# Computer Vision (CSE3010)

**Dr. Susant Kumar Panigrahi**
**Assistant Professor**
**School of Electrical & Electronics Engineering**

VIT BHOPAL

# Module-1 Syllabus

**Digital Image Formation And Low Level Processing**:

- Overview and State-of-the-art, Fundamentals of Image Formation, Transformation: Orthogonal, Euclidean, Affine, Projective, Fourier Transform,
- Convolution and Filtering, Image Enhancement, Restoration, Histogram Processing.

# Module-2 Syllabus

**Depth Estimation And Multi-Camera Views:**

Depth Estimation and Multi-Camera Views: Perspective, Binocular Stereopsis: Camera and Epipolar Geometry; Homography, Rectification, DLT, RANSAC, 3-D reconstruction framework; Auto-calibration. apparel.

# Module-3 Syllabus

**Feature Extraction And Image Segmentation:**

- **Feature Extraction**: Edges - Canny, LOG, DOG; Line detectors (Hough Transform), Corners - Harris and Hessian Affine, Orientation Histogram, SIFT, SURF, HOG, GLOH, Scale-Space Analysis- Image Pyramids and Gaussian derivative filters, Gabor Filters and DWT.
- **Image Segmentation:** Region Growing, Edge Based approaches to segmentation, Graph-Cut, Mean-Shift, MRFs, Texture Segmentation; Object detection.

# Module-4 Syllabus

**Pattern Analysis And Motion Analysis:**

- **Pattern Analysis**: Clustering: K-Means, K-Medoids, Mixture of Gaussians, Classification: Discriminant Function, Supervised, Un-supervised, Semi-supervised; Classifiers: Bayes, KNN, ANN models;
- **Dimensionality Reduction**: PCA, LDA, ICA; Non-parametric methods. Motion Analysis: Background Subtraction and Modelling, Optical Flow, KLT, Spatio-Temporal Analysis, Dynamic Stereo; Motion parameter estimation.

# Module-5 Syllabus

**Shape From X:**

Light at Surfaces; Phong Model; Reflectance Map;

Albedo estimation; Photometric Stereo; Use of Surface Smoothness

Constraint; Shape from Texture, color, motion and edges.

**Guest Lecture on Contemporary Topics**

## Text Books

1. Richard Szeliski, Computer Vision: Algorithms and Applications, Springer-Verlag London Limited 2011.
2. Computer Vision: A Modern Approach, D. A. Forsyth, J. Ponce, Pearson Education, 2003.

## Reference Book(s):

1. R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison- Wesley, 1992.
2. Richard Hartley and Andrew Zisserman, Multiple View Geometry in Computer Vision, Second Edition, Cambridge University Press, March 2004.
3. K. Fukunaga; Introduction to Statistical Pattern Recognition, Second Edition, Academic Press, Morgan Kaufmann, 1990.

## Required Tools/Software/IDLE:

1. Python/jupyter-notebook/google-colab
2. OpenCV
3. MATLAB

## Indicative List of Experiments:

1. Implement image preprocessing and Edge
2. Implement camera calibration methods
3. Implement Projection
4. Determine depth map from Stereo pair
5. Construct 3D model from Stereo pair
6. Implement Segmentation methods
7. Construct 3D model from defocus image
8. Construct 3D model from Images
9. Implement optical flow method
10. Implement object detection and tracking from video
11. Face detection and Recognition
12. Object detection from dynamic Background for Surveillance
13. Content based video retrieval
14. Construct 3D model from single image

# Computer Vision

## Unit – 03

# Object Recognition: SIFT and Scale Space Analysis

Standing on the shoulder of Giants: Ref: Few Slides borrowed from:

1. Prof. Shree Nayar, *First Principles of Computer Vision is a lecture series*.
2. Prof. Mubarak Shah, *Computer Vision Video Lectures.*

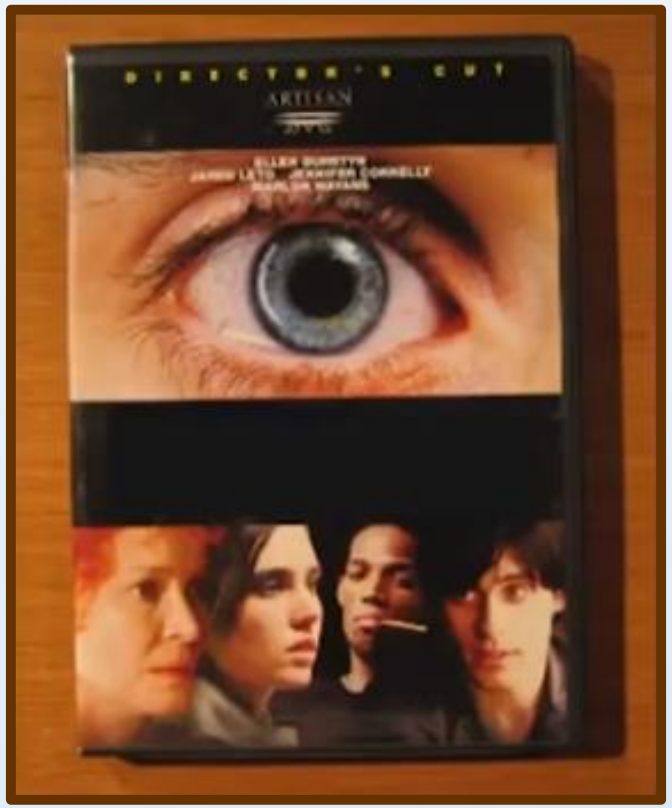
# Object Recognition: Quiz

How would you recognize the following types of objects?

- ✓ This is a fairly simple objects with distinctive background and occupies non-overlapping regions.
- ✓ Simple Image threhsolding or Binary image manupulation can be used to recognize the objects.

# Object Recognition: Quiz

How would you recognize the following types of objects?

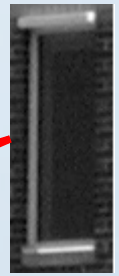



Template


Recognizing and Identifying Selected Template in the 2D rich Image

# Object Recognition: by Template Matching

- ✓ It is a process of finding templates in source (larger) image by comparing the exact match of base template (sub-image).
- ✓ It can be thought of a simplest approach of object detection.
- ✓ It scans the entire (larger) image (where the template is supposed to be located) for a given template by sliding it on the larger image to find the best match to fit. It is a similarity based approach that correlates the template with each locations of larger image.
- ✓ **Note**: The template which is provided must be exact match to the image.



**Template:** $T(x, y)$

**Image:** $f(x, y)$

# Template Matching: Similarity Measures

| | |
|---|---|
| TM_SQDIFF<br>Python: cv.TM_SQDIFF | $$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$<br>with mask:<br>$$R(x,y) = \sum_{x',y'} ((T(x',y') - I(x+x',y+y')) \cdot M(x',y'))^2$$ |
| TM_SQDIFF_NORMED<br>Python: cv.TM_SQDIFF_NORMED | $$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$<br>with mask:<br>$$R(x,y) = \frac{\sum_{x',y'} ((T(x',y') - I(x+x',y+y')) \cdot M(x',y'))^2}{\sqrt{\sum_{x',y'} (T(x',y') \cdot M(x',y'))^2 \cdot \sum_{x',y'} (I(x+x',y+y') \cdot M(x',y'))^2}}$$ |
| TM_CCORR<br>Python: cv.TM_CCORR | $$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$<br>with mask:<br>$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y') \cdot M(x',y')^2)$$ |

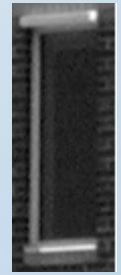
# Template Matching: Similarity Measures

| | |
|---|---|
| TM_CCORR_NORMED<br>Python: cv.TM_CCORR_NORMED | $$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$<br>with mask:<br>$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y') \cdot M(x',y')^2)}{\sqrt{\sum_{x',y'} (T(x',y') \cdot M(x',y'))^2 \cdot \sum_{x',y'} (I(x+x',y+y') \cdot M(x',y'))^2}}$$ |
| TM_CCOEFF<br>Python: cv.TM_CCOEFF | $$R(x,y) = \sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))$$<br>where<br>$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$<br>$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$<br>with mask:<br>$$T'(x',y') = M(x',y') \cdot \left(T(x',y') - \frac{1}{\sum_{x'',y''} M(x'',y'')} \cdot \sum_{x'',y''}(T(x'',y'') \cdot M(x'',y''))\right)$$<br>$$I'(x+x',y+y') = M(x',y') \cdot \left(I(x+x',y+y') - \frac{1}{\sum_{x'',y''} M(x'',y'')} \cdot \sum_{x'',y''}(I(x+x'',y+y'') \cdot M(x'',y''))\right)$$ |
| TM_CCOEFF_NORMED<br>Python: cv.TM_CCOEFF_NORMED | $$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$ |

# Template Matching: Similarity Measures

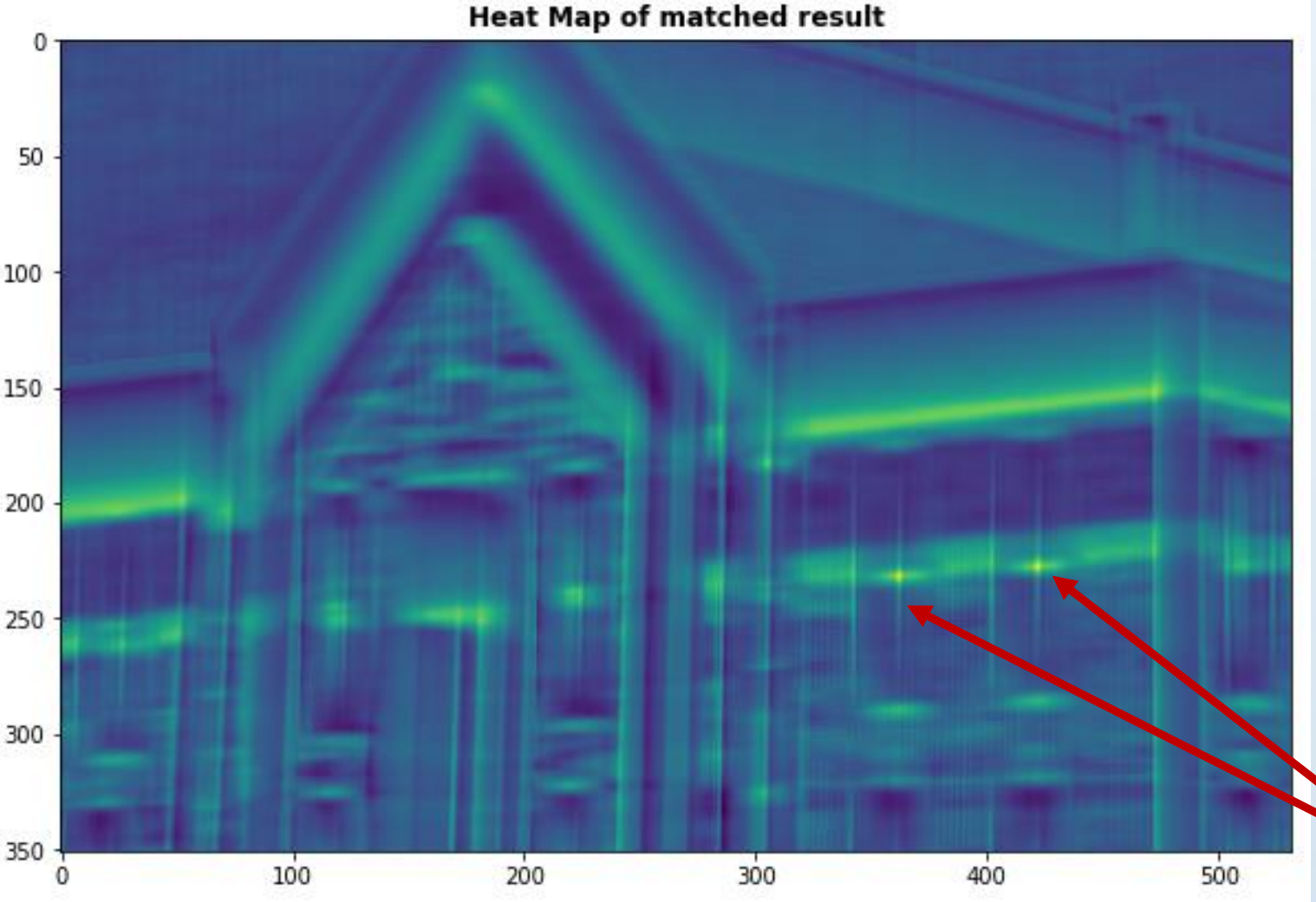
```
image = cv2.imread('/content/Building.tif')
template = cv2.imread('/content/Template.tif')

# # resize images
image = cv2.resize(image, (0,0), fx=0.5, fy=0.5)
template = cv2.resize(template, (0,0), fx=0.5, fy=0.5)

# Convert to grayscale
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
templateGray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

# Find template
# All the 6 methods for comparison in a list
# Note: The last two methods 'cv2.TM_SQDIFF' and 'cv2.TM_SQDIFF_NORMED provides
# best match on minimum value. However other methods provides best match at
# maximum value.
# methods = ['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED', 'cv2.TM_CCORR',
#            'cv2.TM_CCORR_NORMED', 'cv2.TM_SQDIFF', 'cv2.TM_SQDIFF_NORMED']
result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
top_left = max_loc
h,w = templateGray.shape
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(image,top_left, bottom_right,(255,0,0),4)

cv2_imshow(image)
```

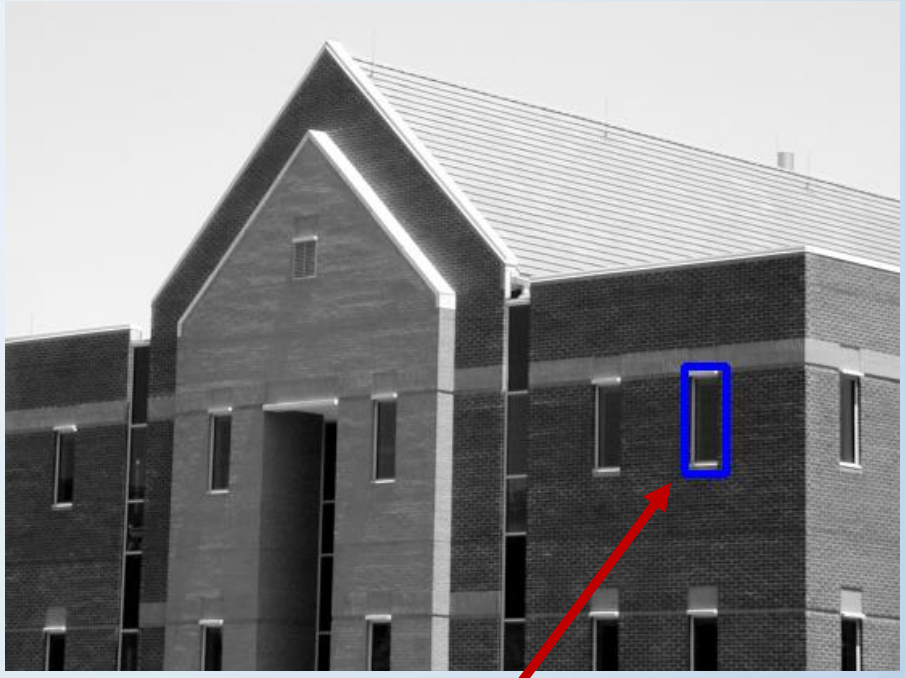
**Input Image:** $f(x, y)$



**Template:** $T(x, y)$

# Template Matching: Similarity Measures



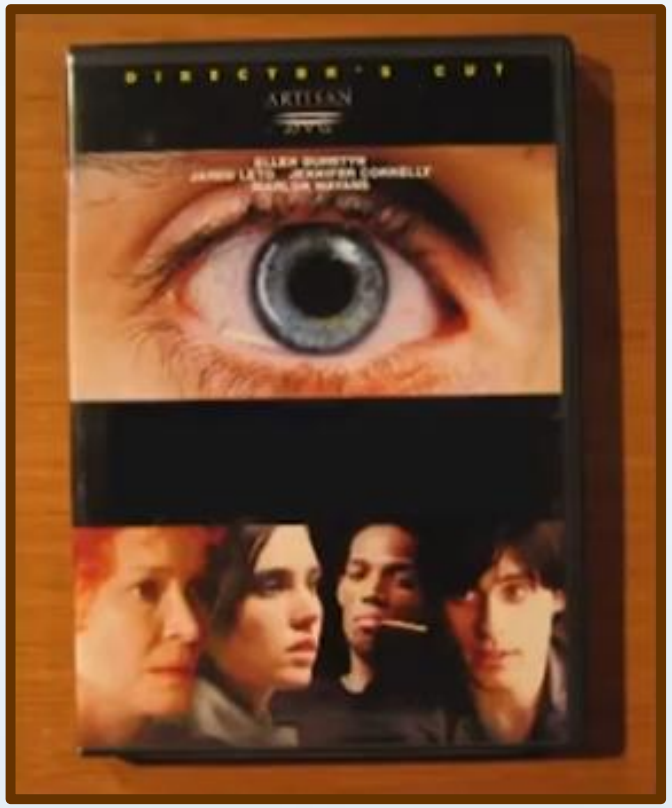



Detected Template

Two Brighter points are observed in the resultant heat-map. These points indicates the detected highest match.

These are two most likely positions of template matching

# Object Recognition: Interest Point or Feature Based




Template


Recognizing and Identifying Selected Template in the 2D rich Image

- ✓ Template matching is not an robust solution under this situation.
- ✓ Solution: Find and Match "Interest Point or Features"

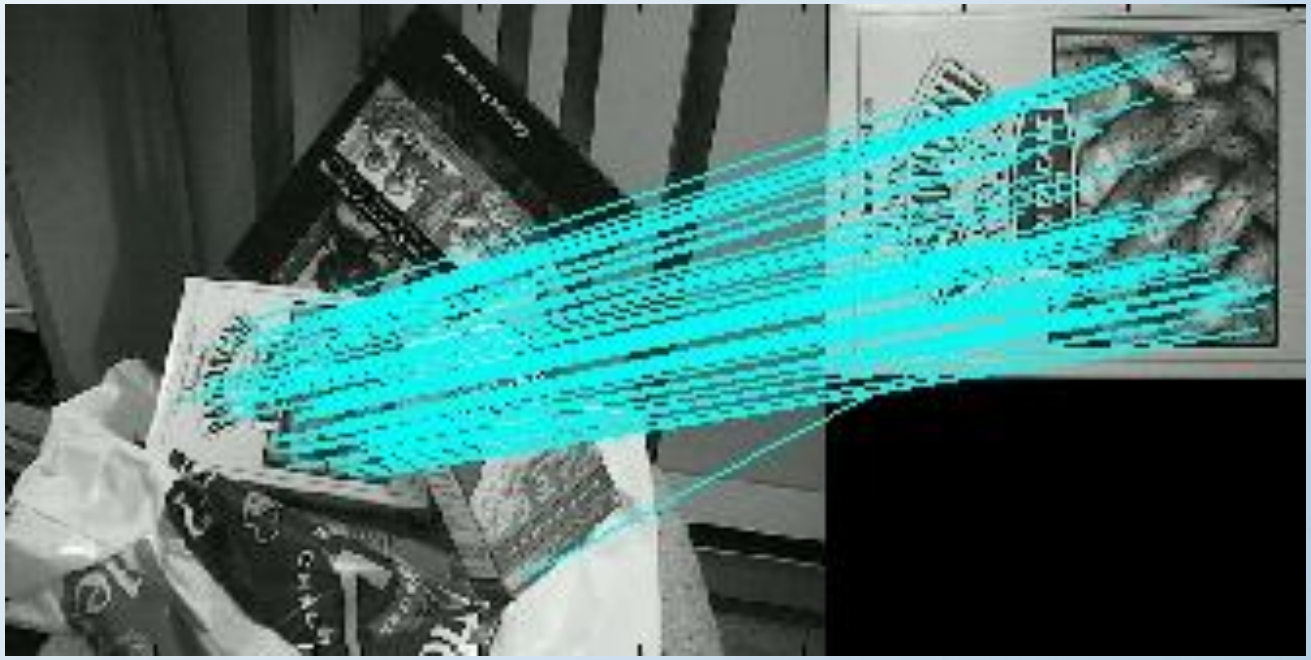
# SIFT as Key-point Detector and Descriptor

- ✓ **Scale Invariant Feature Transform**: Proposed by David G. Lowe (1999), "Object recognition from local scale-invariant features" Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157.
  - David G. Lowe: https://www.cs.ubc.ca/~lowe/keypoints/

## Motivation

Image Matching

- Correspondence Problem



Pic Courtesy: David G. Lowe: https://www.cs.ubc.ca/~lowe/keypoints/

# Feature Descriptor

- Image features are interest points located at various pixel positions of an image.

**Feature point located at $(x, y)$**

- A descriptor on other hand is an N-dimensional vector that describes the information around the detected interest point. Thus it illustrates the summery of the image information around the feature point.

**Feature point located at $(x, y)$**

**Descriptor: $\{f_1, f_2, \ldots\ldots\ldots, f_N\}$**

# Properties of feature descriptor

- ✓ Robustness and invariance to scale, orientation, rotation, illumination variation.
  - The scale invariance property is one of the most researched problem in computer vision.
  - Remember that Eigenvalue based detectors (Harris Corner Detector) are not robust to scale variance.

=

*Bikes* (image blur, structured scene)

# Properties of feature descriptor

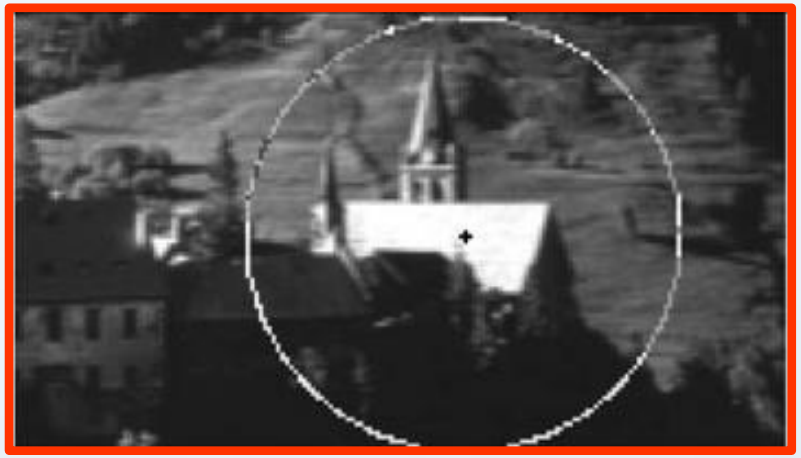
- ✓ Distinctive and Uniqueness.
  - The feature descriptors are a part of feature matching requires to effectively distinguish between two close-by features.

≠

*Bikes* (image blur, structured scene)

# SIFT as Key-point Detector and Descriptor

✓ **Scale Invariant Feature Transform**: Used for image alignment and 2D object recognition.



**Topics**:

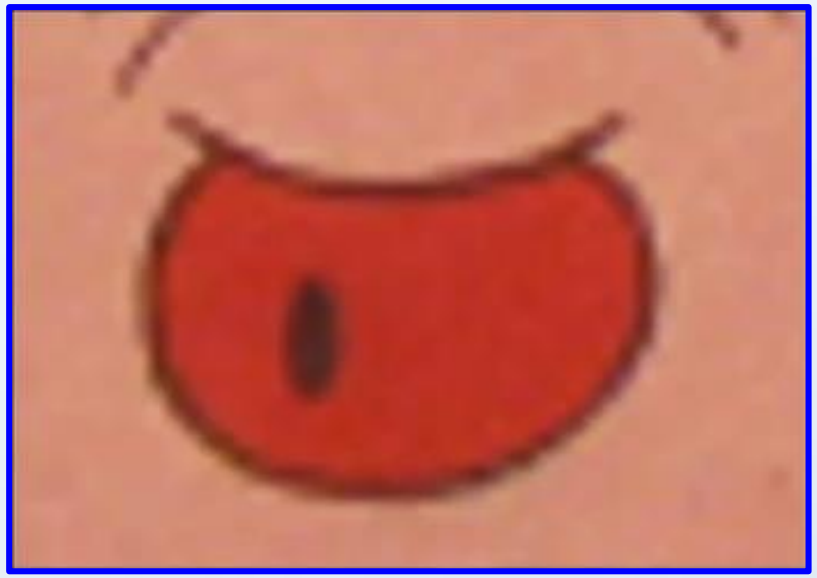
1. What is an Interest Point?
2. Detecting blobs (image patch with local appearance)
3. SIFT Detector
4. SIFT Descriptor

# What is an Interest Point?

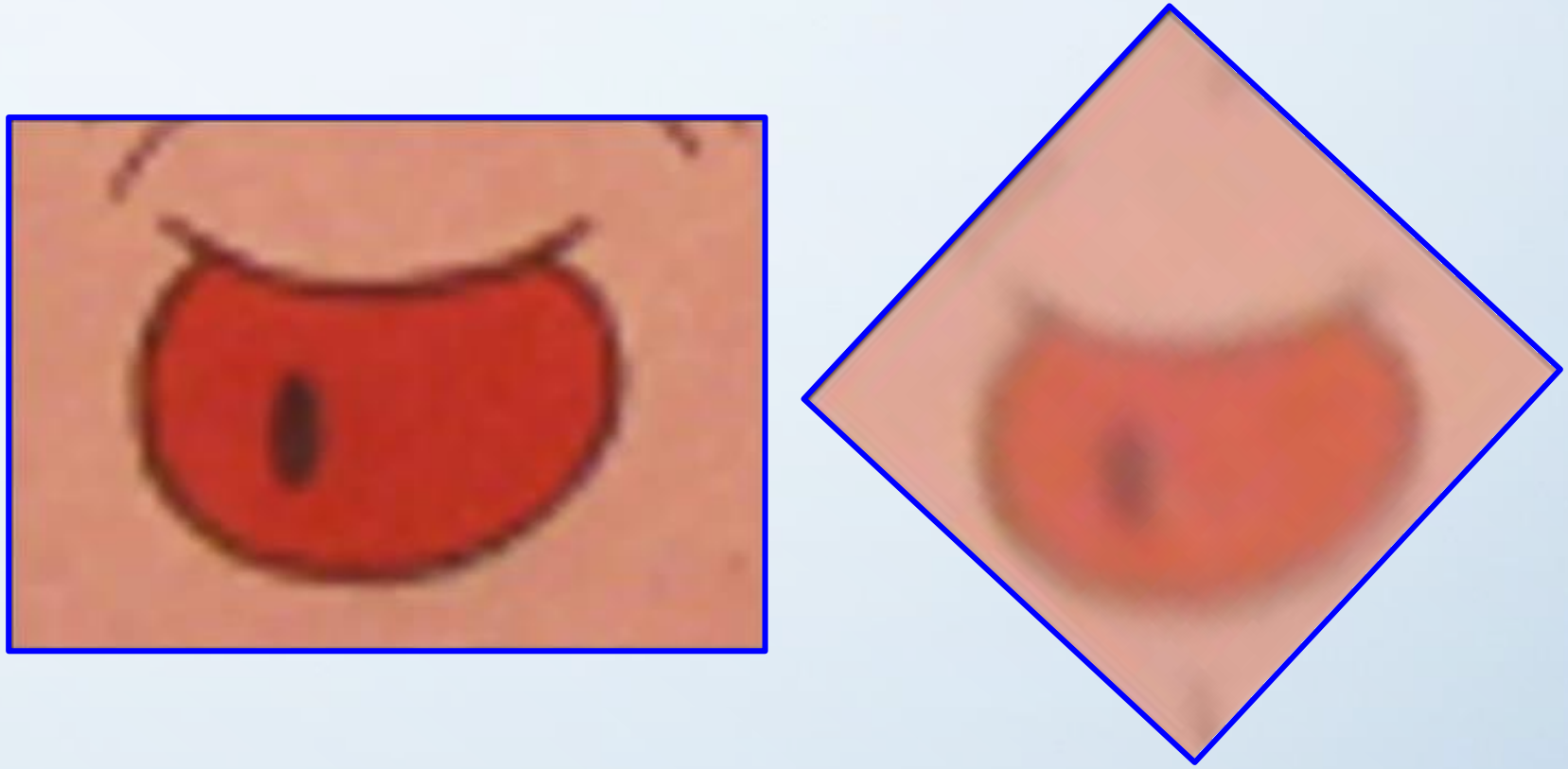
✓ **Raw images are hard to match**: Identifying unique features can aide in object recognition and matching.

**Different size, orientation, illumination and occlusion etc…**

# Removing Source Variation



# Removing Source Variation



Matching features can be easier if we can remove variations
like size and orientation

# Some patches are not interesting....!

(a)

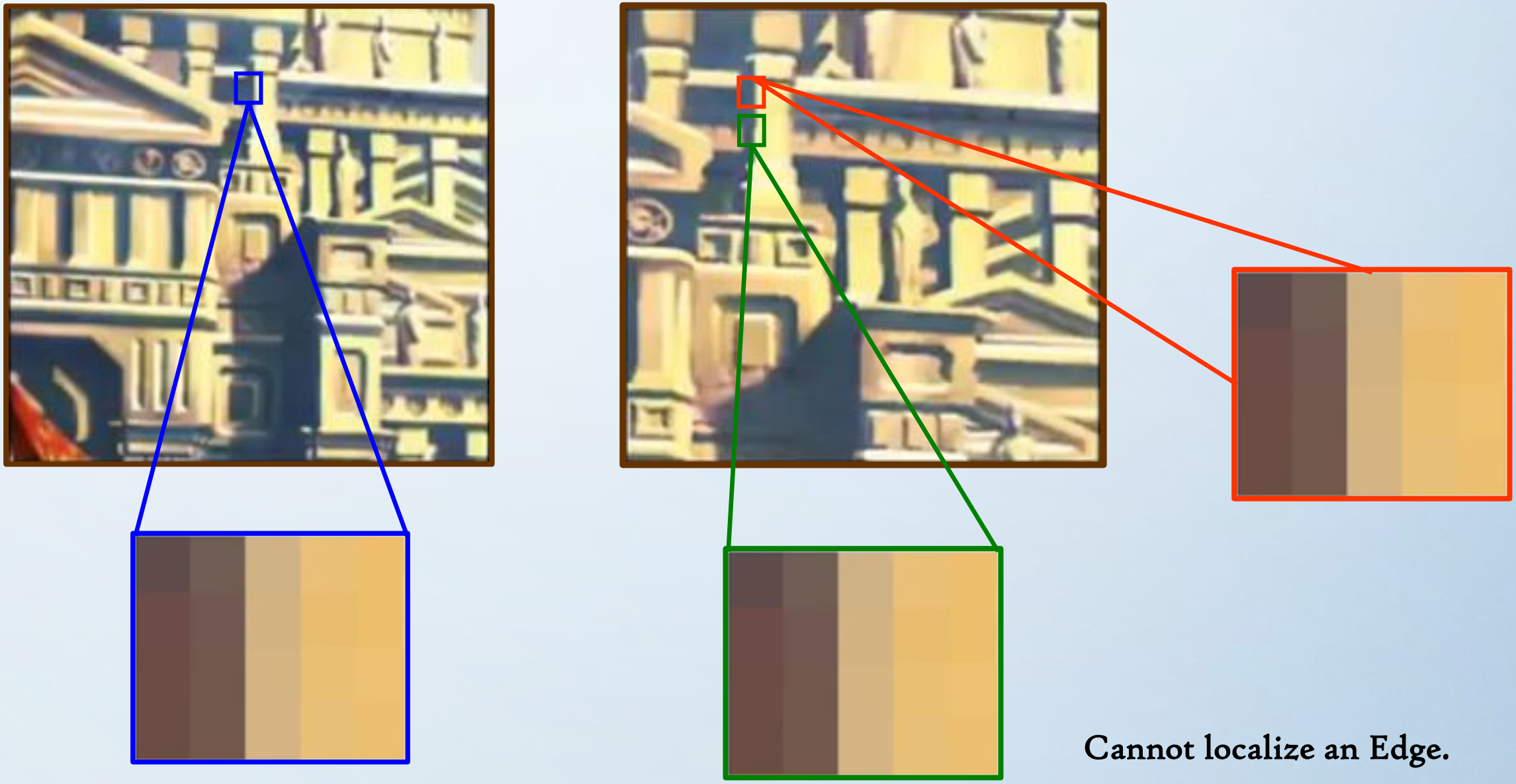
(b)

Image Pair. Few patches extracted from each image.
Notice some patches are very distinctive and unique compared to others.
The unique patches localized in an image are called interest point (features).

# What is an Interest Point?

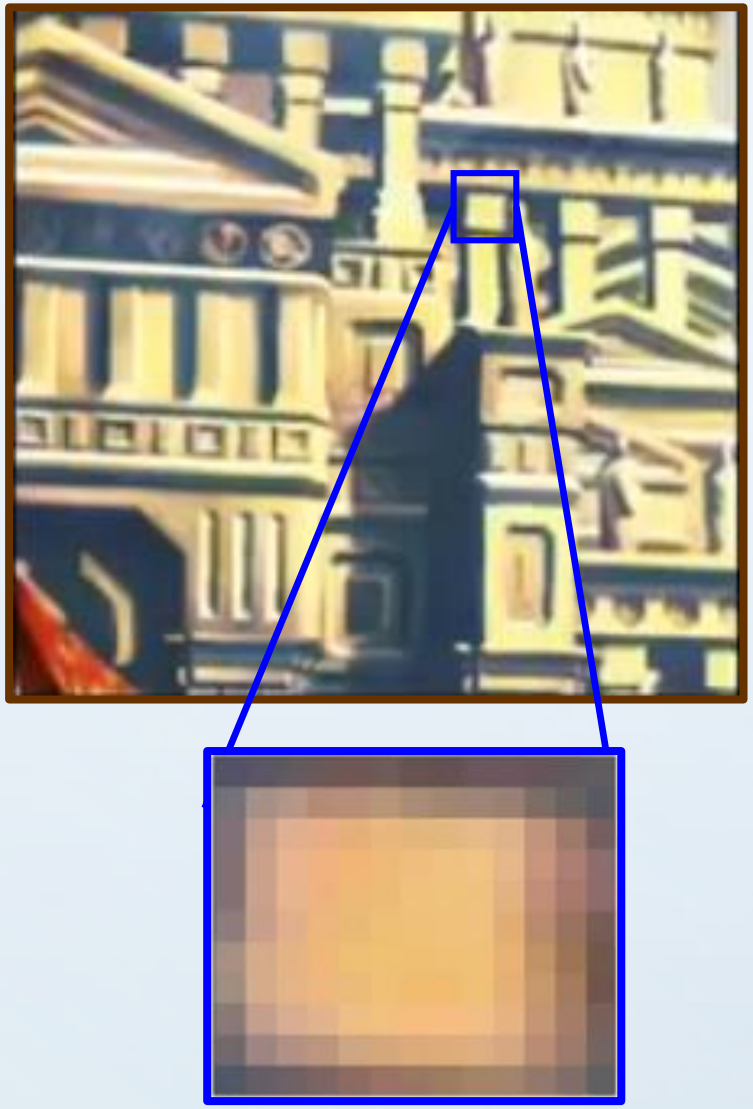
- ✓ Has **rich image content** (brightness variation, color variation etc…) with in the local window.
- ✓ Has well-defined **representation** (**signature**) for matching and comparison with other points in another image.
- ✓ Has a well defined **position** in the image.
- ✓ Should be invariant to **image rotation and scaling**.
- ✓ Should be invariant to **illumination/lighting** changes.
- ✓ Should **distinctive and unique** (different from the near neighborhood)

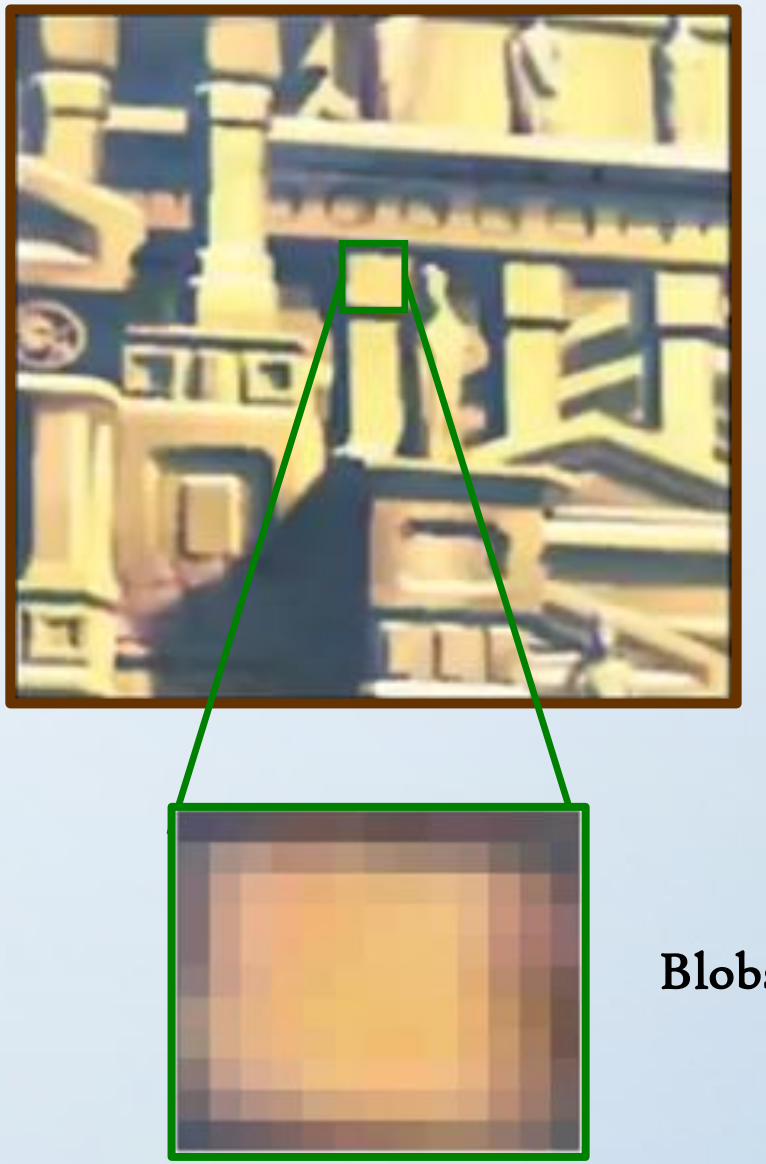
# Are Lines and/or Edges Interesting?



Cannot localize an Edge.

# Are Blobs Interesting?

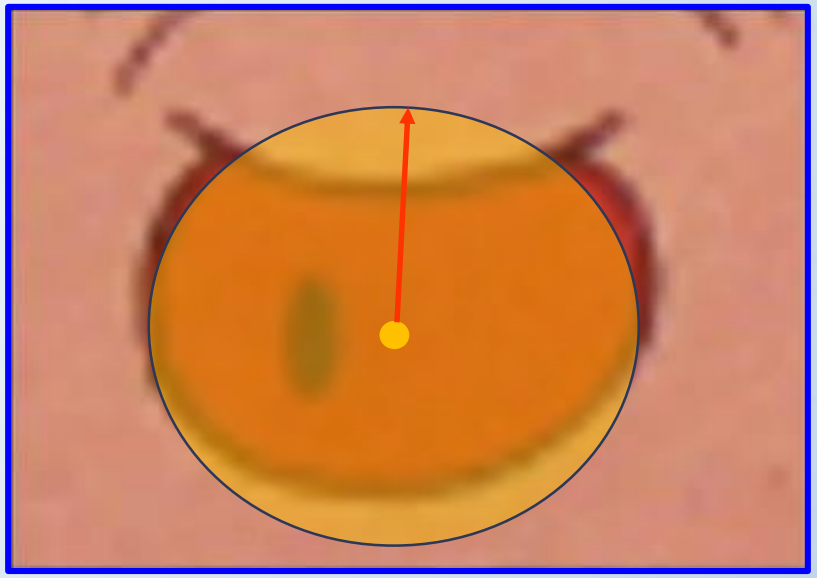




Blobs have Fixed position and definite size.

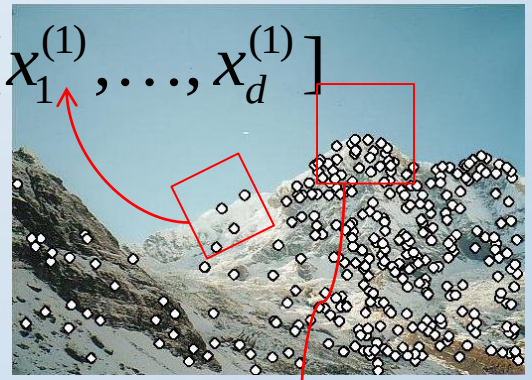
# Blobs as Interesting Pont

- ✓ For a blob like features 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.



**Description:** *Extracting the multi-dimensional feature vectors in the neighborhood of the interest point.*

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

# Detecting Blobs

- ✓ In computer vision, blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or color, compared to surrounding regions.

- ✓ Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be similar to each other.

- ✓ Blobs can be detected using:
  - ✓ Laplacian of Gaussian (LoG).
  - ✓ Derivative of Gaussian (DoG)

# Review: Derivative of Gaussian (DoG) ($\nabla(\eta_\sigma * f)$)

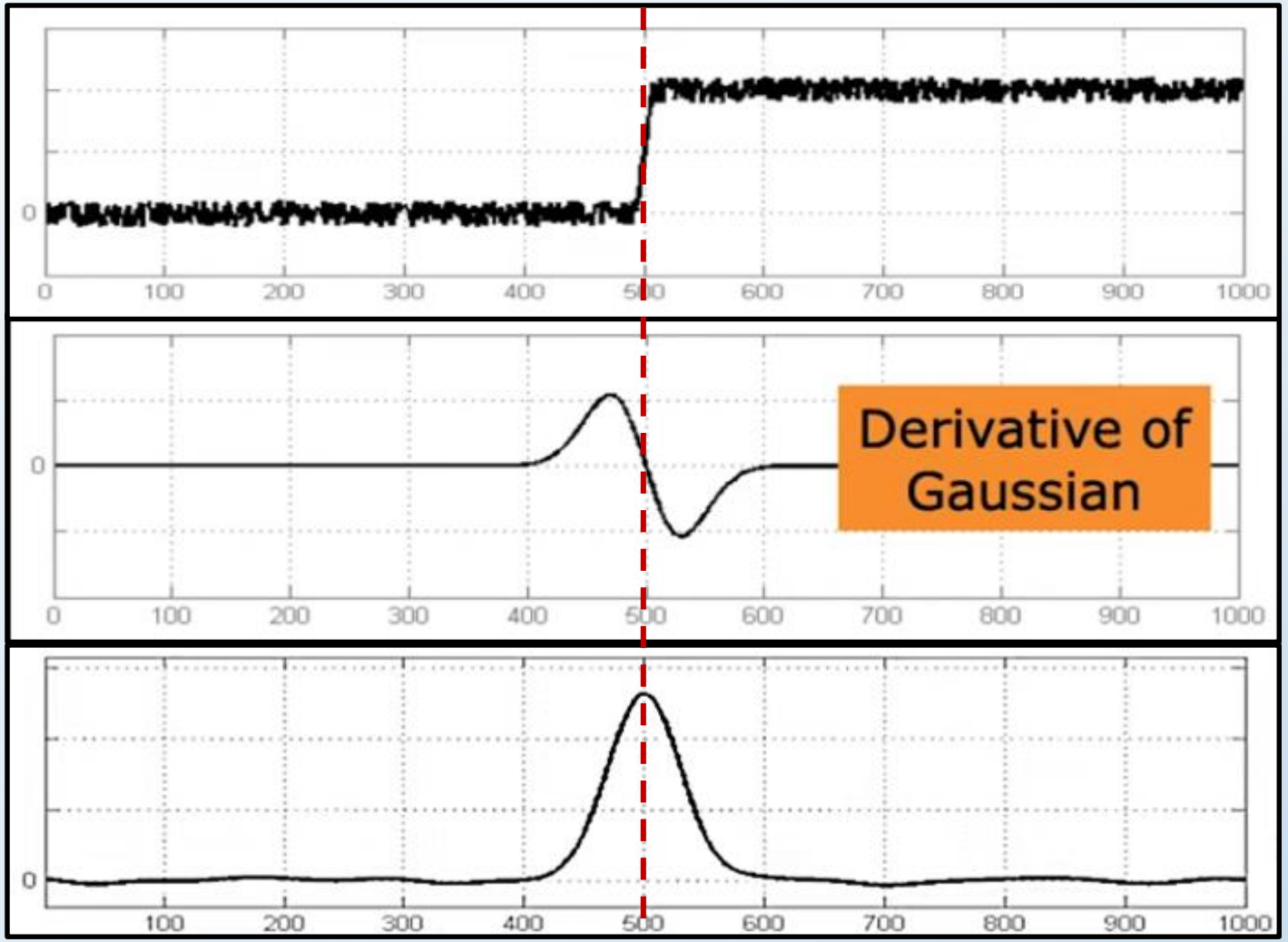
$\nabla(\eta_\sigma * f) = \nabla(\eta_\sigma) * f$ : This operation save us one operation

Note: The derivative of Gaussian $\nabla(\eta_\sigma)$ is a constant and can be calculated before hand.

$f$

$\nabla(\eta_\sigma)$
(DoG Operator)

$\nabla(\eta_\sigma) * f$
(Maxima as Edge)




Extremum of derivative of Gaussian (DoG) denotes an edge.

# Review: Laplacian of Gaussian (LoG) ($\nabla^2(\eta_\sigma * f)$)

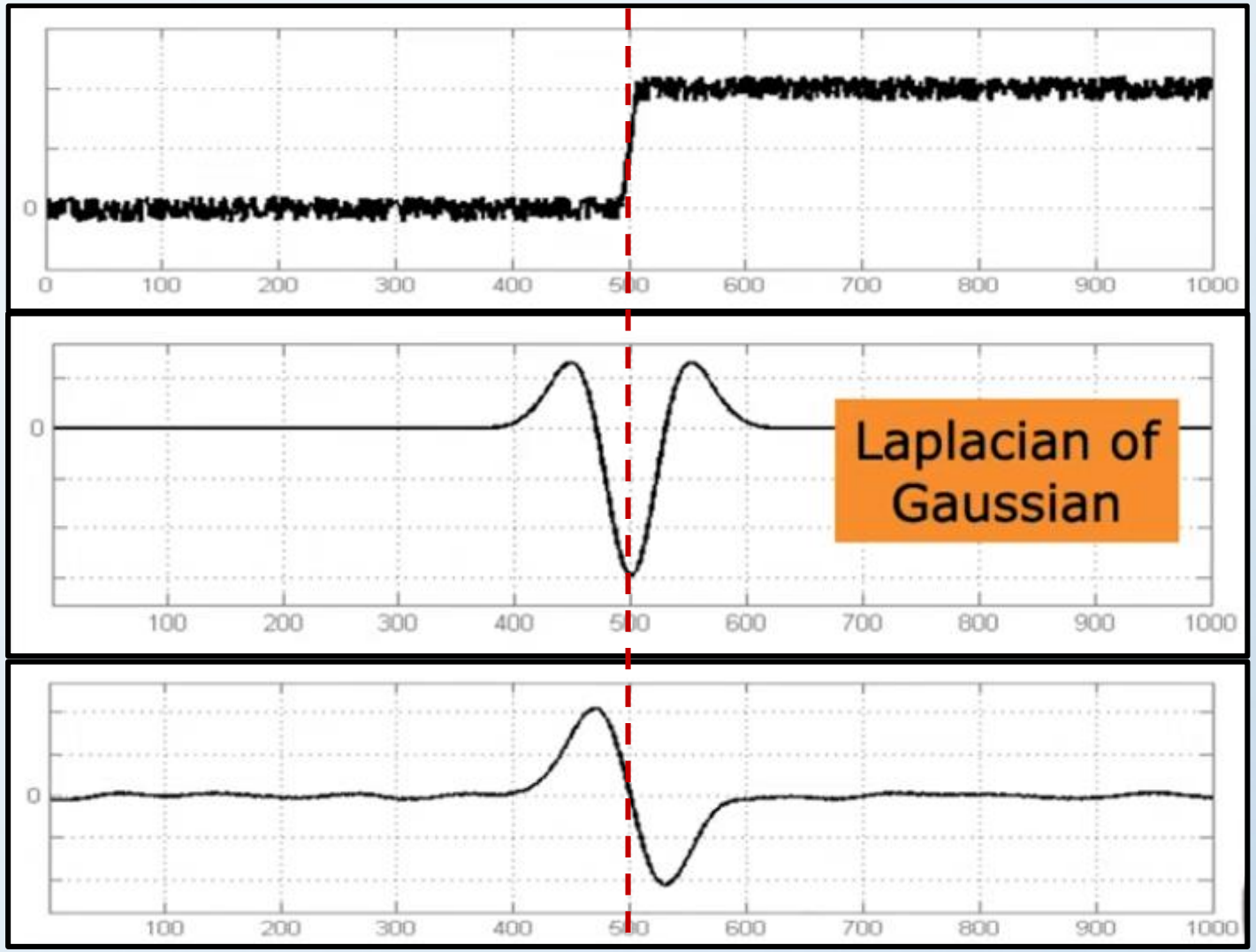
$\nabla^2(\eta_\sigma * f) = \nabla^2(\eta_\sigma) * f$ : This operation save us one operation

Note: The derivative of Laplacian $\nabla^2(\eta_\sigma)$ is a constant and can be calculated before hand.

$f$

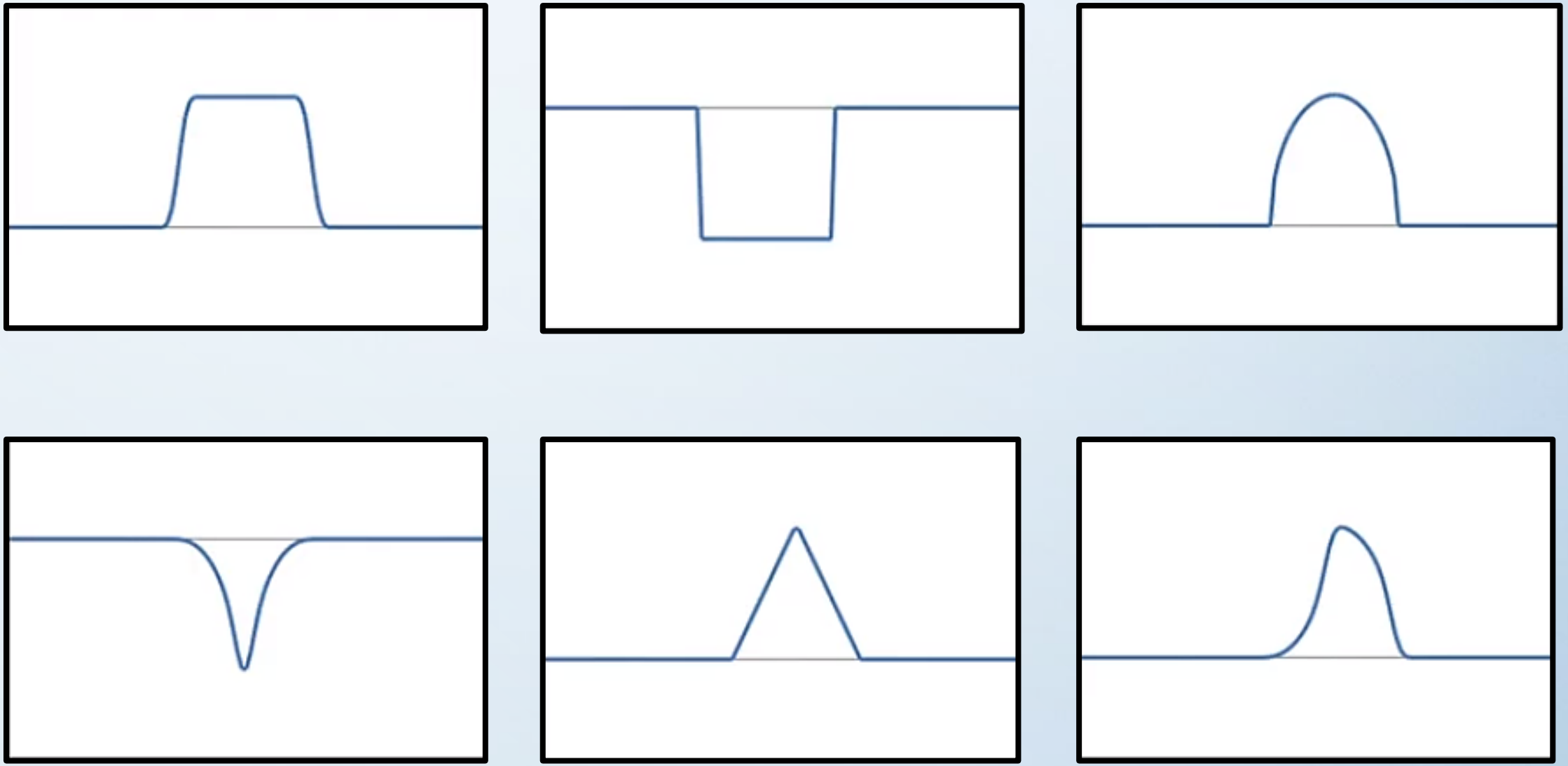
$\nabla^2(\eta_\sigma)$ (DoG Operator)

$\nabla^2(\eta_\sigma) * f$ (Zero-Crossing as Edge)




Zero crossing of 2nd derivative of Gaussian denotes an edge.

# 1D Blobs



Example of 1D blob like structures.

# 1D Blobs & $2^{nd}$ Derivative of Gaussian

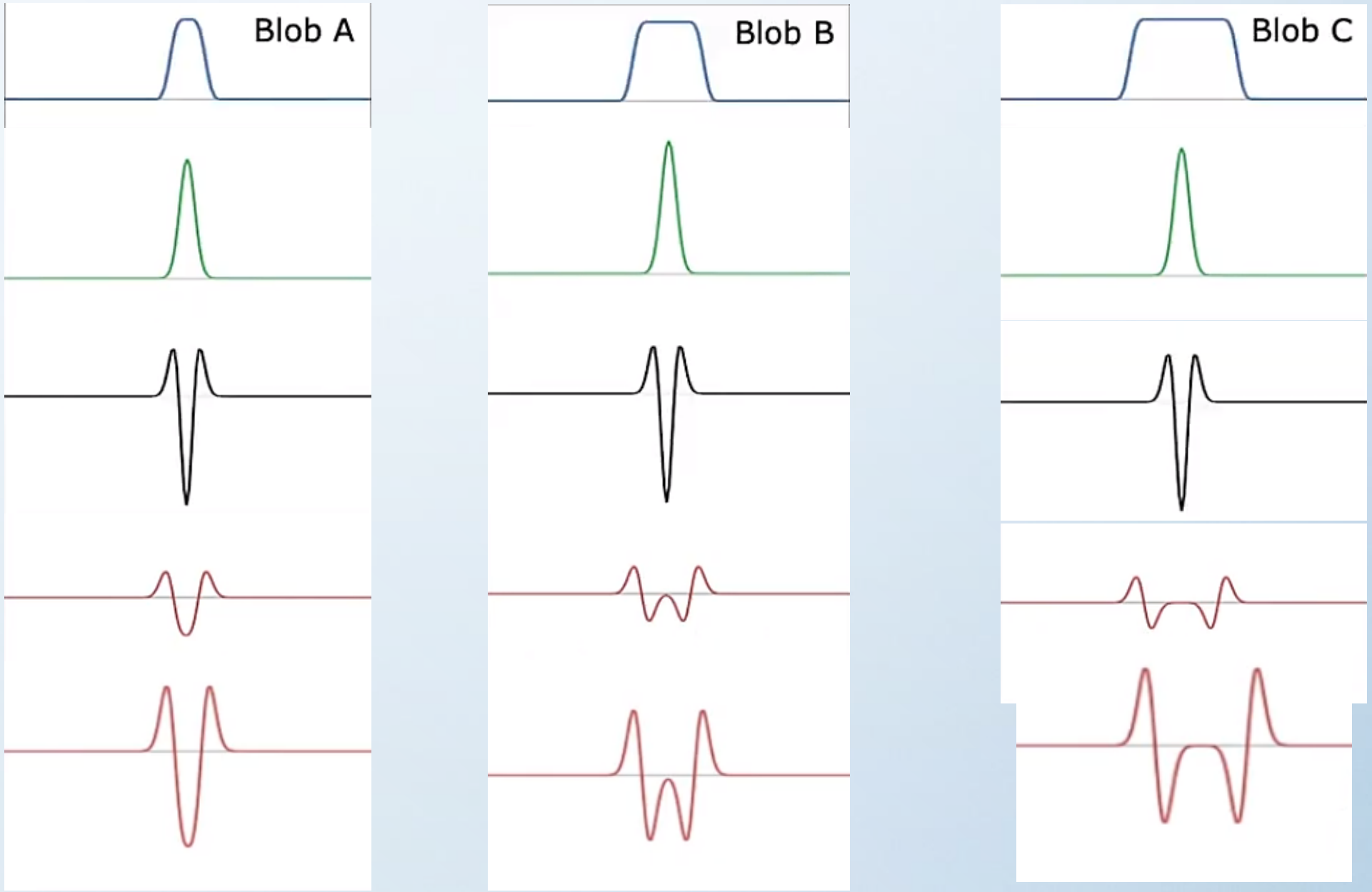
$f(x)$

$\eta_\sigma$

$\frac{\partial^2 \eta_\sigma}{\partial x^2}$

$\frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)$

$\sigma^2 \frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)$



# 1D Blobs & $2^{nd}$ Derivative of Gaussian (Different $\sigma$)

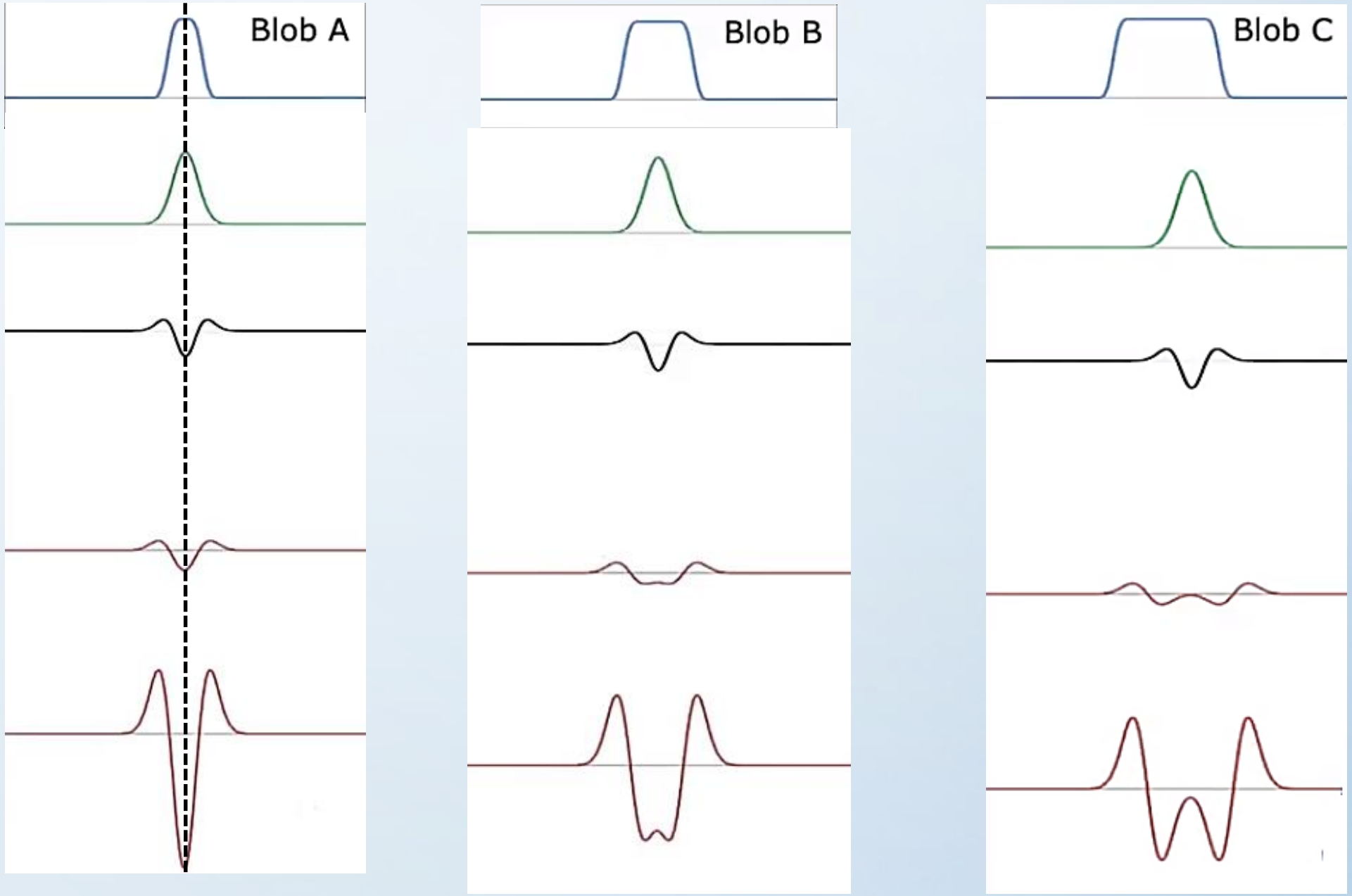
$f(x)$

$\eta_\sigma$

$\frac{\partial^2 \eta_\sigma}{\partial x^2}$

$\frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)$

$\sigma^2 \frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)$



# 1D Blobs & $2^{nd}$ Derivative of Gaussian (Different $\sigma$)

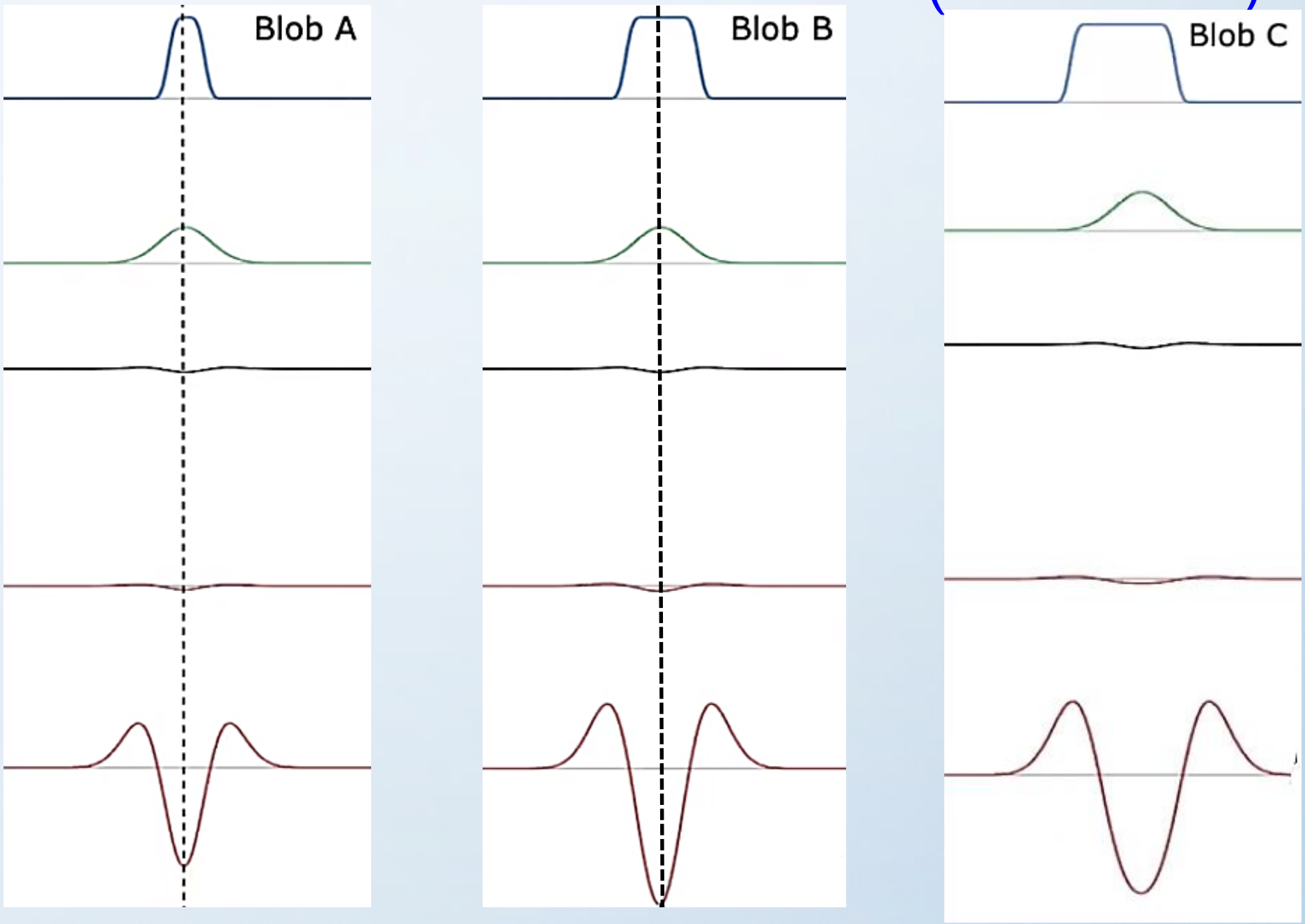
$f(x)$

$\eta_\sigma$

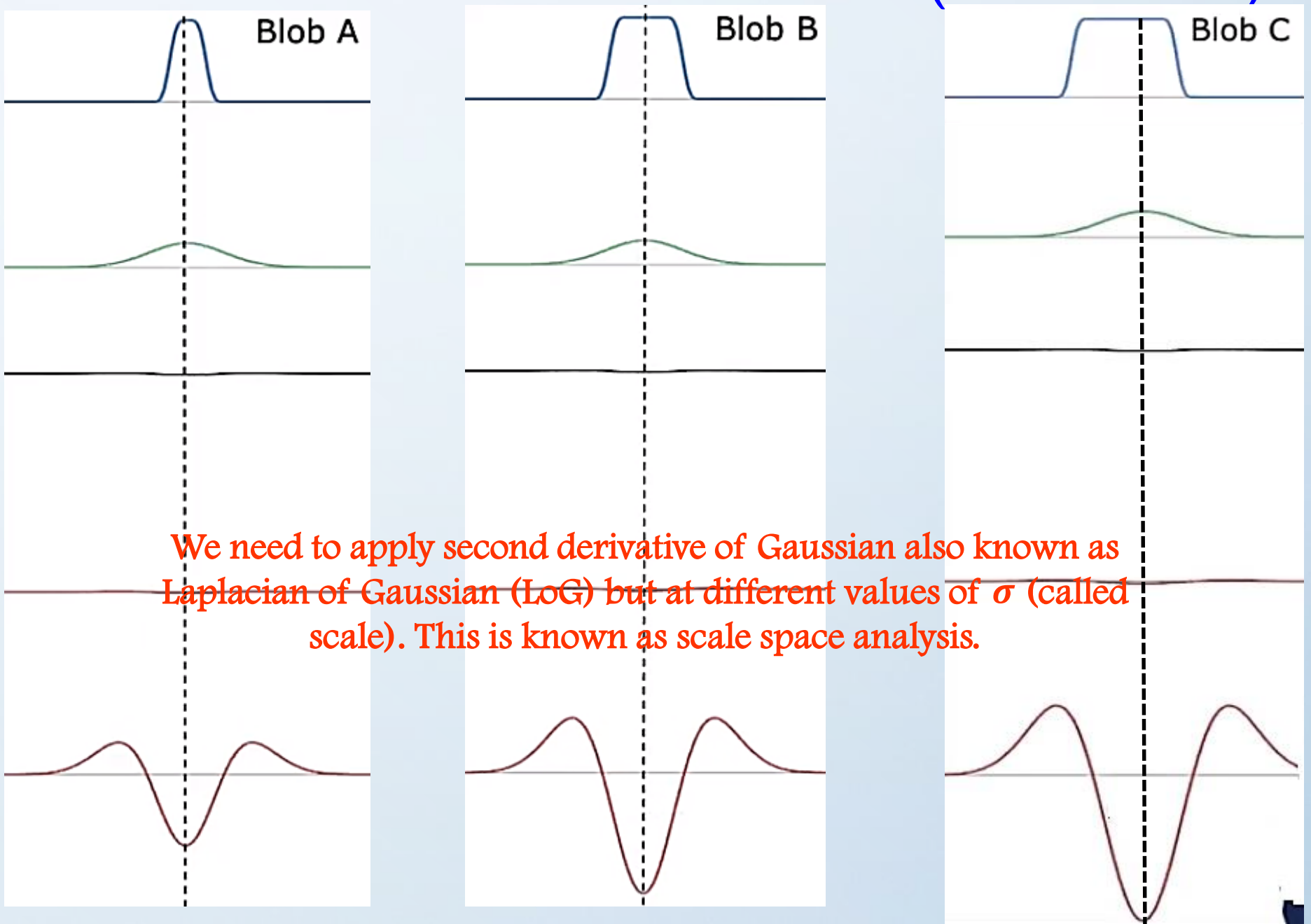
$\frac{\partial^2 \eta_\sigma}{\partial x^2}$

$\frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)$

$\sigma^2 \frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)$

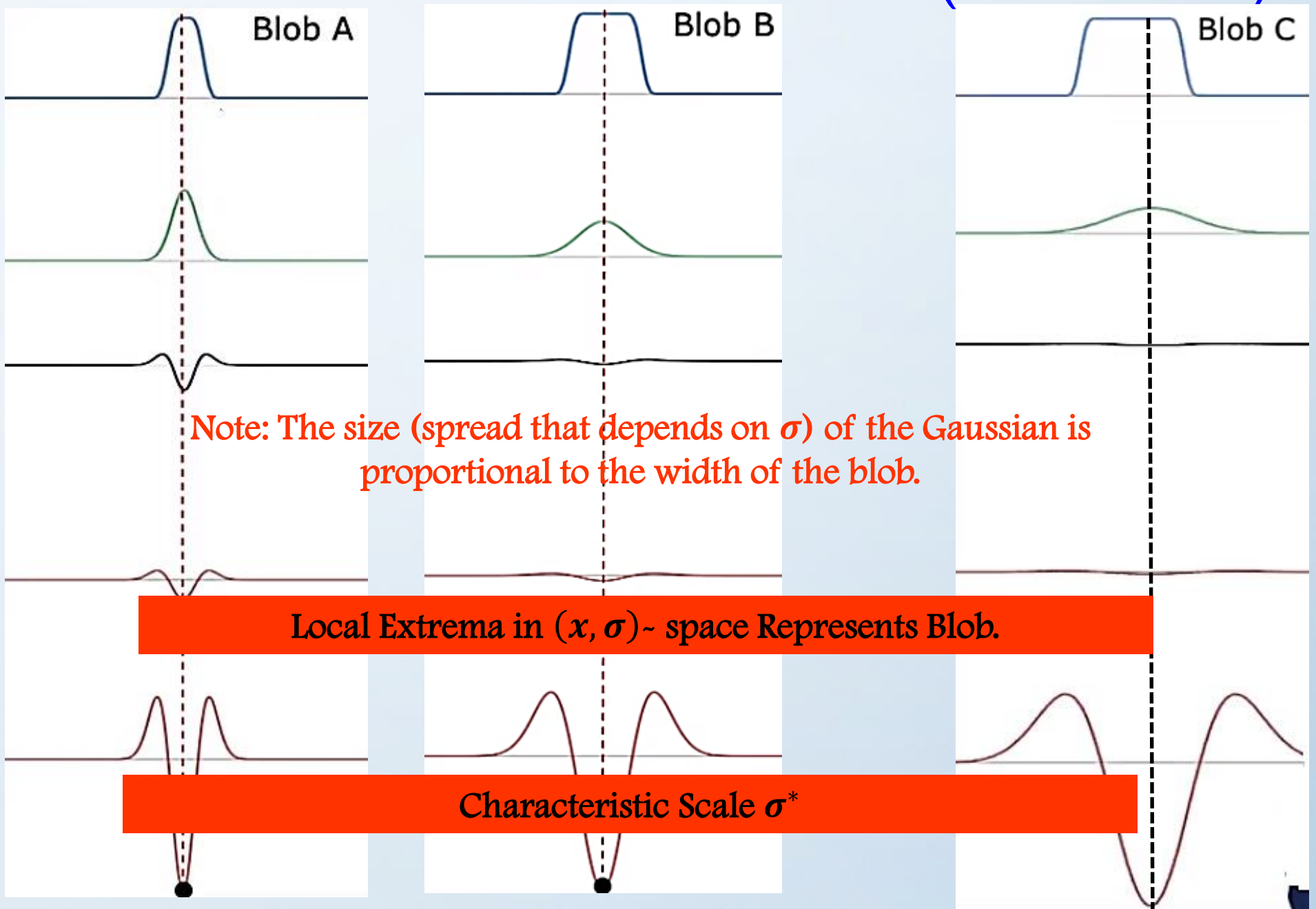


# 1D Blobs & $2^{nd}$ Derivative of Gaussian (Different $\sigma$)

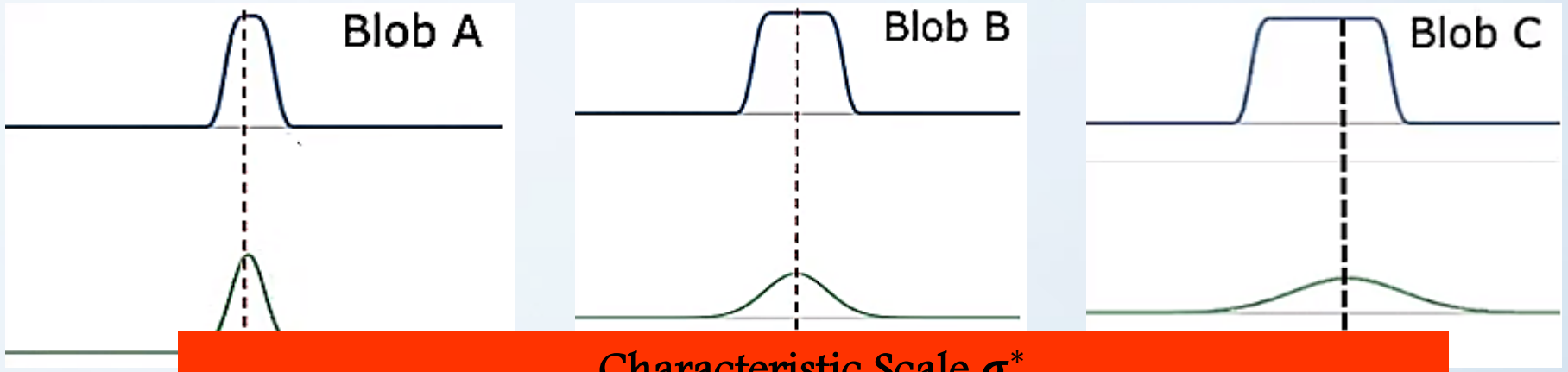



We need to apply second derivative of Gaussian also known as Laplacian of Gaussian (LoG) but at different values of $\sigma$ (called scale). This is known as scale space analysis.

# 1D Blobs & $2^{nd}$ Derivative of Gaussian (Different $\sigma$)



# Scale Space Analysis: Characteristic Scale and Blob Size




$\sigma_A^* = \sigma_1$ $\sigma_B^* = \sigma_2$ $\sigma_C^* = \sigma_3$

✓ Characteristic scale: The $\sigma$ at which $\sigma$–normalized second derivative attains its extremum value.

*Characteristic Scale* $\propto$ *Size of the Blob*

$$\frac{Size\ of\ Blob\ A}{Size\ of\ Blob\ B} = \frac{\sigma_A^*}{\sigma_B^*}$$

$$\frac{Size\ of\ Blob\ B}{Size\ of\ Blob\ C} = \frac{\sigma_B^*}{\sigma_C^*}$$

# 1D Blob Detection: Summery

✓ Given: 1D Signal $f(x)$.

✓ Compute: $\boldsymbol{\sigma^2 \frac{\partial^2 \eta_\sigma}{\partial x^2} * f(x)}$ at many scales $(\sigma_0, \sigma_1, \sigma_2 \dots \dots \sigma_k)$

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

$x^*$: Blob Position

$\sigma^*$: Characteristic Scale (Blob Size)

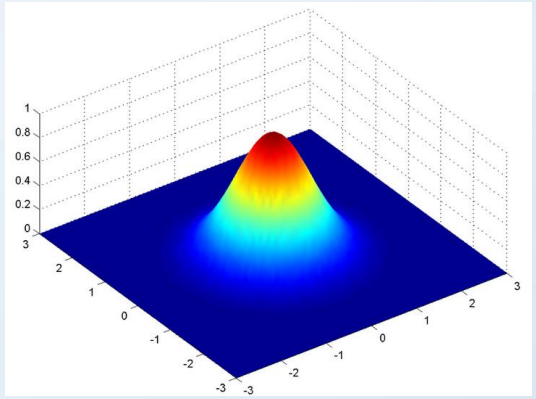
# 2D Blob Detector

✓ **Normalized Laplacian of Gaussian** (**NLoG**) is used as the equivalent for 2D blob detection.

✓ Laplacian:

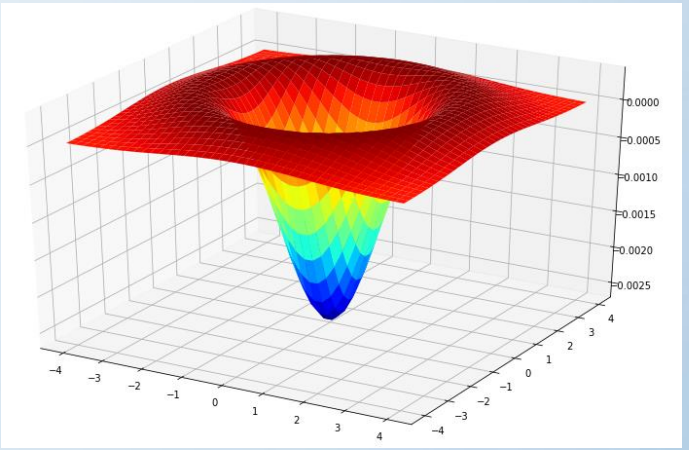
$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

Gaussian



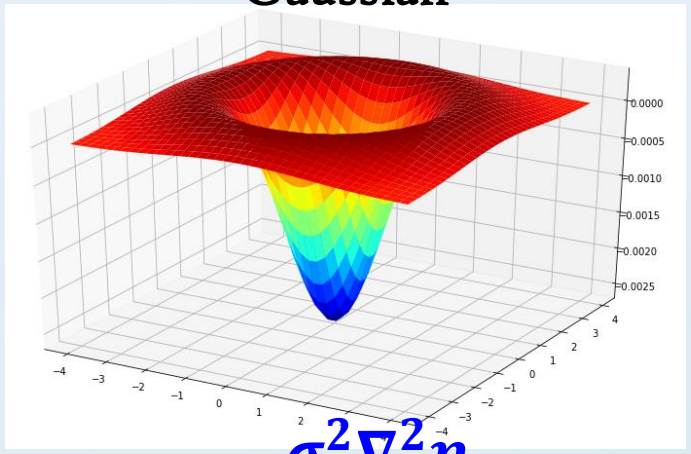
$\eta_\sigma$

Laplacian of Gaussian



$\nabla^2 \eta_\sigma$

Normalized Laplacian of Gaussian



$\sigma^2 \nabla^2 \eta_\sigma$

*Location of blob given by local extrema after applying normalized Laplacian of Gaussian at many scales.*

# Scale Space Analysis: 2D Blob Detector


$S(x, y\ \sigma_0)$


$S(x, y\ \sigma_1)$


$S(x, y\ \sigma_2)$


$S(x, y\ \sigma_3)$

...

Increase $\sigma$, higher scale lower Resolution

✓ **Scale-Space:** Stack created by filtering an image with Gaussians of different $\sigma$. As $\sigma$ changes different resolution images gets created.

$$S(x, y, \sigma) = \eta(x, y, \sigma) * I(x, y)$$

# Scale Space Analysis: 2D Blob Detector


$S(x, y\ \sigma_0)$


$S(x, y\ \sigma_1)$


$S(x, y\ \sigma_2)$


$S(x, y\ \sigma_3)$

...

Increase $\sigma$, higher scale lower Resolution

✓ **Selecting $\sigma$ :** The stacked images are created by considering the following $\sigma$ relationship.

$$\sigma_K = \sigma_0 s^k \qquad k = 0, 1, 2, 3 \ldots$$

$\boldsymbol{s}$: constant multiplier.

$\boldsymbol{\sigma_0}$: Initial sigma value considered.

# Scale Space Analysis: 2D Blob Detector

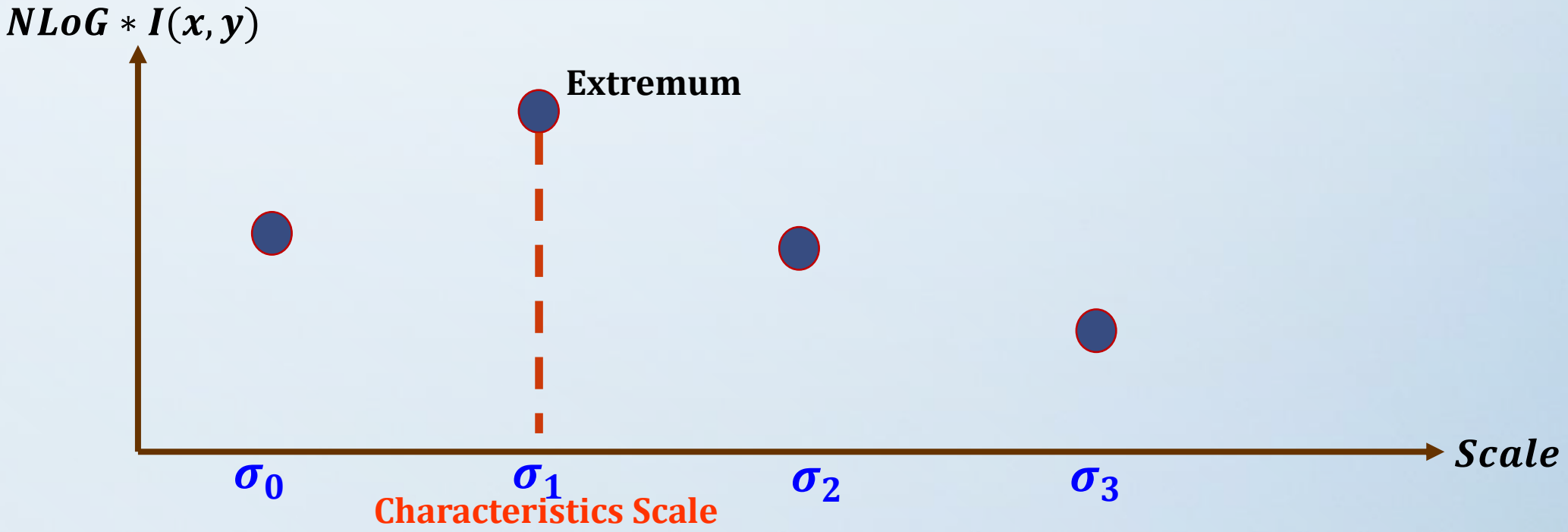
$S(x, y, \sigma_0)$

$S(x, y, \sigma_1)$

$S(x, y, \sigma_2)$

$S(x, y, \sigma_3)$

...

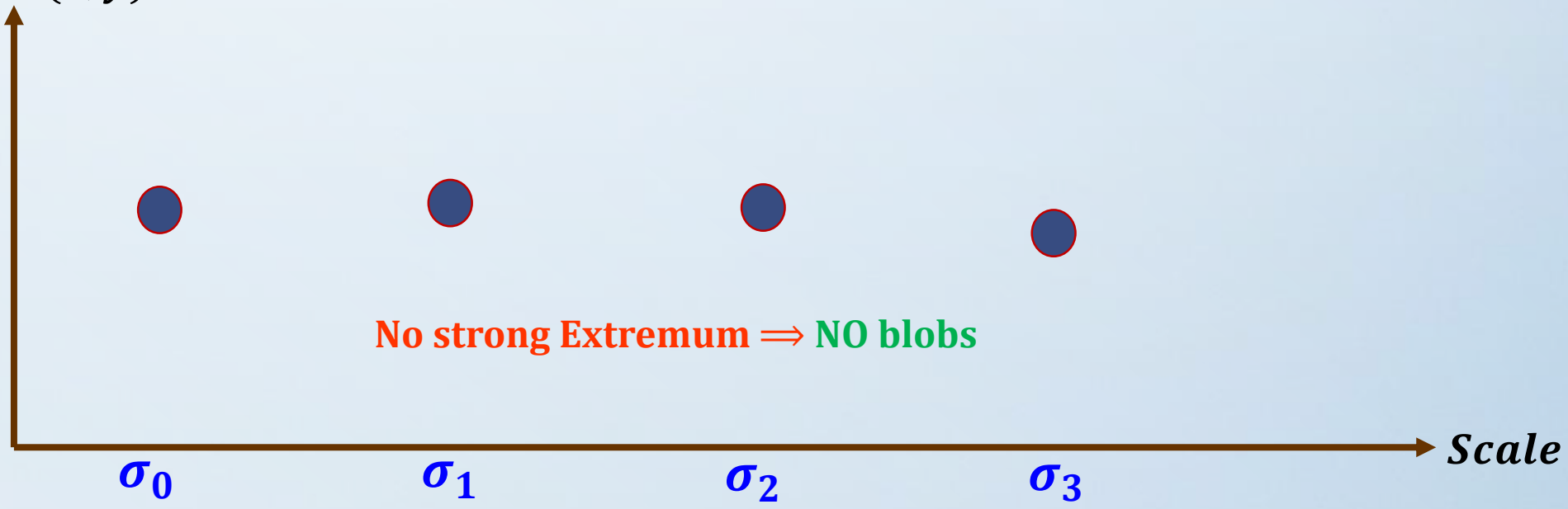


# Scale Space Analysis: 2D Blob Detector




$S(x, y, \sigma_0)$ $S(x, y, \sigma_1)$ $S(x, y, \sigma_2)$ $S(x, y, \sigma_3)$ ...

$NLoG * I(x, y)$




No strong Extremum $\Longrightarrow$ NO blobs

# 2D Blob Detection: Summery

✓ Given: 2D Image $I(x, y)$.

✓ Convolve the image using NLoG at many scale $\sigma$

✓ Find: $$(x^*, y^*, \sigma^*) = \arg\max_{(x,y,\sigma)} \left| \boldsymbol{\sigma^2 \nabla^2 \eta_\sigma} * \boldsymbol{f(x)} \right|$$

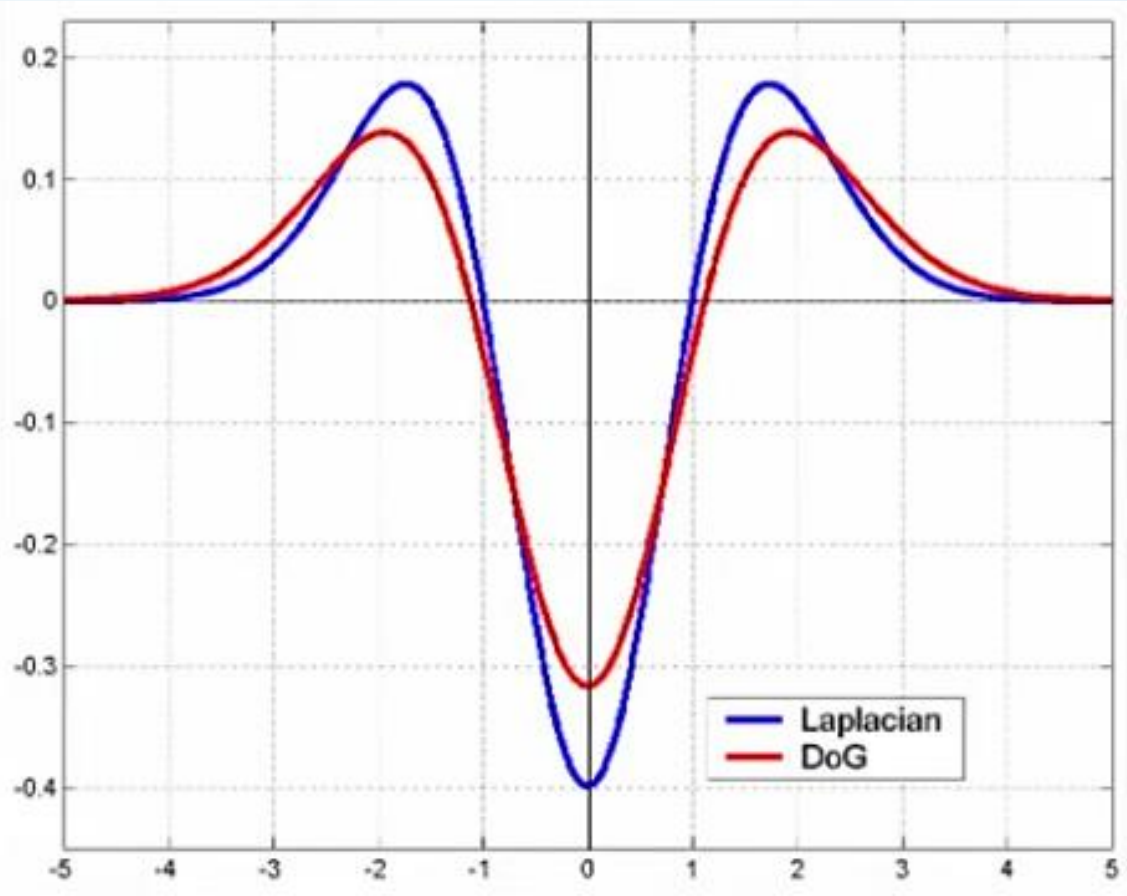
$(x^*, y^*)$: Blob Position

$\sigma^*$: Characteristic Scale (Blob Size)

# SIFT Detector: Principle and Formulation

✓ Difference of Gaussian (DoG) = $(\eta_{s\sigma} - \eta_{\sigma}) \approx (s-1)\sigma^2\nabla^2\eta_{\sigma}$

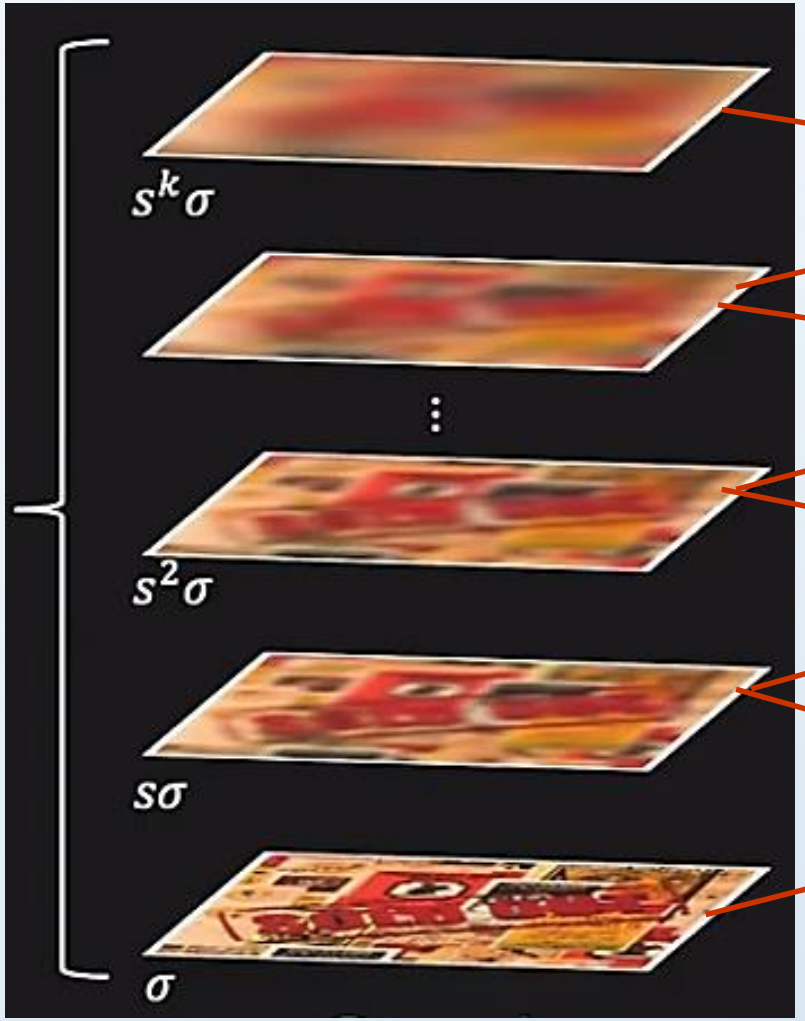
NLoG



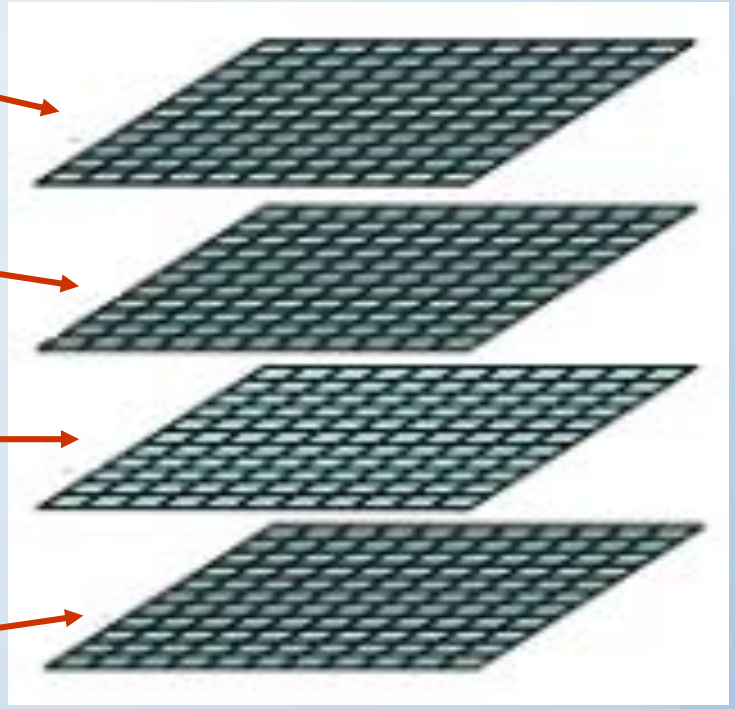

$$DoG \approx (s-1)NLoG$$

# SIFT Detector: Extracting SIFT Interest Point


Image
$I(x, y)$

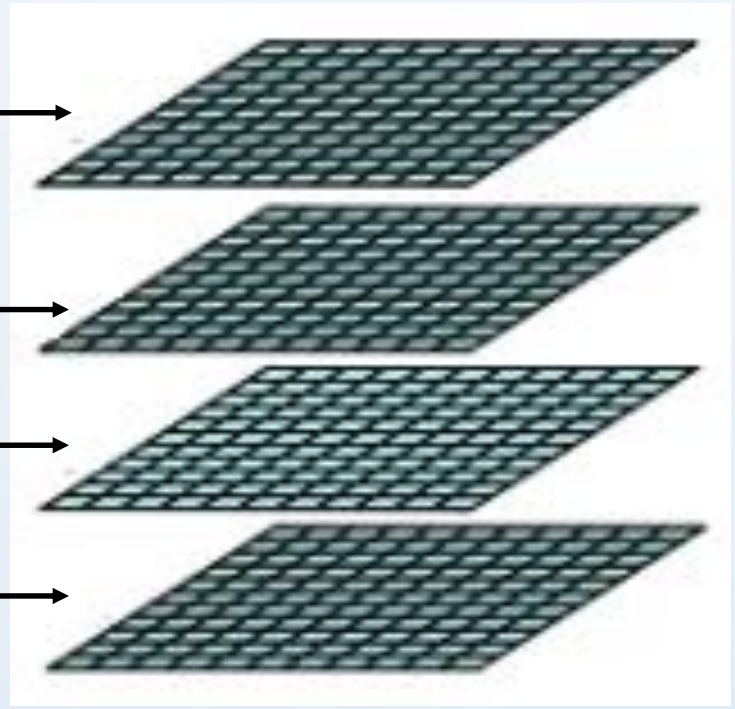



Gaussian Scale Space
$S(x, y, \sigma)$

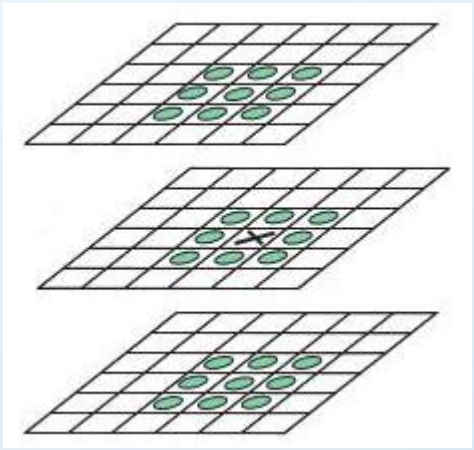


Difference of Gaussian (DoG)
$\approx (s-1)\sigma^2\nabla^2 S(x, y, \sigma)$

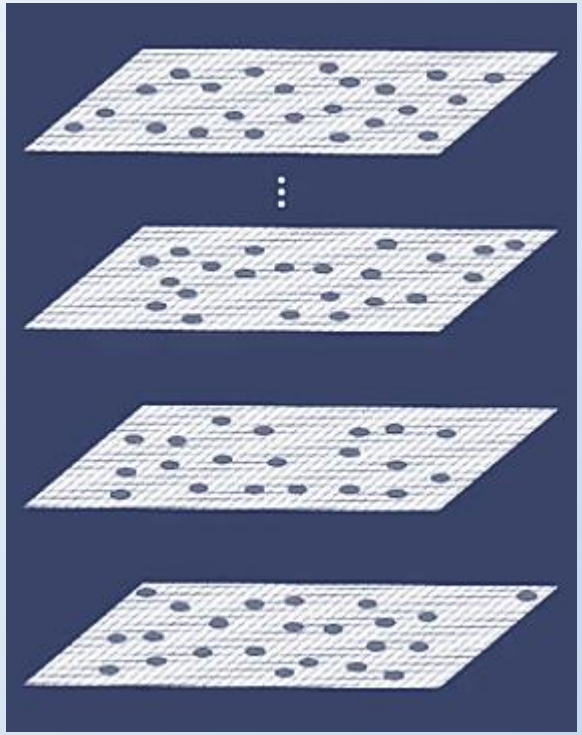
# SIFT Detector: Extracting SIFT Interest Point



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

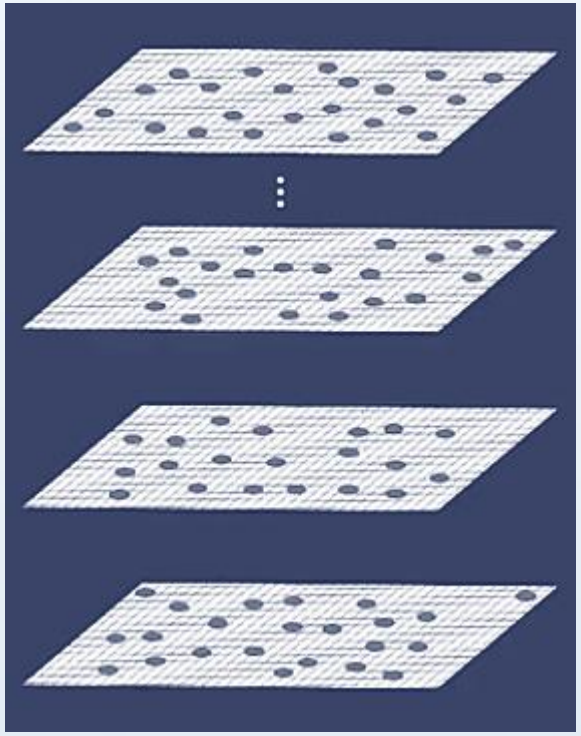


Find Extremum in every
$3 \times 3 \times 3$ grid

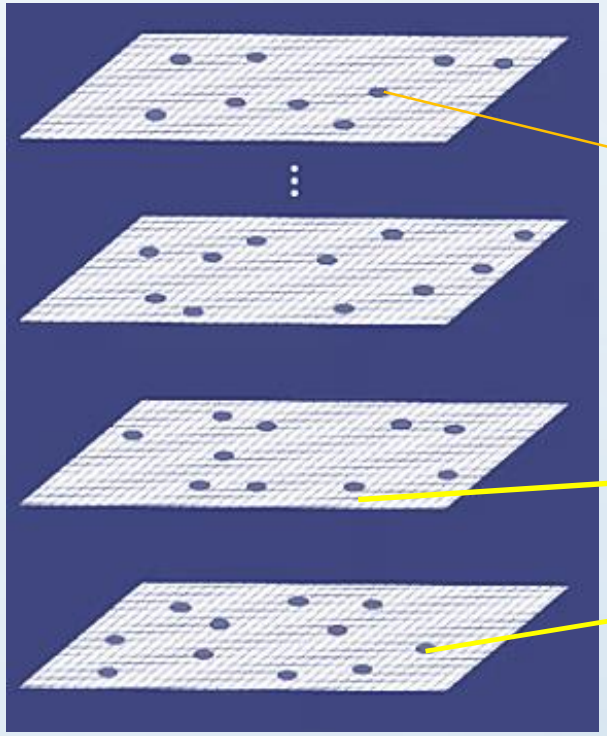


Interest Point
Candidates

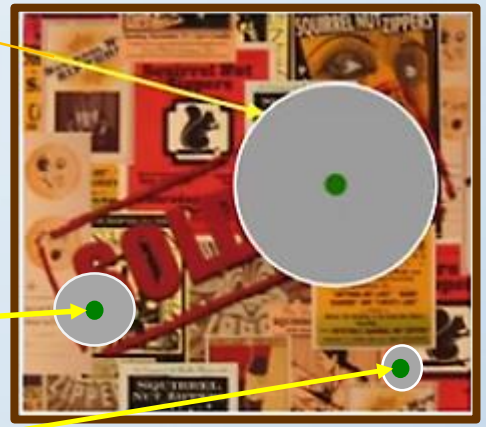
# SIFT Detector: Extracting SIFT Interest Point



Interest Point Candidates

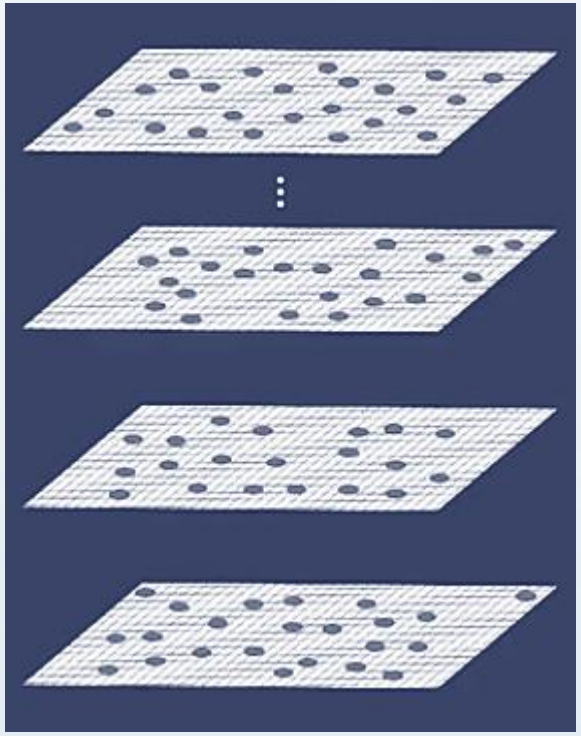


SIFT Interest Points (After removing Weak Interest Points)

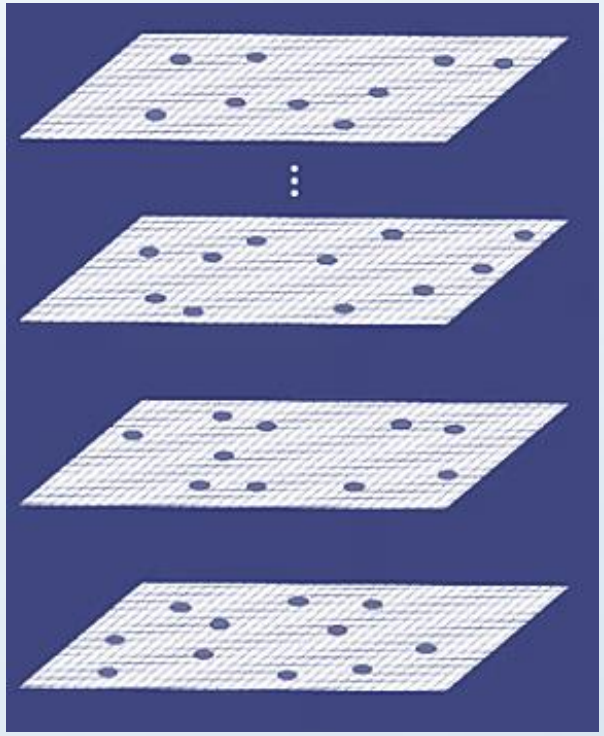


SIFT Interest Points (Visualization)

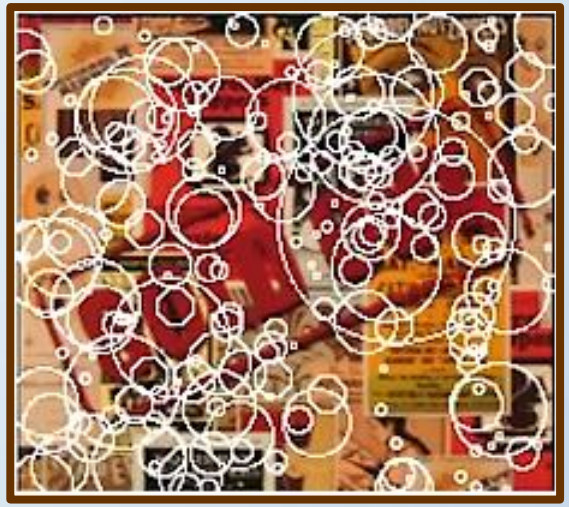
# SIFT Detector: Extracting SIFT Interest Point



Interest Point Candidates

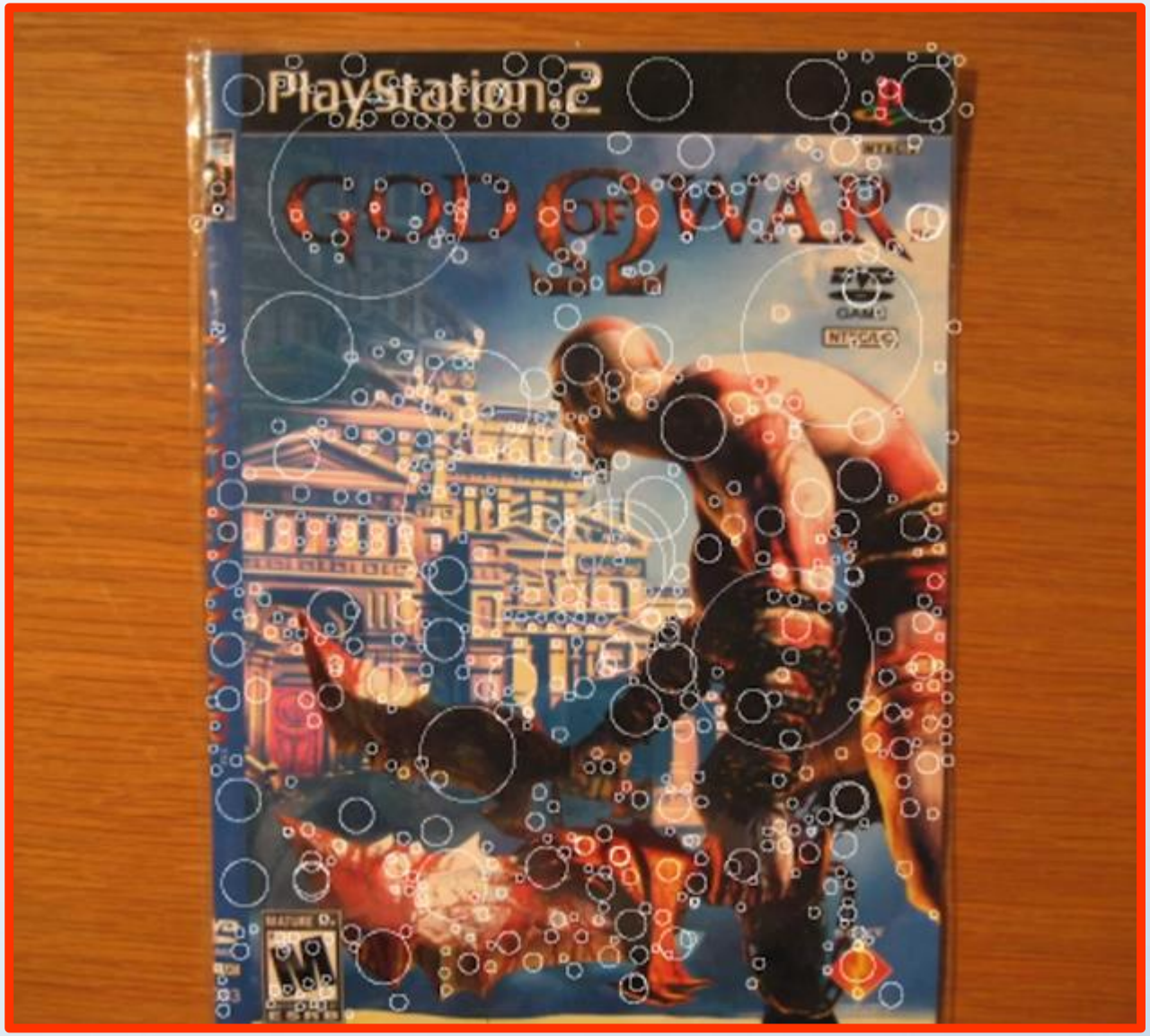


SIFT Interest Points (After removing Weak Interest Points)



SIFT Interest Points (Visualization)

# SIFT Detector: Examples

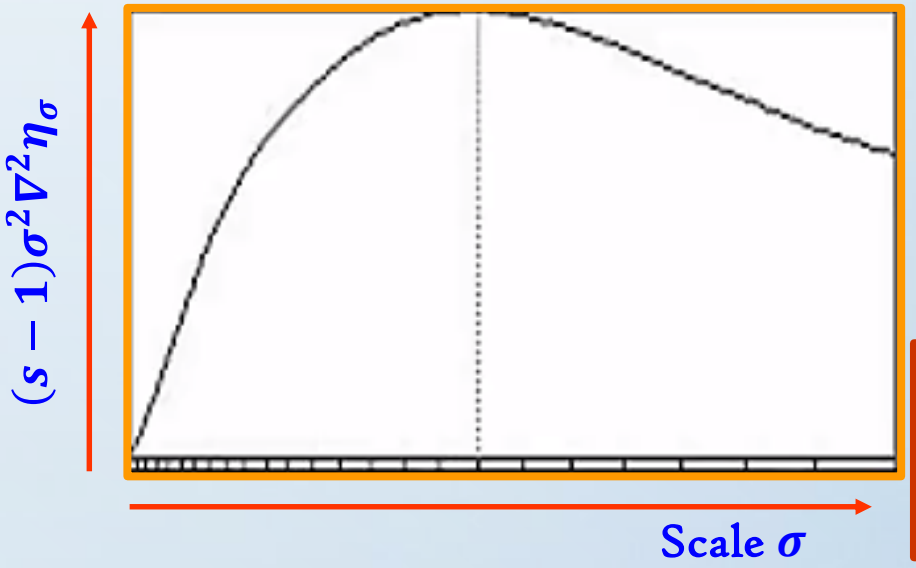


# SIFT Detector: Examples

# SIFT Detector: Examples

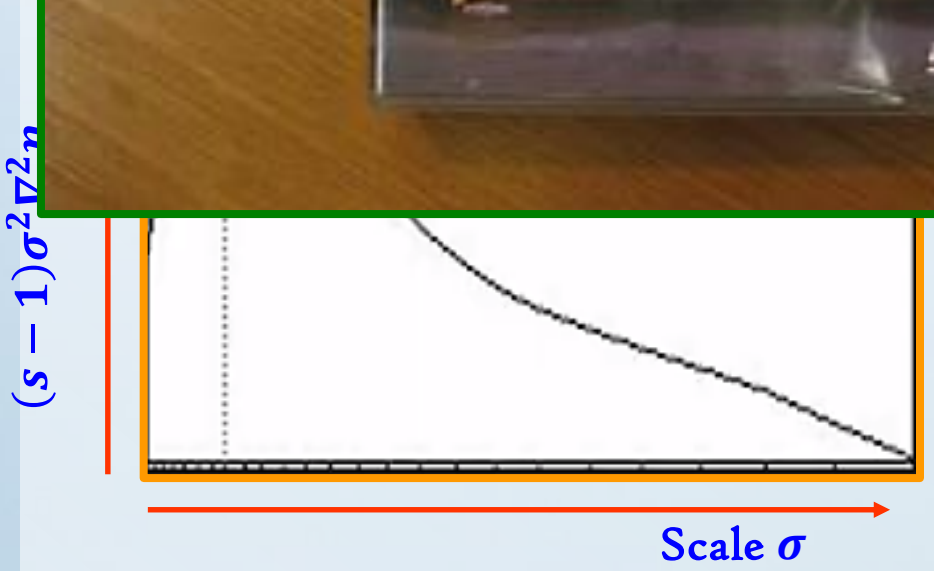
# SIFT Detector: Scale Invariance

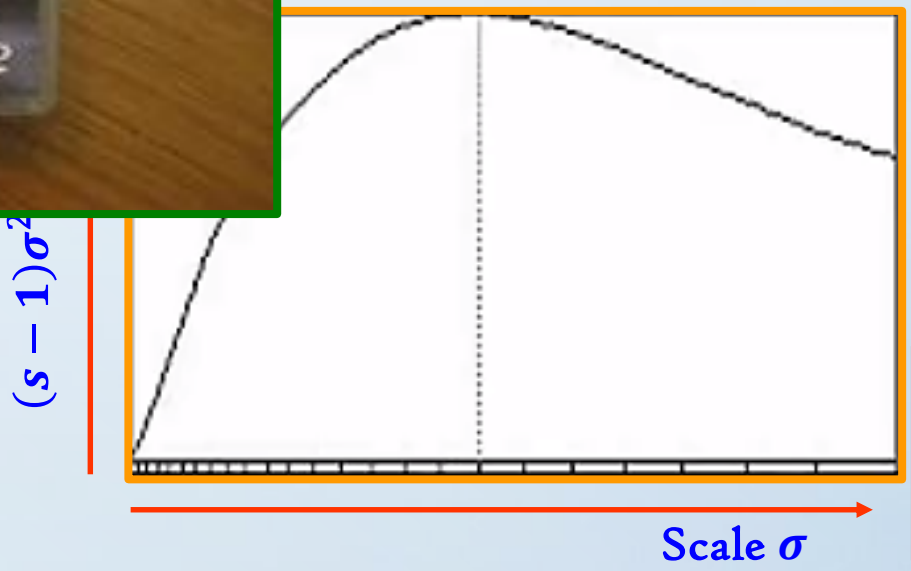





$\frac{\sigma_1^*}{\sigma_2^*}$ = *Ratio of Blob Size*

# SIFT Detector: Scale Invariance


$\frac{\sigma_1^*}{\sigma_2^*}$ = *Ratio of Blob Size*

# SIFT Detector: Computing the Principal Orientation

✓ Using the histogram of gradient direction inside a given scale at different grids.

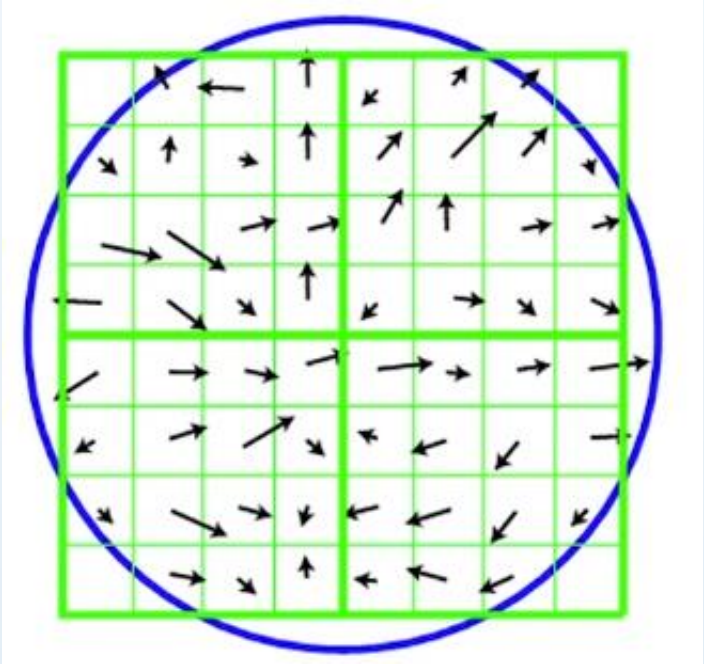
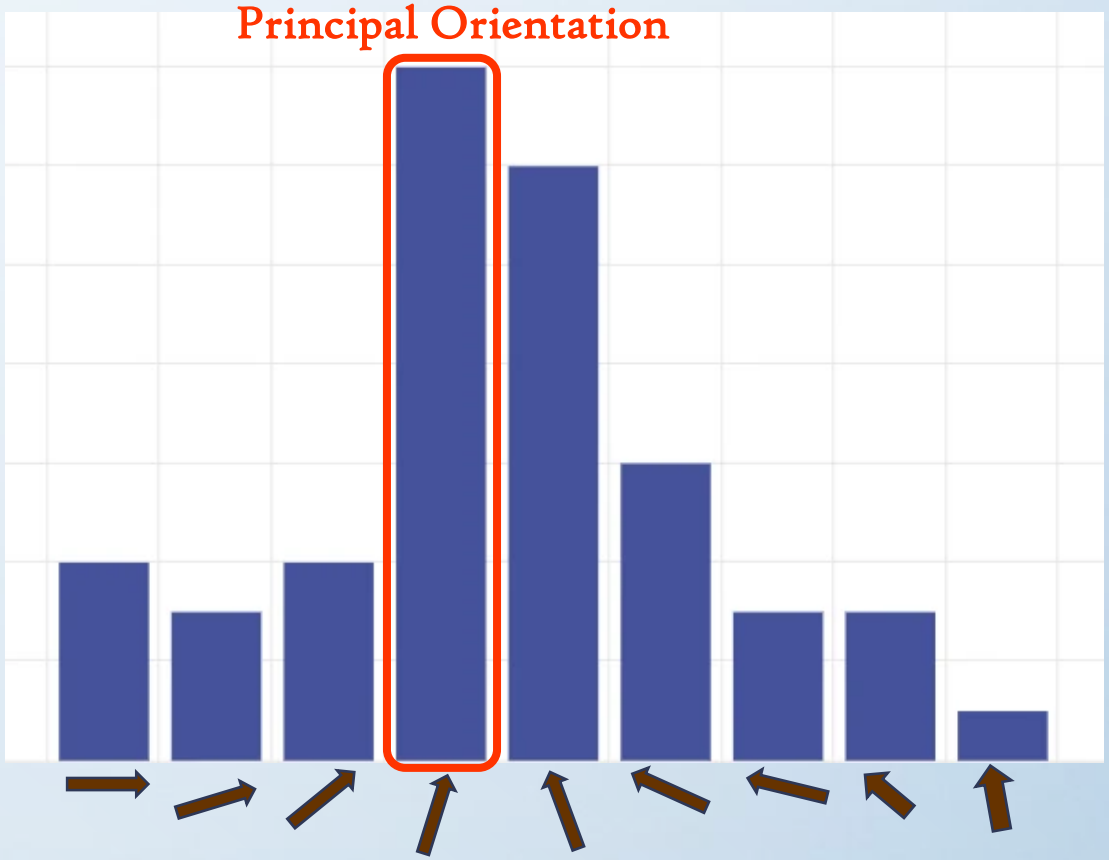


Image Gradient Directions

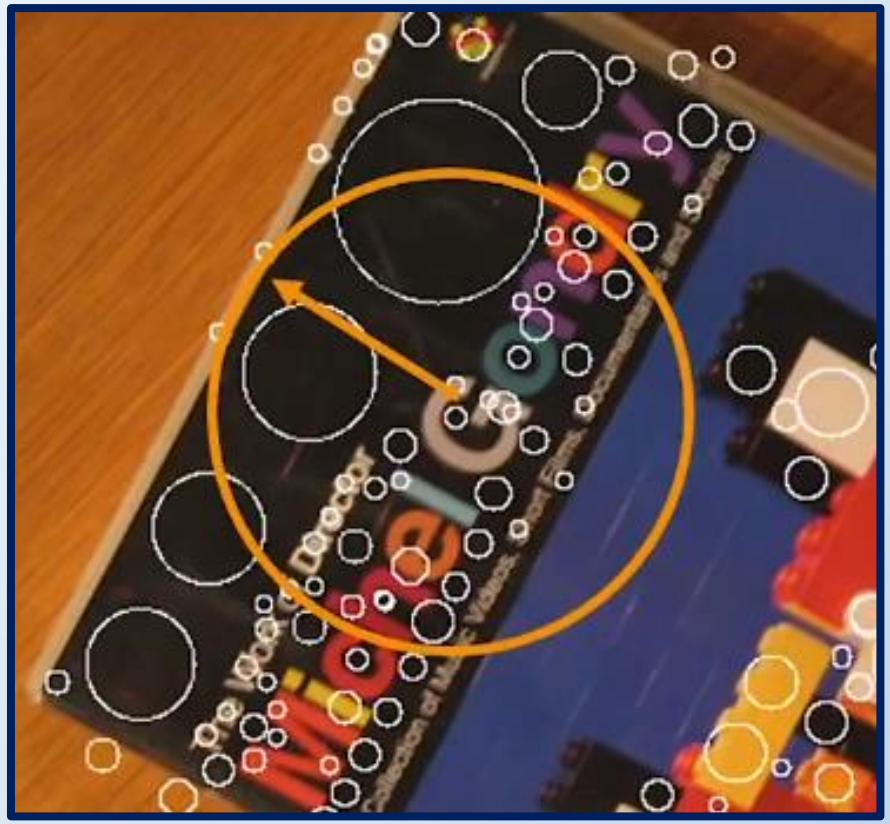
$$\theta = tan^{-1}\left(\frac{\partial I}{\partial y} \Big/ \frac{\partial I}{\partial x}\right)$$




Choosing the most prominent gradient direction

# SIFT Detector: Rotation Invariant

✓ Use the principal orientation to undo rotation.

# SIFT Detector: Rotation Invariant

✓ Use the principal orientation to undo rotation.


# SIFT Descriptor

✓ Histogram of gradient directions over spatial regions.

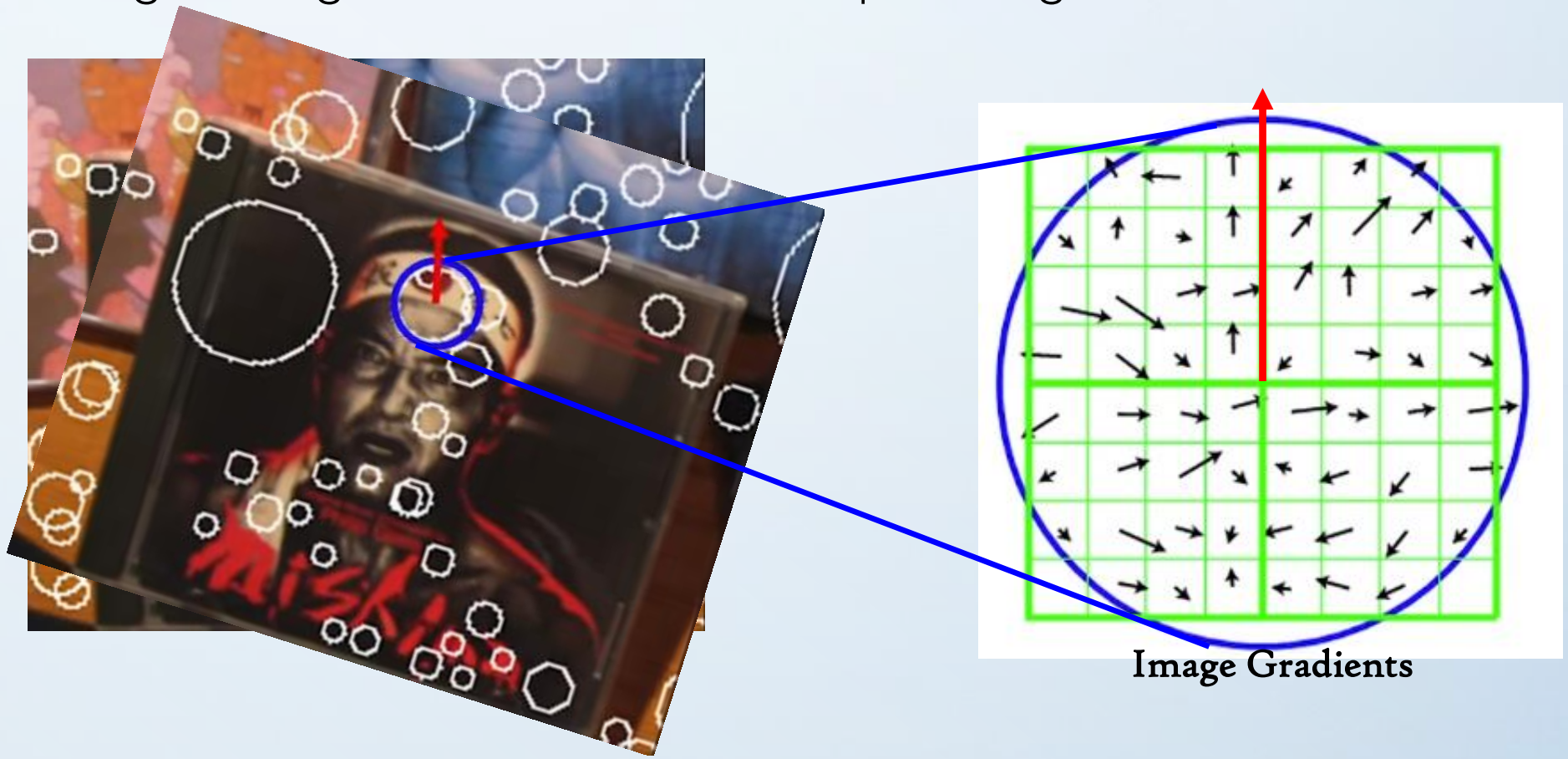
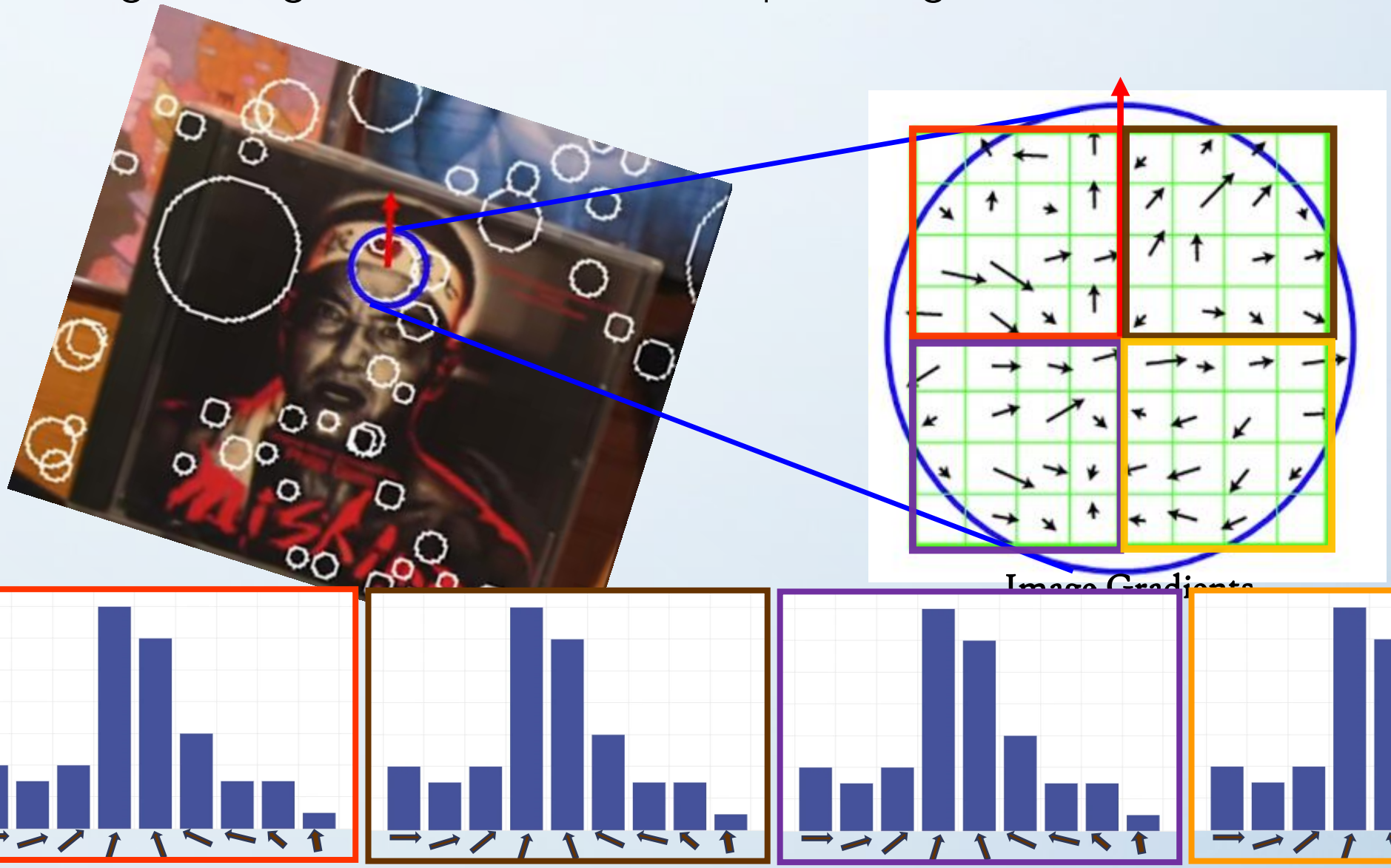


Image Gradients

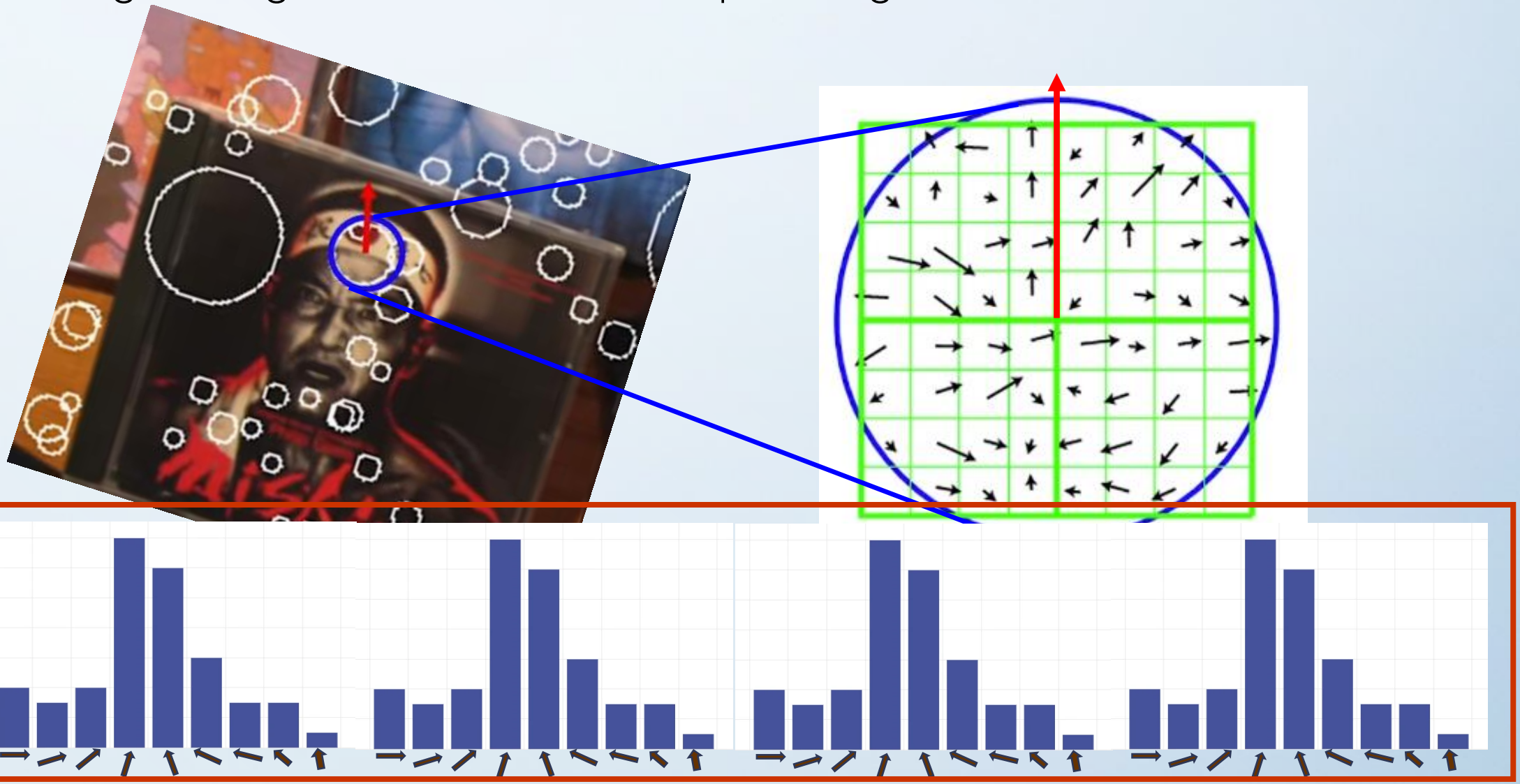
# SIFT Descriptor

✓ Histogram of gradient directions over spatial regions.



# SIFT Descriptor

✓ Histogram of gradient directions over spatial regions.



Normalized histogram: Invariant to scale, rotation and brightness.

# Comparing SIFT Descriptors

- ✓ Essentially comparing two arrays of data.
- ✓ Let $H_1(k)$ and $H_2(k)$ be the two arrays (histograms) of data of length $N$.

- ✓ $L_2$ Distance:

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

Smaller the distance $d(\mathrm{H}_1, \mathrm{H}_2)$ better the match.

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

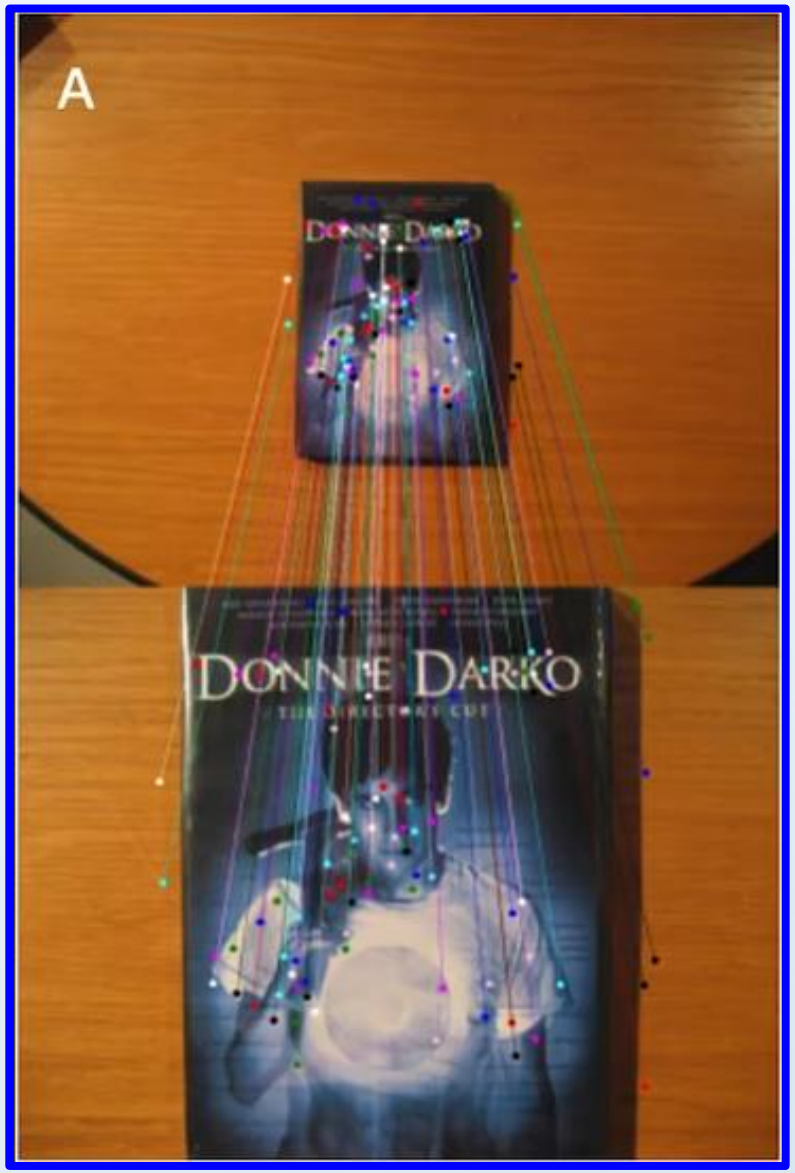
**Normalized Correlation:**

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

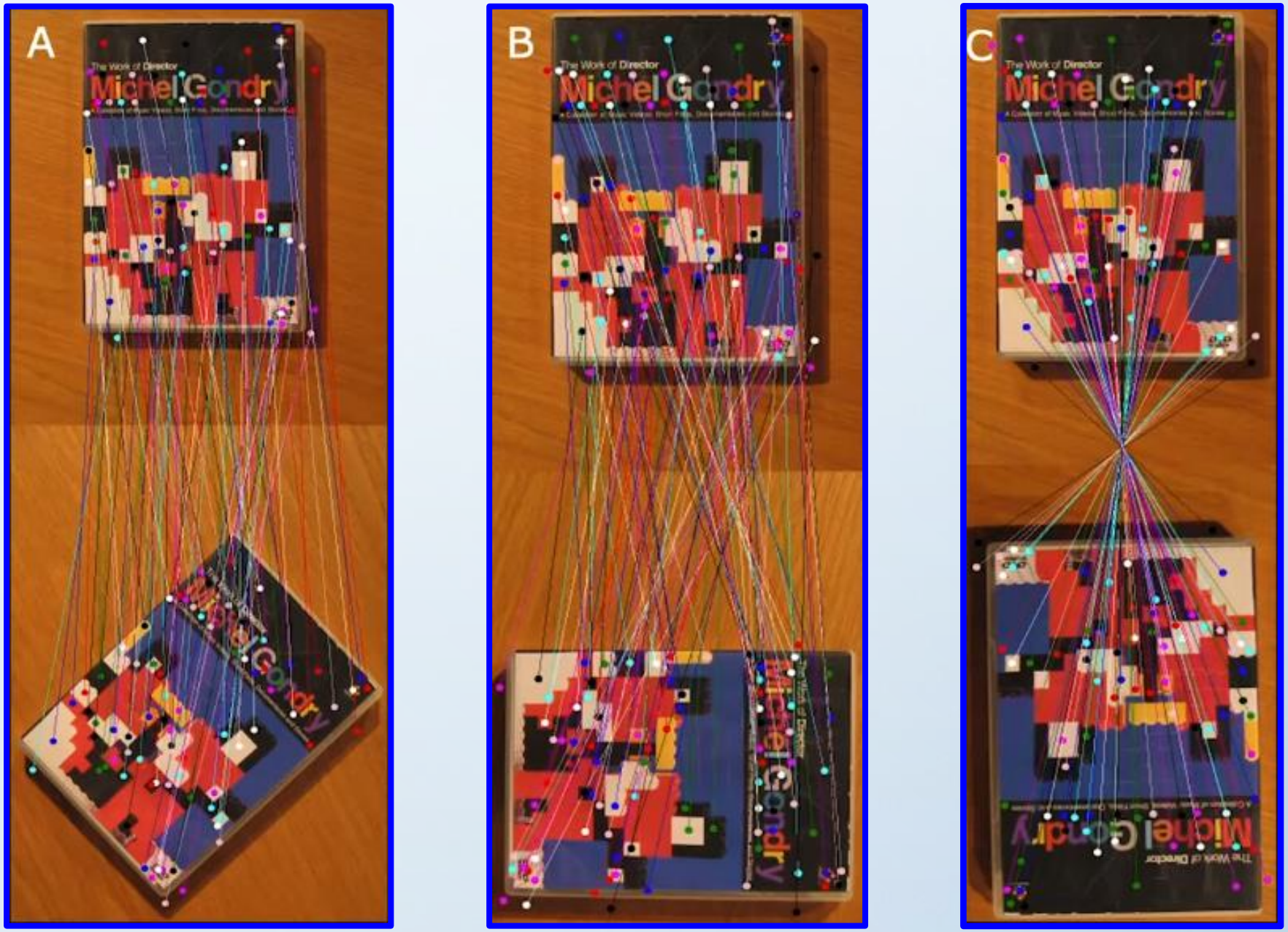
Larger the distance $d(\mathrm{H}_1, \mathrm{H}_2)$ better the match.

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

# SIFT Results: Scale Invariance




# SIFT Results: Rotation Invariance



# SIFT Results: Robustness to Cluster

# SIFT (Python Implementation)

```
# Check Version ---- Must be version 4.4.0
cv2.__version__
# Otherwise install OpenCV Version = 4.4.0----
!pip install opencv-contrib-python==4.4.0.44
import cv2 as cv2
```

```
rgb_img = cv2.cvtColor(cv2.imread("/content/img1.bmp"),
cv2.COLOR_BGR2RGB)
gray_img = cv2.cvtColor(rgb_img, cv2.COLOR_RGB2GRAY)
# Apply SIFT detector for Key-Point Extraction ---
feature_extractor = cv2.SIFT_create()

# find the keypoints and descriptors with chosen feature_extra
ctor
kp_1, desc_1 = feature_extractor.detectAndCompute(gray_img,
None)

SIFT_DetImg = cv2.drawKeypoints(rgb_img, kp_1, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

plt.figure(figsize=(12, 12))
plt.imshow(SIFT_DetImg)
plt.title("Detected Feature Points: SIFT", fontweight = 'bold'
)
plt.xticks([]), plt.yticks([])
```

## SIFT (as patented) and Version of Python

the first thing to do is see the exact version you are using, all just running:

```
print (cv2 .__ version__)
```

if **version** = **4.4.0** then `sift = cv2.SIFT_create ()`

if **version** = **4.3.x** then `sift = cv2.xfeatures2d.SIFT_create ()`

if **Version** = **4.2.x or 4.1.xu 4.0.x**, then SIFT will not work, it is not taken into consideration during the construction of the python package, the activation of the open-contrib module as well as the use of algorithms non free have not been activated.

on google colab you can install the opencv version you want by simply using a **pip** command preceded by an exclamation point **"!"** and specify the opencv version as follows:

```
!pip install opencv-contrib-python==4.4.0.44
```

**Note:** As I write this, the last available version of openv in C ++ is version 4.5.0 , and the latest version of opencv python package is 4.4.0.44

Ref: https://stackoverflow.com/questions/64474059/how-to-use-sift-and-surf-algorithm-in-google-colab-with-opencv4

# Other Descriptors

- ✓ Speed-Up Robust Features (SURF)
  - ✓ Both SIFT and SURF are patented. Require authors permission for commercial use.
- ✓ Gradient Location-Orientation Histogram (GLOH)
- ✓ Binary Robust Independent Elementary Features (BRIEF)
- ✓ Oriented Fast and Rotated Brief (ORB)
- ✓ Many more...