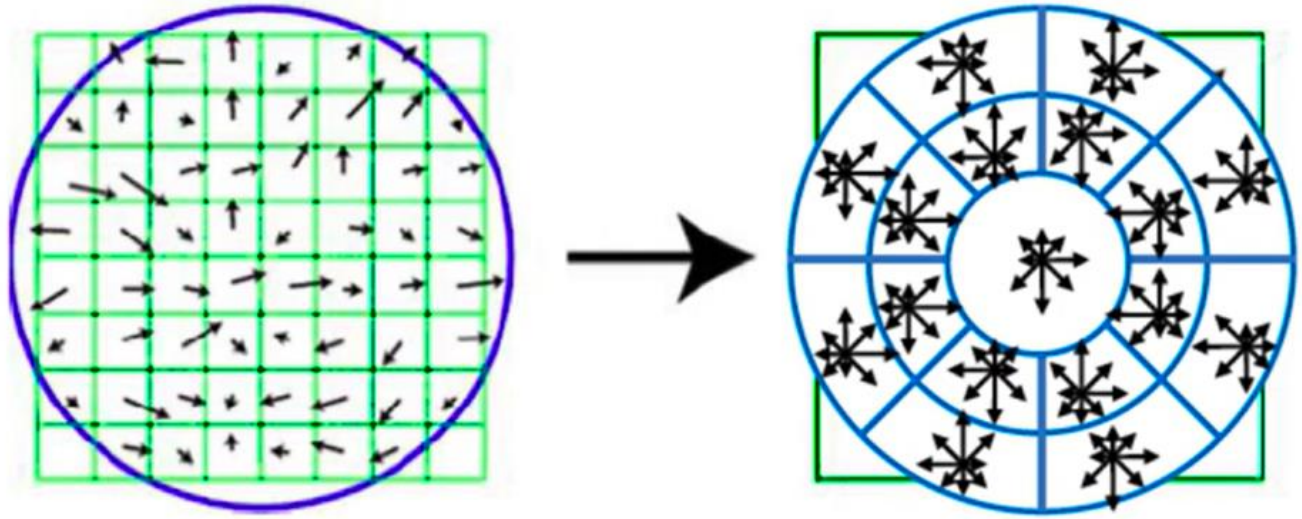
- **M**ultiscale **O**riented **P**atche**S** descriptor
- Rotate patch according to its dominant gradient orientation.
- This puts the patches into a canonical orientation



Credit: Matthew Brown, Kristen Grauman, Raquel Urtasun

Gradient Location-Orientation Histogram (GLOH)

- Variant of SIFT that uses a log-polar binning structure instead of four quadrants.
- Uses 17 spatial bins and 16 orientation bins.
- The 272D histogram is then projected onto a 128D descriptor using PCA trained on a large dataset.



Credit: Matthew Brown, Kristen Grauman, Raquel Urtasun

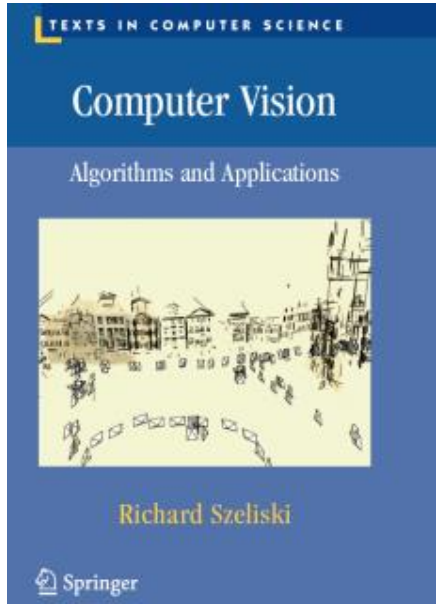
Text Books and References

❖Text Books

- Richard Szeliski. ”Computer vision: Algorithms and Applications. Springer Nature, Second Edition”, 2022
- E. R. Davies, Computer Vision Principles, Algorithms, Applications, Learning, Elsevier,5th Edition, 2017

❖ References

- Rafael C. Gonzales, Richard E. Woods, “Digital Image Processing”, Fourth Edition, Pearson Education, 2018.
- Richard Szeliski ,”Computer Vision: Algorithms and Applications” , Springer 2015
- Richard Szeliski. ”Computer Vision: Algorithms and Applications 2nd Edition – final draft”, 2021
- Intro to Digital Image Processing
- Digital Image Processing 2nd edition



PDF online

❖ Credits: Slides and Video content borrowed from:

- First Principles of Computer Vision
- https://robotacademy.net.au/
- MIT 6.S094: Computer Vision
- MIT Introduction to Deep Learning | 6.S191
- Digital Image Processing 2nd edition
- CSE/EE486 Computer Vision I (Penn State University)
- https://www.cs.umd.edu/class/fall2019/cmsc426-0201/files/21_SURF.pdf
- https://theailearner.com/2021/10/12/sift-scale-space-extrema-detection/