Welcome to Computer Vision

Computer Vision

Dr. Muhammad Tahir

DEPARTMENT OF COMPUTER SCIENCE,

FAST-NUCES, Peshawar

Course Details

LECTURES: Monday
& Wednesday

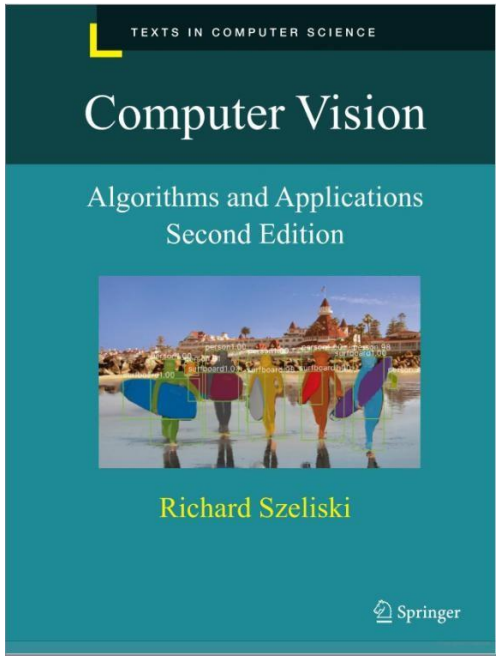
TIMINGS:

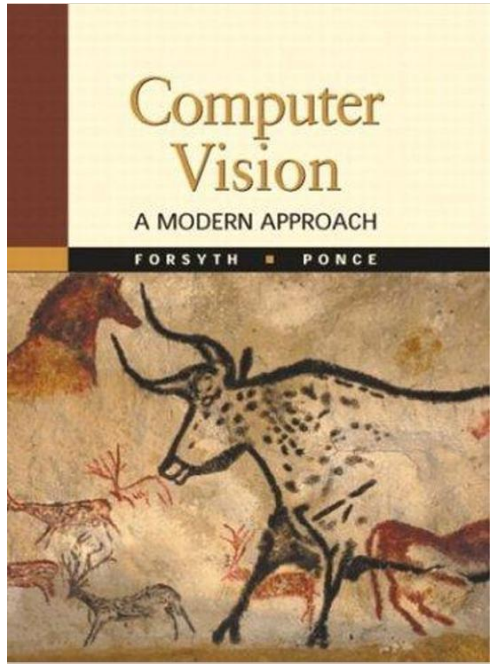
9:30 am – 11:00 am

MY OFFICE:

OFFICE HOURS:

EMAIL: m.tahir@nu.edu.pk





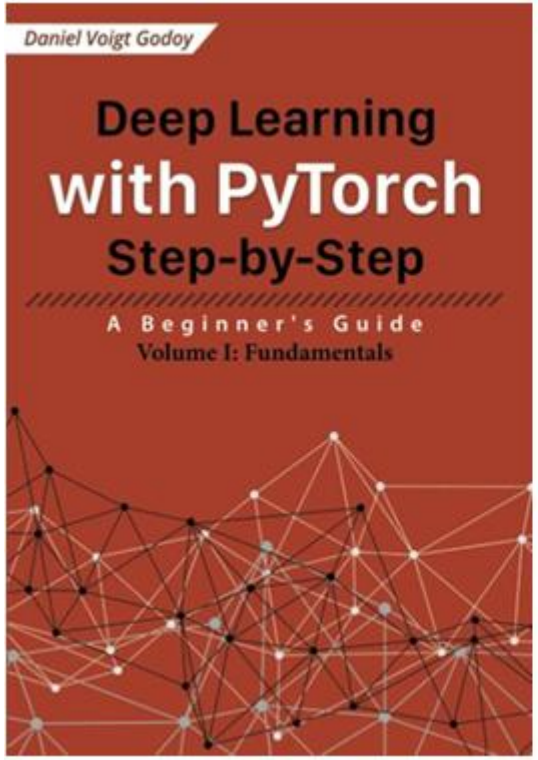
References

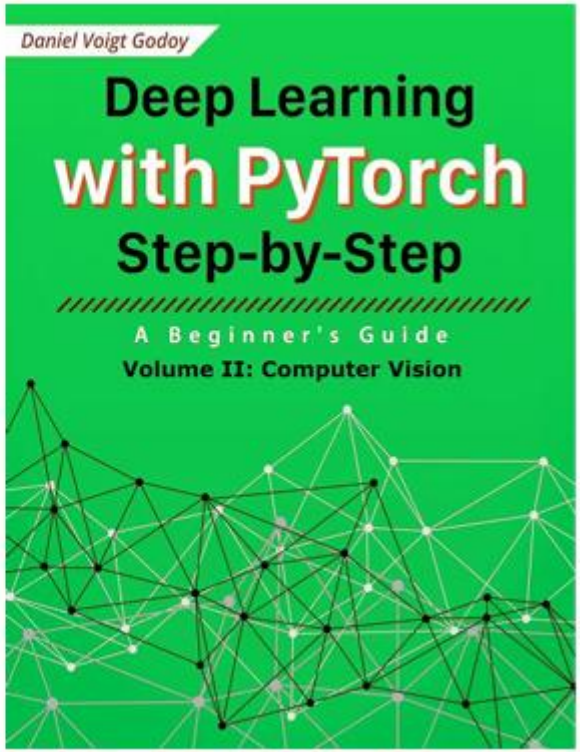
The material in these slides are based on:

1. Rick Szeliski's book: Computer Vision: Algorithms and Applications
2. Forsythe and Ponce: Computer Vision: A Modern Approach

Recommended Books

Deep Learning with PyTorch Step-by-Step by Daniel Voigt Godoy





Course Learning Outcomes

No	CLO (Tentative)	Domain	Taxonomy Level	PLO
1	Understanding basics of Computer Vision: algorithms, tools, and techniques	Cognitive	2	
2	Develop solutions for image/video understanding and recognition	Cognitive	3	
3	Design solutions to solve practical Computer Vision problems	Cognitive	3	

Outline

- Feature Extraction
- Interest Points
- Corner Detectors

A Problem with Edges

- Edges are insensitive to intensity changes, but not to other image transformations

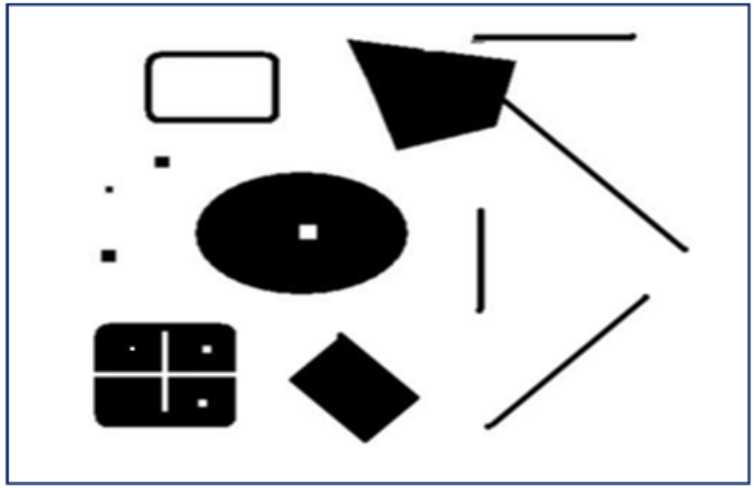
- **Insensitive to Intensity Changes**
  - Even though the **brightness** of the two images is different (left = bright, right = darker), the **edges** (places where pixel intensity changes sharply) are still in the same locations.
  - So, edge detectors (like Sobel, Canny) will detect similar boundaries in both images. This shows that **edges are robust to illumination changes**.

A Problem with Edges

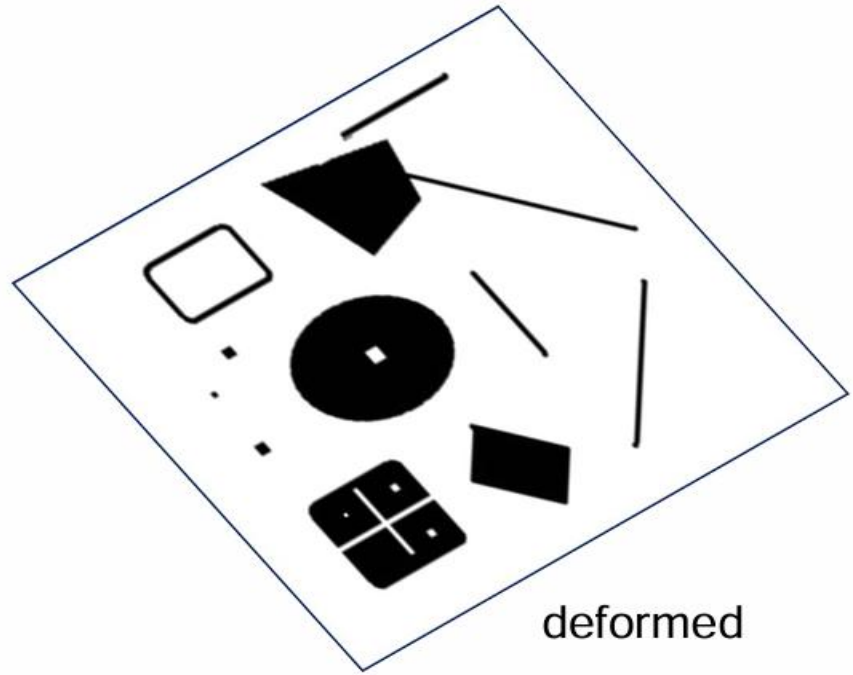
- **Not Invariant to Other Transformations**
  - If the cow was **rotated, scaled, or deformed** (instead of just brightness changed):
    - The positions of the edges would change.
    - Edge detection is not robust to **rotation, scale, or affine transformations**.
- Example: If we rotate the cow 90°, the detected edge map would also rotate, so direct comparison fails.
- This means **edges alone are not good for recognizing objects across different viewpoints**.

Interest Points

- interest points = keypoints = features
  - These are locations in an image that are *distinctive* and can be found again after transformations (rotation, scaling, deformation, etc.).
- A low-level building block in many computer vision applications
- Suppose you have to click on some point, go away and come back after I deform the image, and click on the same points again.
- Which points would you choose?



original



deformed

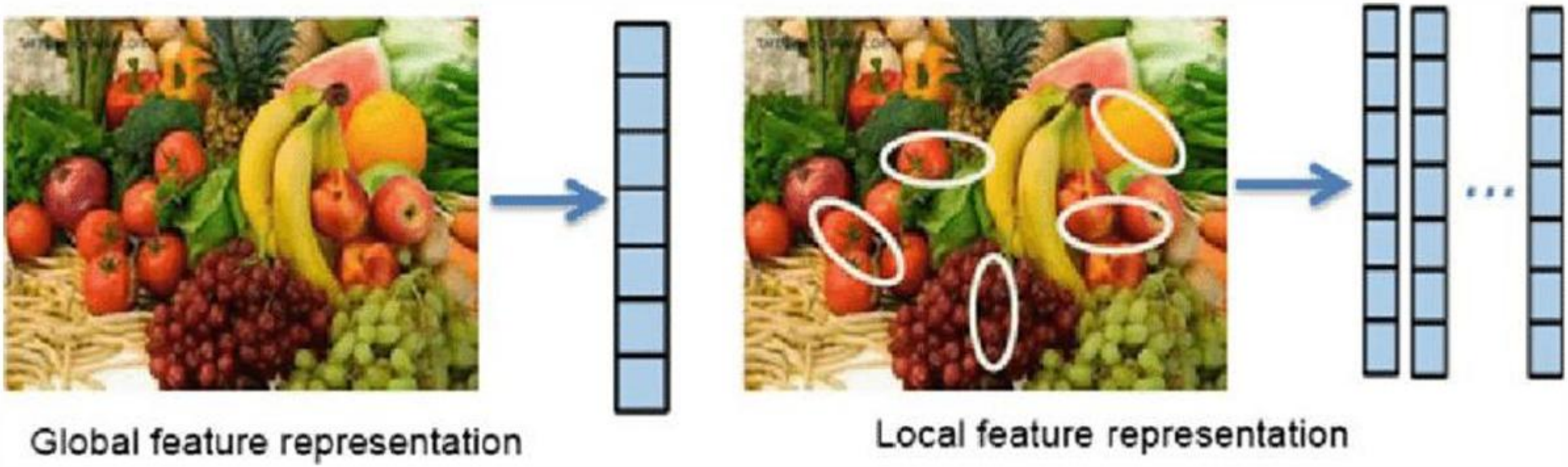
Not all points in an image are equally good choices for repeatable features

- **Bad choices** → Points on **flat regions** (inside a white area, or inside a solid black shape).
  - These don't have local uniqueness. After transformation, you won't know *exactly* where you clicked.
- **Bad choices** → Points on **edges/lines**.
  - Along an edge, many points look similar. Small shifts will confuse the detector.
- **Good choices** → Points on **corners, junctions, or blobs**.
  - These are unique, well-defined spots (e.g., the corner of a rectangle, or where lines intersect).
  - Even after deformation (rotation, scaling, affine transformation), you can find these points again.
- That's why **corner detectors (Harris, Shi–Tomasi)** are so important: they automatically find the most reliable points.

Local Image Features

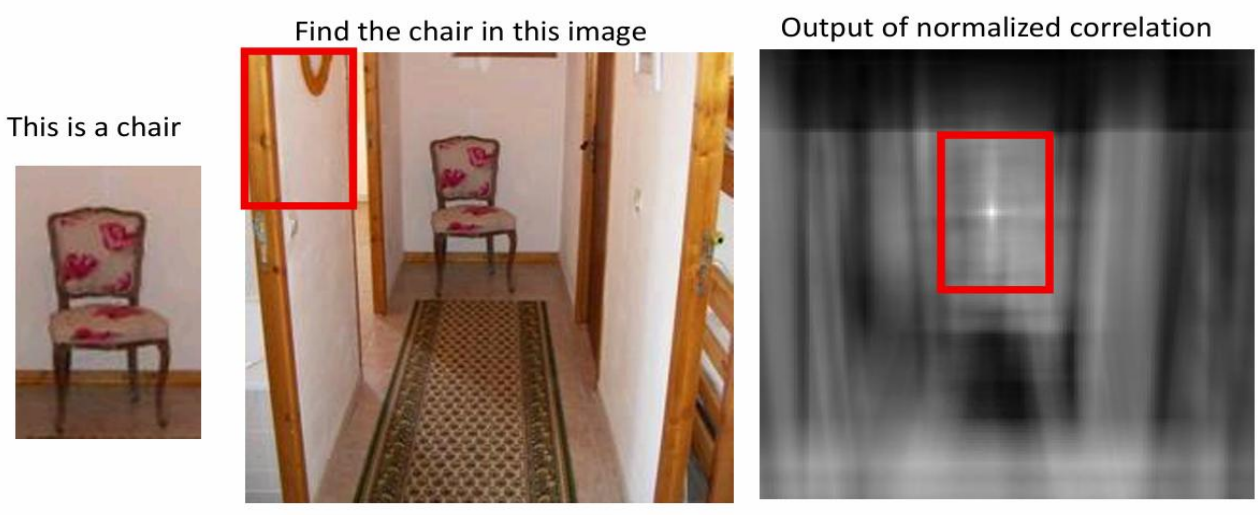
- What Are Local Features / Interest Points?
- Local features refer to a pattern or distinct structure found in an image, such as
  - A point. An edge. Small image patch, Corner, Blob
- They are usually associated with an image patch that differs from its immediate surroundings by texture, color, or intensity.
- What the feature represents does not matter, just that it is distinct from its surrounding

Local Image Features



Object recognition: Is it really so hard?

- **Global Template Matching** works by comparing the template image with every possible sub-region of the larger image (using correlation).
- if the object looks exactly the same (same scale, orientation, lighting), this works.



But in reality:

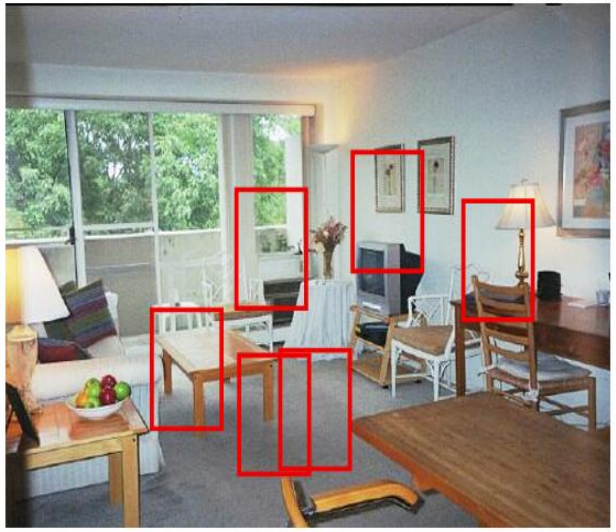
- Objects appear at different **scales** (closer/farther).
- They appear at different **orientations** (rotated, tilted).
- They appear under different **lighting or occlusions**.
- Backgrounds may confuse the correlation (false matches).

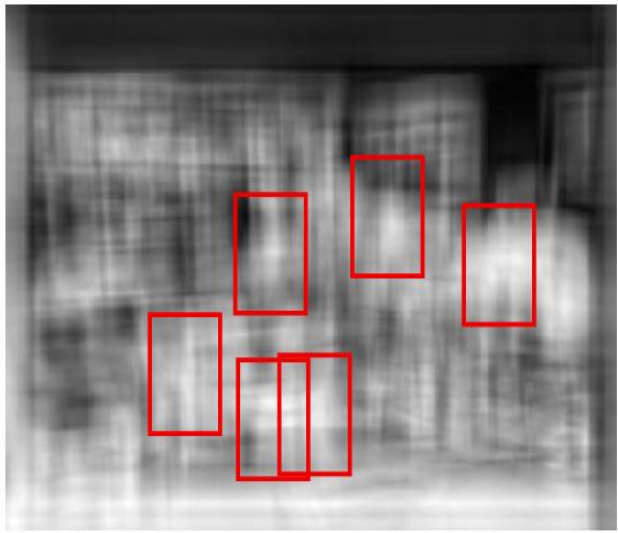
That's why the method often fails or gives misleading results

Object recognition: Is it really so hard?

- Real-world object recognition is hard because objects rarely look identical across scenes.
- These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts.” Nivatia & Binford, 1977.
- That’s why we need better features (corners, keypoints, descriptors like SIFT/SURF/ORB) that are repeatable and invariant.

Find the chair in this image

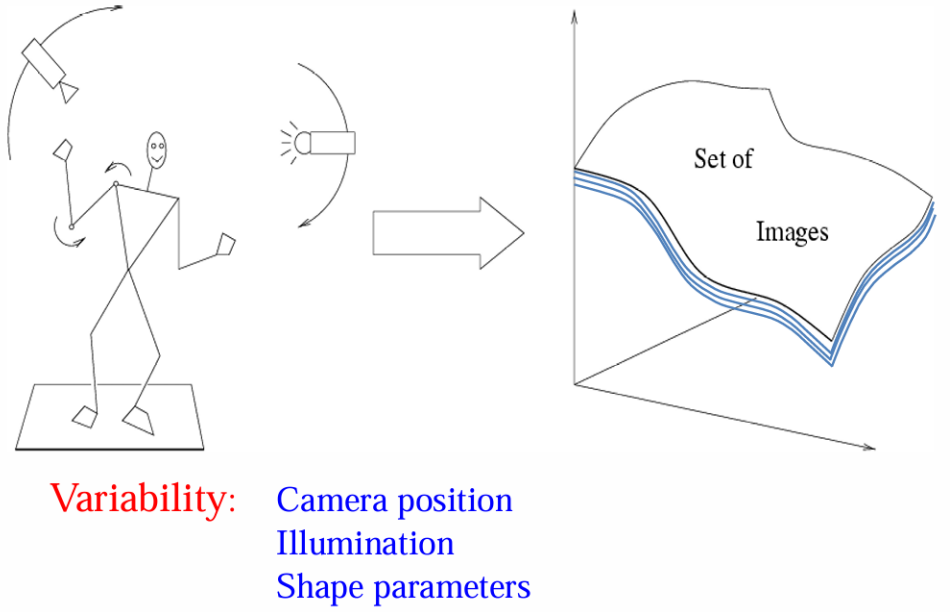




Pretty much garbage
Simple template matching is not going to make it

And it can get a lot harder

Why is this hard?



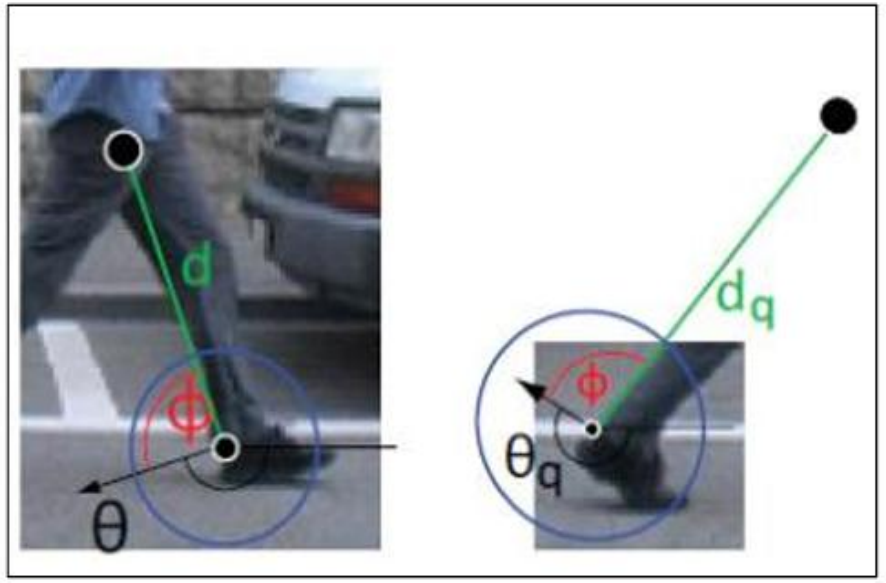
Variability:
Camera position
Illumination
Shape parameters

- Several sources of **variability** affect how the person looks in an image:
  - **Camera position** – if the camera moves (different viewpoints, angles), the same object looks very different.
  - **Illumination** – changing the lighting direction/intensity produces different shadows and highlights.
  - **Shape parameters** – the object itself may deform or move (person raises arms, changes posture).

- Object recognition is challenging because we don't just need to match one picture of an object.
- We must recognize the object across its variability (viewpoint, lighting, deformation).
- A robust recognition system should learn to group all these images into the same object identity, even though the raw pixels are different.

Motivation for using Local Features

- Global representation have major limitations
  - Occlusions
  - Articulation
  - Intra-Category Variations

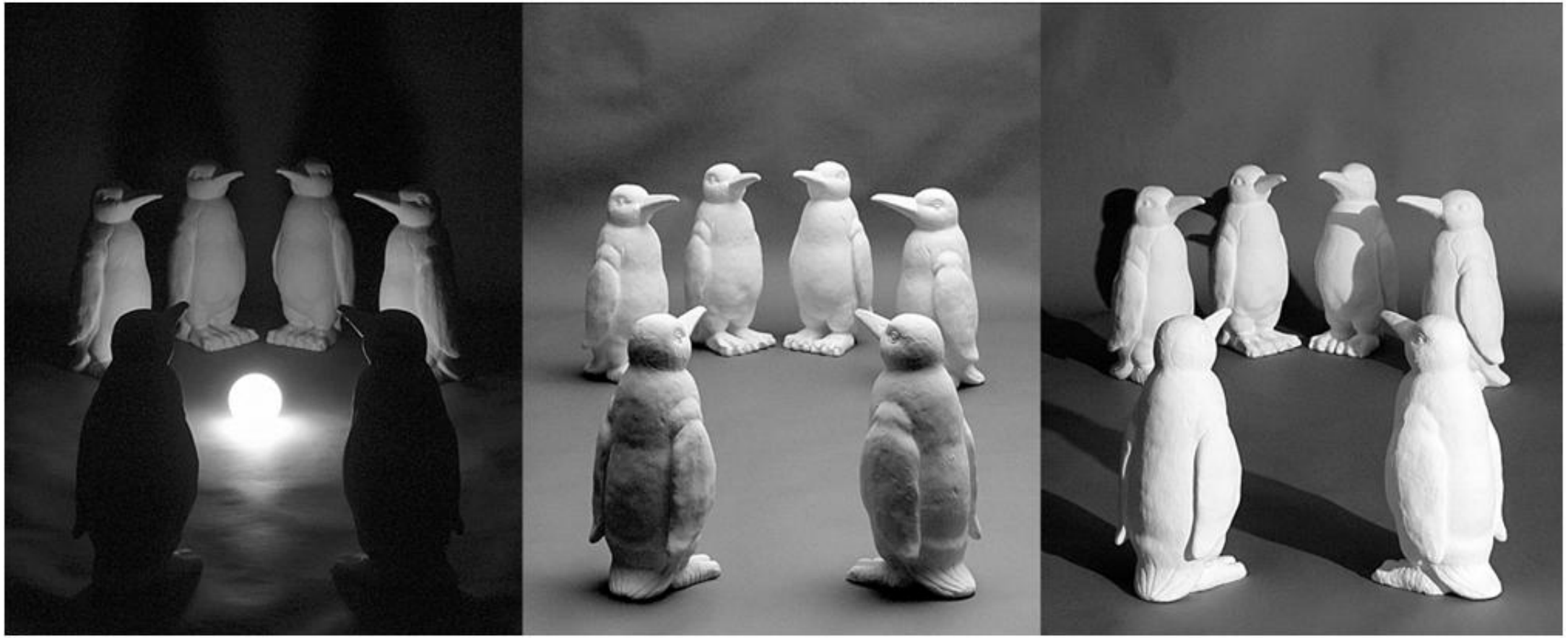


How many object categories are there?

Challenge: variable viewpoint

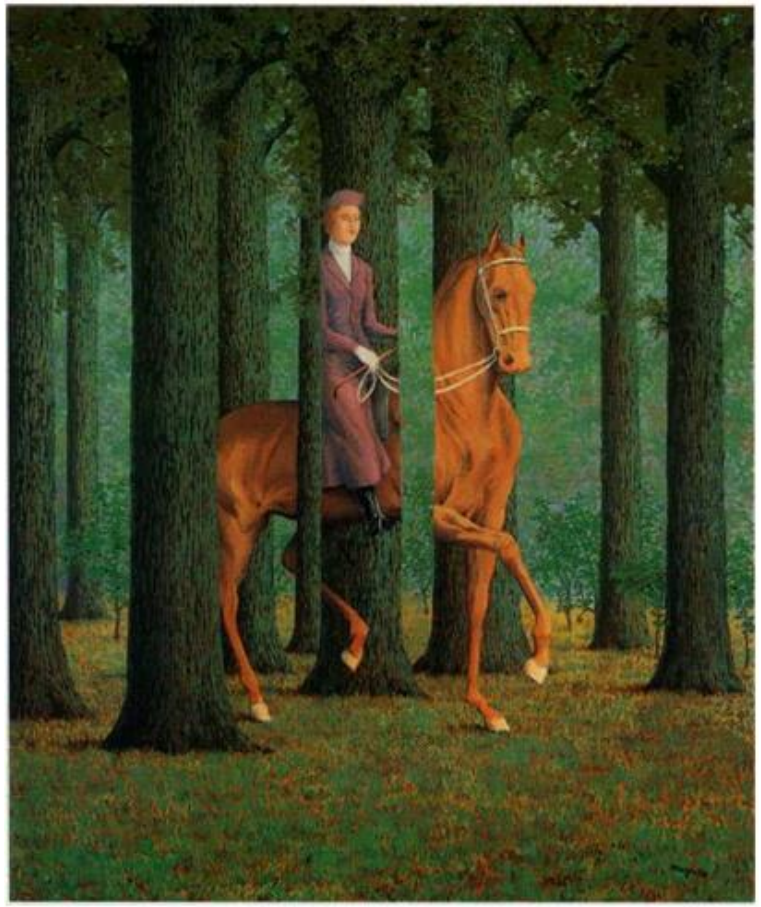
Michelangelo 1475-1564

Challenge: variable illumination



Challenge: deformation

Challenge: Occlusion



Magritte, 1957

Occlusion

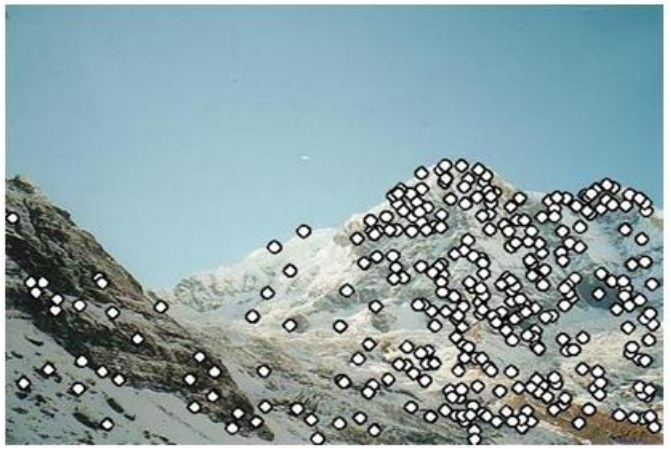
Challenge: background clutter

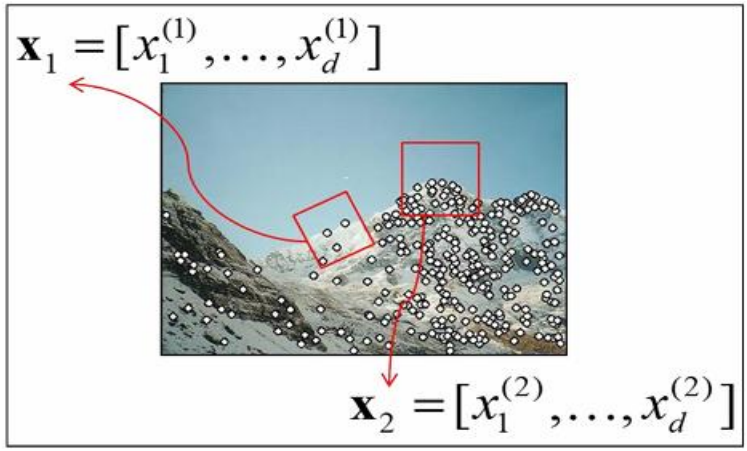
Kilmeny Niland. 1995

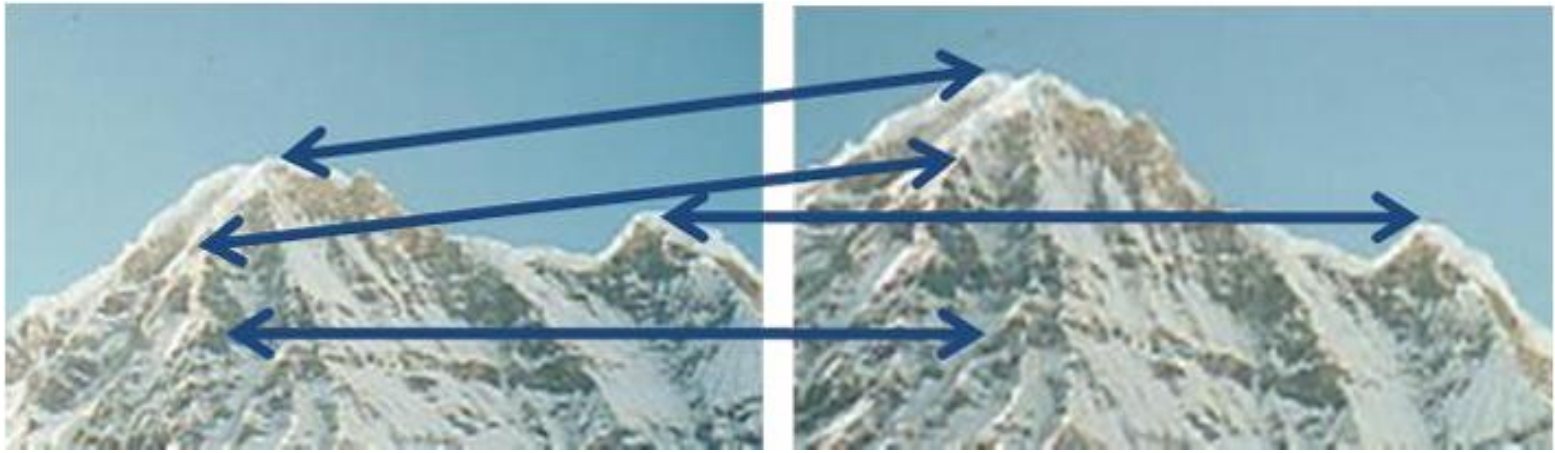
Challenge: intra-class variations

Local or Interest Point Based Matching Components

- **Detection**: Identify the interest points
- **Description**: Extract vector feature descriptor surrounding each interest point.
- **Matching**: Determine correspondence between descriptors in two views

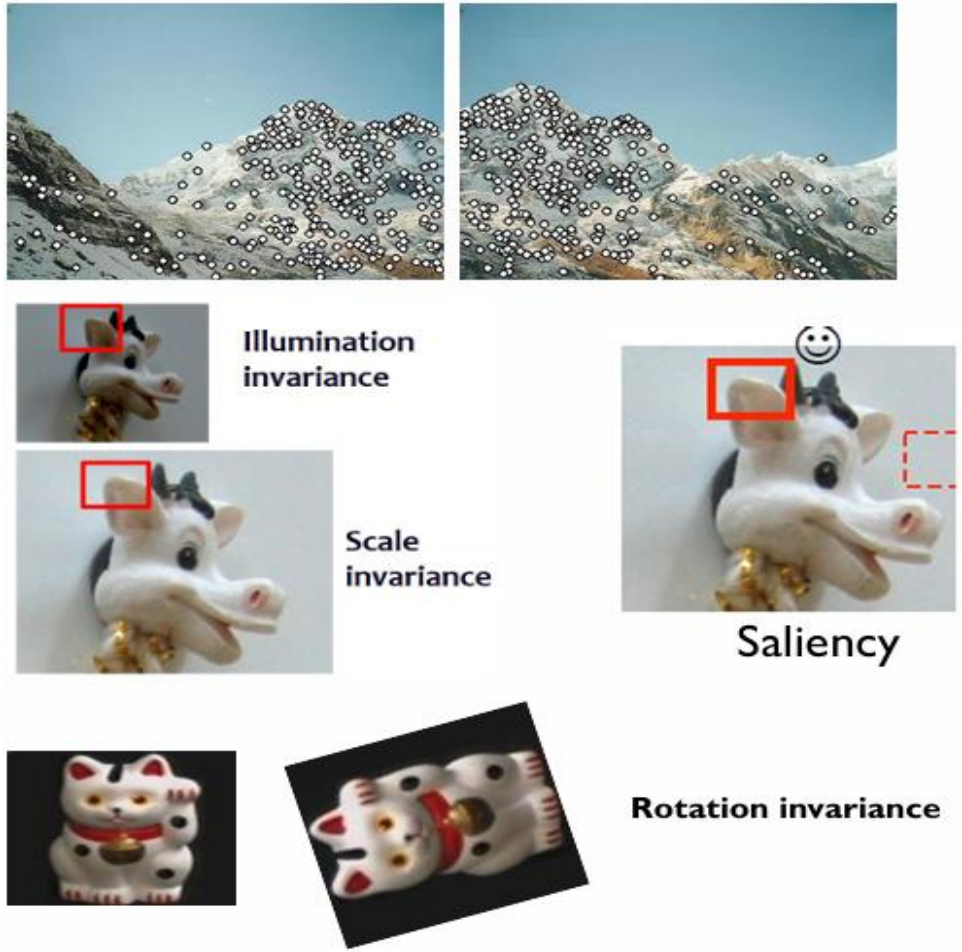






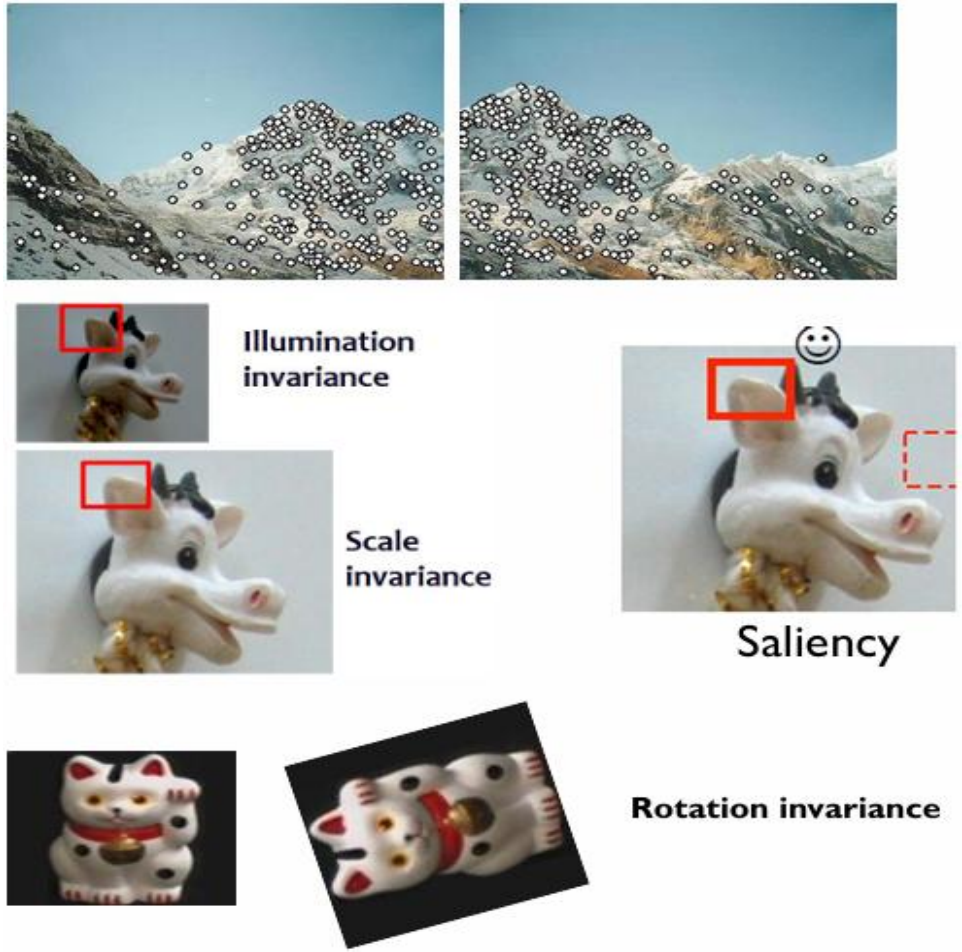
Goals/Characteristics of Good Interest Points

- **Repeatability**
  - A good feature should appear consistently in multiple images of the same object, even under changes in:
    - **Illumination** (lighting conditions),
    - **Scale** (zooming in or out),
    - **Rotation** (object rotated).
  - Example: The same corner on a cow toy can be detected whether the image is bright, dark, zoomed, or rotated.
  - This ensures reliable matching across different conditions.
- **Saliency**
  - Each feature should be **distinctive** (not confused with others).
  - It should provide a **unique signature** that makes it easy to match with its counterpart in another image.
  - Example: The cow's ear is a unique feature, whereas a flat region may not be distinctive.



Goals/Characteristics of Good Interest Points

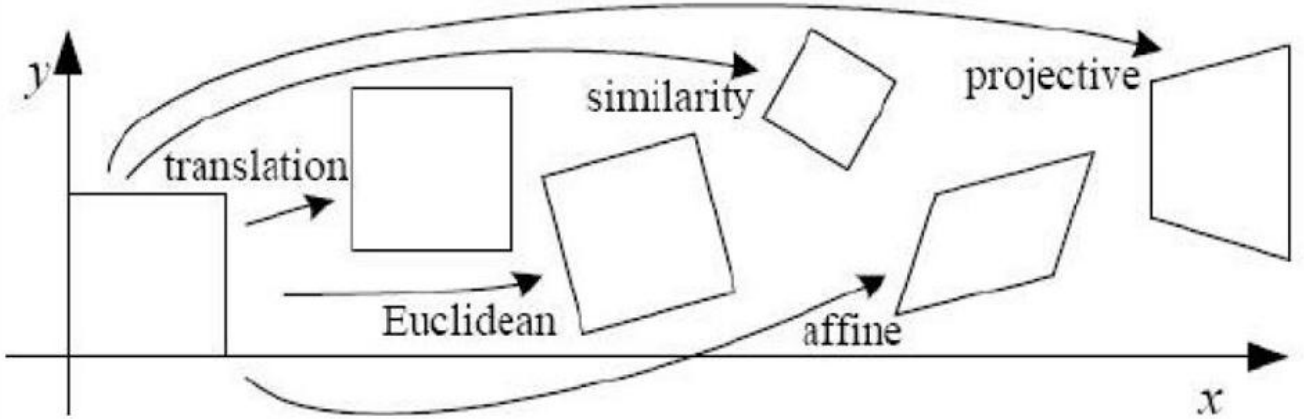
- **Compactness and Efficiency**
  - Ideally, we want **fewer features than pixels**.
  - Storing and matching millions of pixels is impractical, so extracting only the most informative points is efficient.
  - Example: Instead of all pixels, only strong corners or blobs are extracted.
- **Locality**
  - A good feature should occupy a **small, local area** of the image.
  - This makes it more **robust to clutter and occlusion** (e.g., if part of the image is hidden, the visible local features can still help recognition).
  - Example: Even if the cow is partially hidden, some keypoints (like the ear or nose) remain detectable.



Goals/Characteristics of Good Interest Points

- Interest point must provide some invariance to geometric and photometric differences between two views.
- Geometric Transformations
  - Translation
  - Euclidean Transformation = translation + rotation
  - Similarity Transformation = translation + rotation + uniform scaling
  - Affine Transformation = non-uniform scaling, rotation, shear
  - Projective Transformation (Homography) = viewing the object from an angle
- Photometric Variability
  - Includes **illumination changes** (lighting direction, brightness, contrast)

Interest points must be stable under transformations like translation, scaling, rotation, affine, projective distortions, and lighting changes.



Goals/Characteristics of Good Interest Points

- Same object can look brighter in sunlight, darker in shadows, or tinted under colored lighting

- A simple way to model this is using a **linear transformation** on pixel intensities:

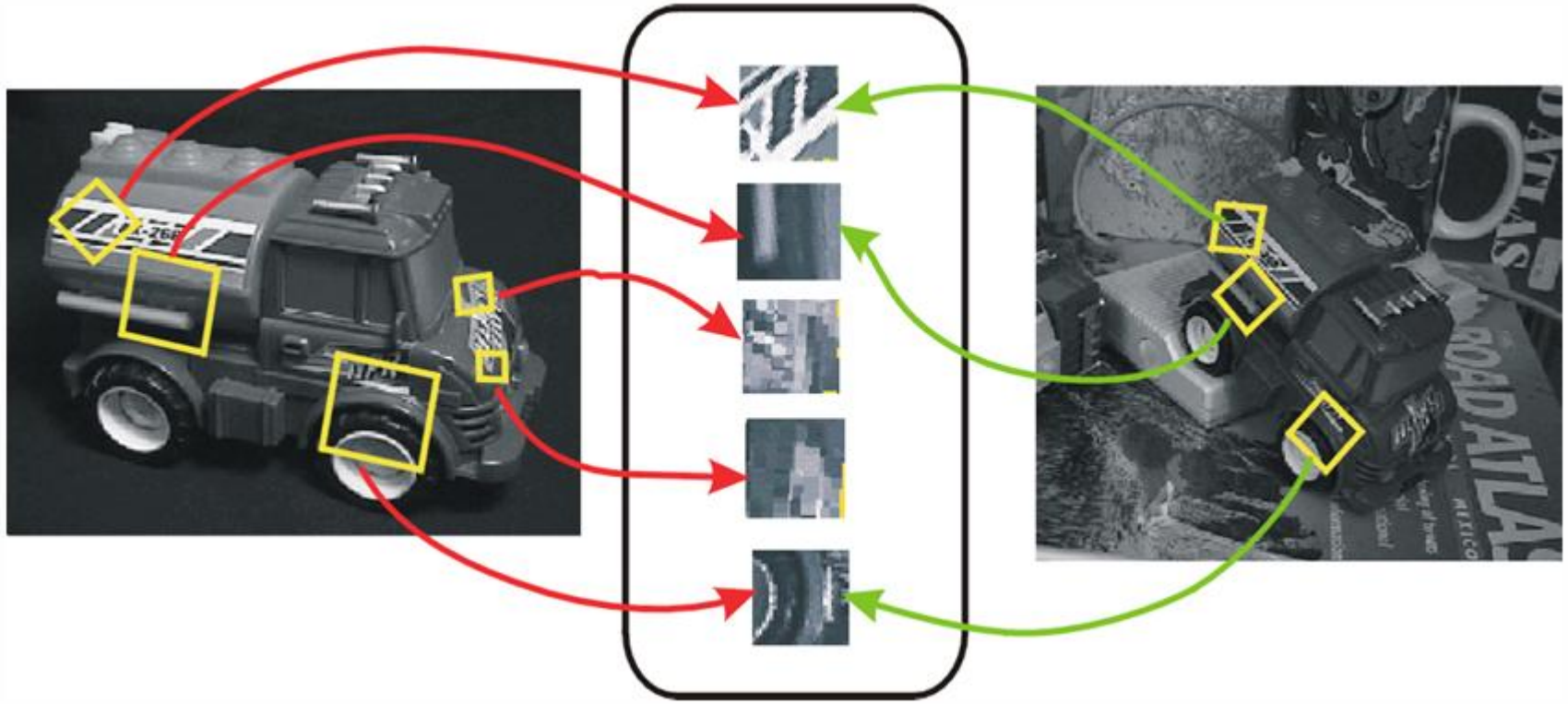
$$\acute{I}(x, y) = a.I(x, y) + b$$

- $I(x, y)$: Original pixel intensity
- a: Scaling (contrast change, e.g. brighter/darker)
- b: Offset (brightness shift, e.g. adding uniform light)
- $I'(x, y)$: New pixel intensity

Goals/Characteristics of Good Interest Points

Invariance: Geometric Transformations

Invariant Example



Applications

- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition

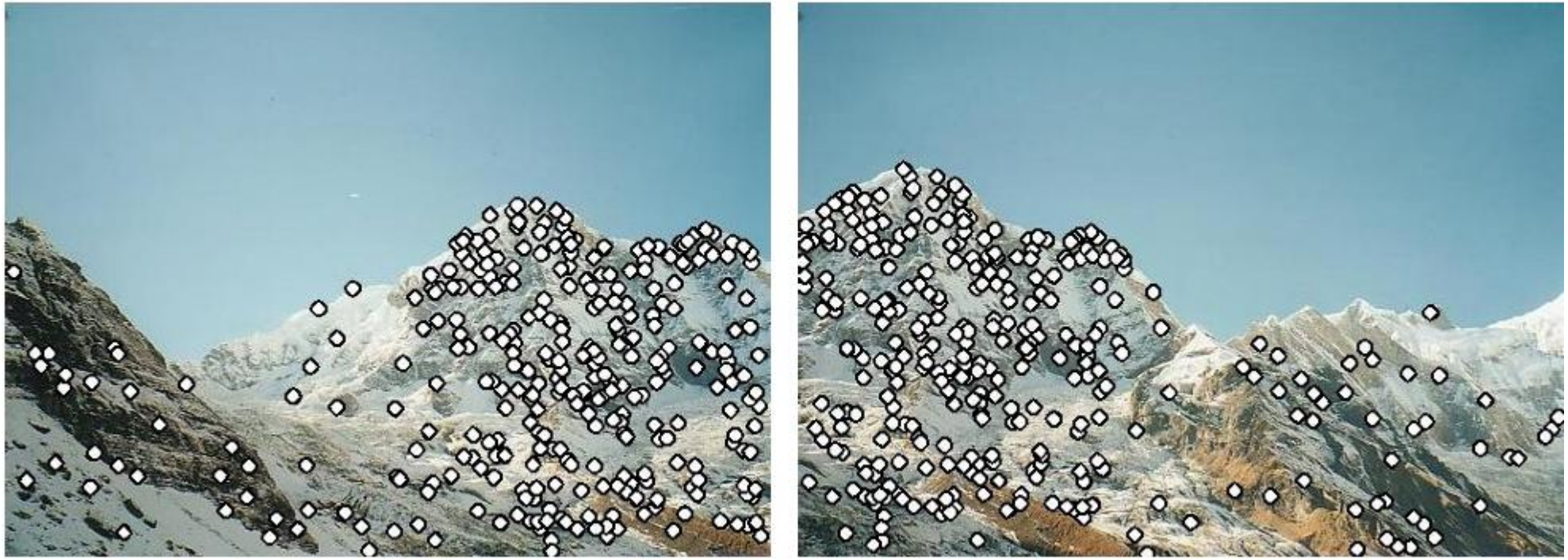
Application 1: Build a Panorama

How do we build panorama?

We need to match (align) images

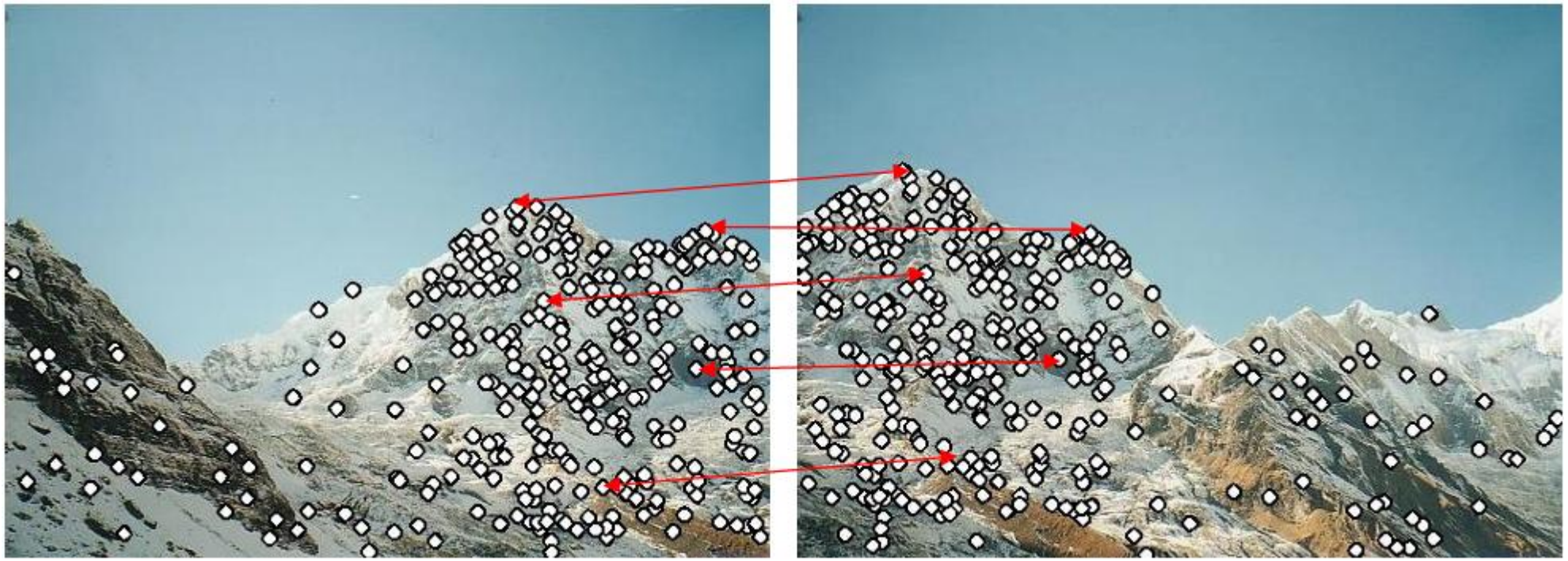
Matching with Features

Detect features (feature points) in both images



Matching with Features

- Detect features (feature points) in both images
- Match features - find corresponding pairs



Matching with Features

- Detect features (feature points) in both images
- Match features - find corresponding pairs
- Use these pairs to align images

Matching with Features

Problem 1:

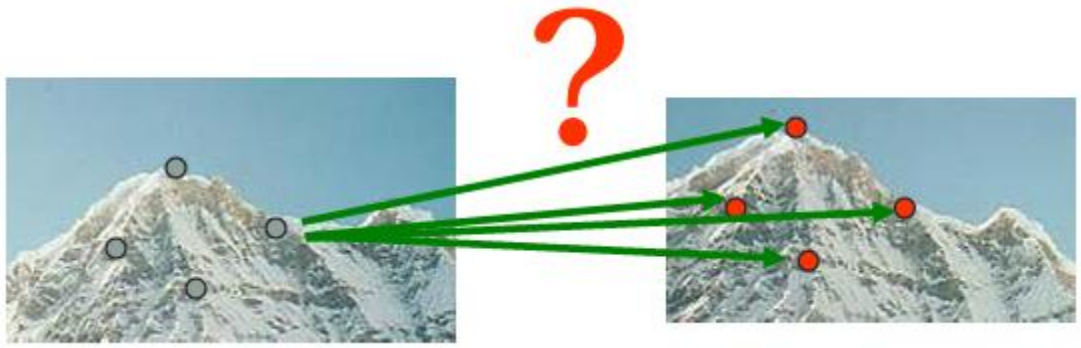
- Detect the same point independently in both images

- no chance to match!
- **We need a repeatable detector**

Matching with Features

Problem 2:

- For each point correctly recognize the corresponding one



- **We need a reliable and distinctive descriptor**

Application 1: Build a Panorama

- **The Problem: Stitching Images Together**
  - You take two overlapping photos of a mountain scene (left and right).
  - To create a wide panorama, we need to **align** them properly.
- **Finding Correspondences (Keypoints)**
  - Detect interest points (corners, distinctive patches) in both images.
  - Match pairs of points across the two images (e.g., the peak of the mountain, rocks, etc.).
  - These **matching pairs** allow us to understand how one image relates to the other.

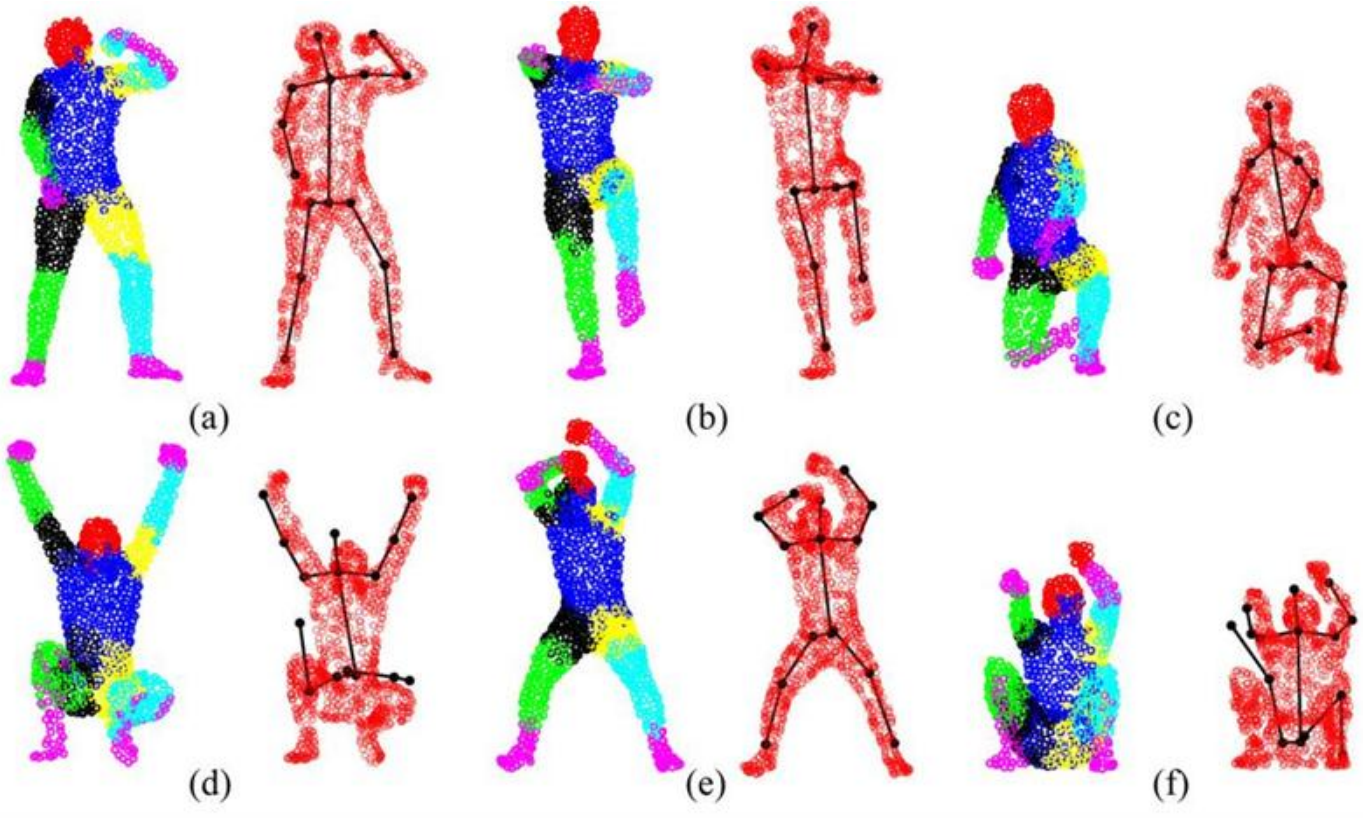
Application 2: Image Registration / Fusion

- **Image Registration:** Aligning two or more images of the same scene taken at different times, from different sensors, or from different viewpoints.
- To make sure that corresponding features (roads, buildings, objects, etc.) overlap correctly.
- Raw images may be misaligned due to scale, ngle, distortion, or sensor differences.

- **Image Fusion:** Combining information from multiple registered (aligned) images into a single image that contains more useful information than any of the individual images.
- To enhance image interpretation or analysis.
- **Example**: Fusing a thermal image with a visible-light image to get both heat information and structural detail.

Application 3: Pose Estimation

- **Pose Estimation** is the process of detecting and tracking the positions of human body joints (like head, arms, legs, torso) from images or videos.
- It essentially maps human movement into a set of **keypoints** (skeleton-like structure).
- These keypoints can then be used to understand the orientation and movement of the person.



Application of local Features

Local image features are widely used in various computer vision applications due to their ability to capture distinctive and repeatable patterns in images. Some common applications include:

1. **Feature Matching:** Local features are used to find correspondences between different images, which is essential for tasks like image alignment, panorama stitching, and object recognition.
2. **Object Detection and Recognition:** Local features are used to detect and recognize objects in images. They provide robustness to changes in scale, rotation, and illumination, making them suitable for object detection in challenging conditions.
3. **Image Registration:** Local features are used to align images from different sources or at different times, such as in medical image analysis or satellite image processing.

Application of local Features

4. **Augmented Reality:** Local features are used to track objects or scenes in real time, enabling the overlay of virtual objects onto the real world in applications like AR games or navigation systems.
5. **Image Retrieval:** Local features are used to index and retrieve images from large databases based on their visual content, enabling content-based image retrieval systems.
6. **3D Reconstruction:** Local features are used to reconstruct the 3D structure of objects or scenes from multiple images, as in structure-from-motion (SfM) and simultaneous localization and mapping (SLAM) systems.
7. **Gesture Recognition:** Local features can be used to recognize hand gestures or body poses in applications such as sign language recognition or human computer interaction.

Available Detectors

These detectors are basic building block for many Computer Vision Applications.

Harris	[Beaudet ‘78], [Harris ‘88]
Laplacian, DoG	[Lindeberg ‘98], [Lowe 1999]
Hessian-Laplace	[Mikolajczyk & Schmid ‘01]
Harris-/Hessian-Affine	[Mikolajczyk & Schmid ‘04]
EBR and IBR	[Tuytelaars & Van Gool ‘04]
MSER	[Matas ‘02]
Salient Regions	[Kadir & Brady ‘01]
Others…	

Available Descriptors

- Scale Invariant Feature Transform (SIFT)
- Speed-Up Robust Feature (SURF)
- Histogram of Oriented Gradient (HOG)
- Gradient Location Orientation Histogram (GLOH)
- Pyramidal HOG (PHOG)
- Pyramidal Histogram Of visual Words (PHOW)
- LBP, LTP and variants, HAAR;
- PCA-SIFT, VLAD, MOSIFT,
- Deepfeatures, CNN, Fisher vector,
- SV-DSIFT, BF-DSIFT, LL-MO1SIFT, 1SIFT, VM1SIFT, VLADSIFT,
- DECAF, Fisher vector pyramid, IFV
- Dirichlet Histogram
- Simplex based STV (3-D), MSDR;

Harris Detector

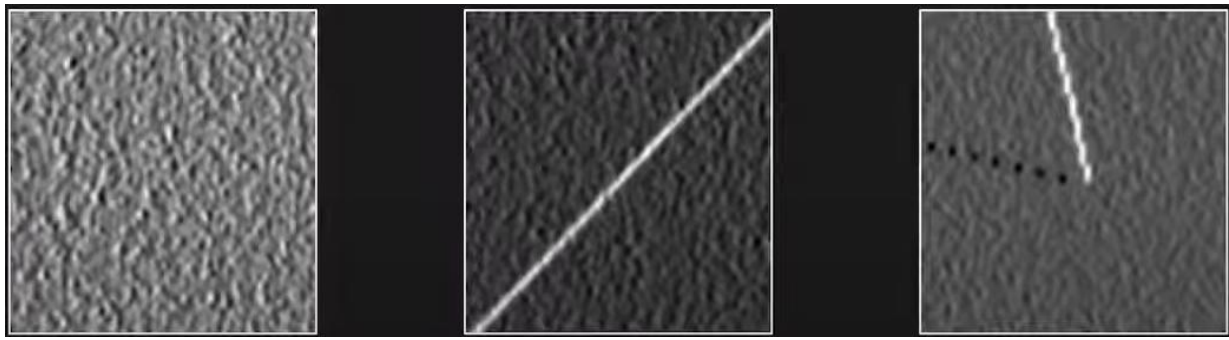
Corners

- A **corner** is the point where **two edges meet** in an image.
- Mathematically, it's defined as a region where there are **rapid changes in image intensity in two directions** (horizontal and vertical).
- Corners are very important because they are unique and easy to detect, making them excellent **features** for computer vision tasks like tracking, matching, and recognition.

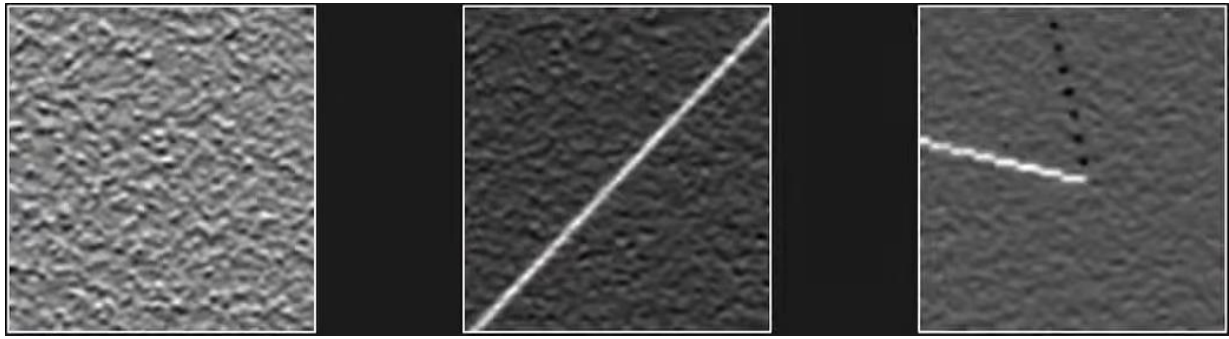
Image Gradients

I

$$I_x = \frac{\partial I}{\partial x}$$

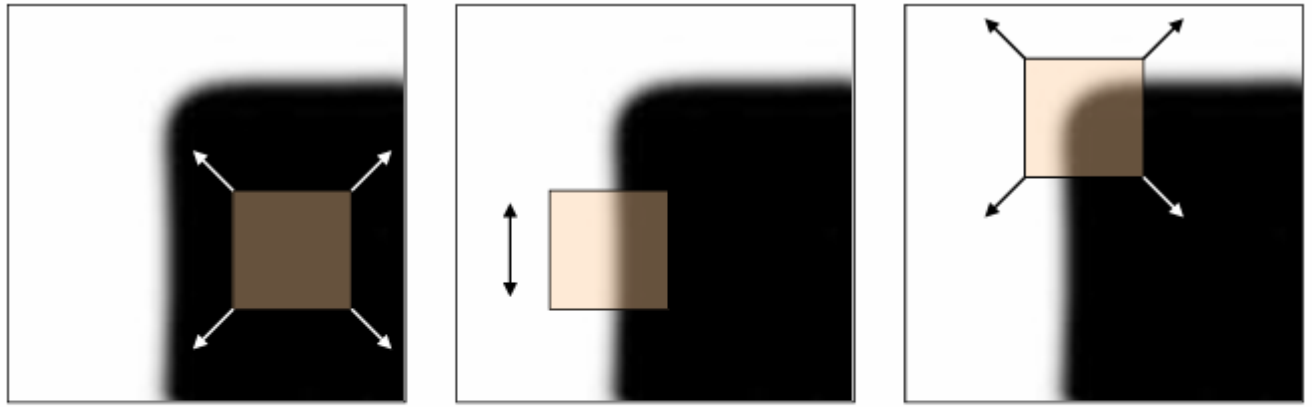


$$I_y = \frac{\partial I}{\partial y}$$



Corner Detection: Basic Idea

We should easily recognize the corner by looking through a small window
Shifting a window in any direction should give a large change in intensity



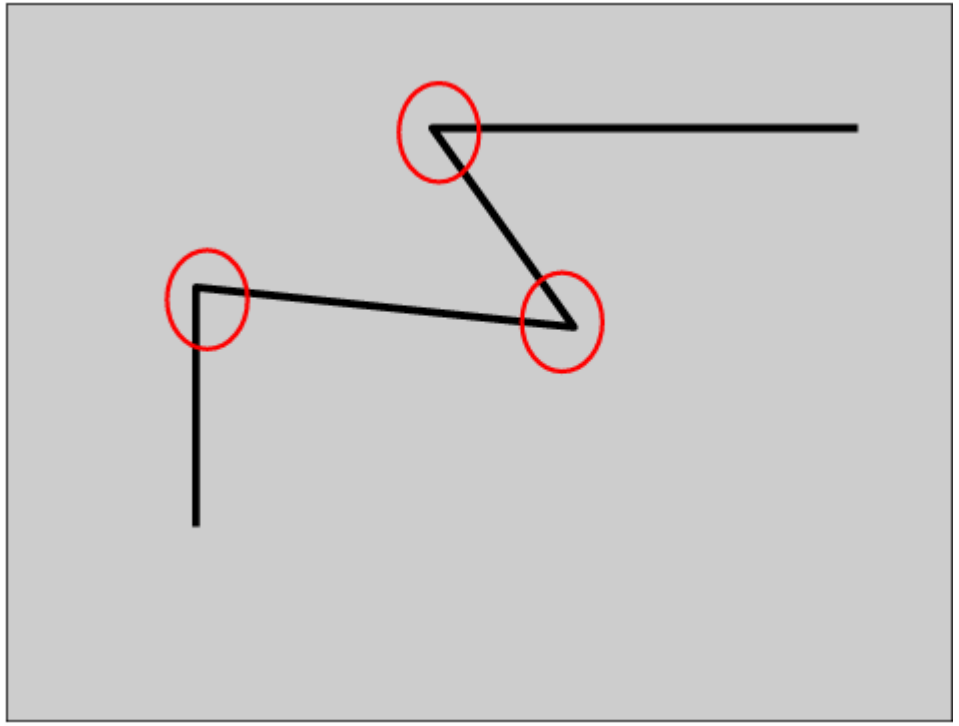
"flat" region:
no change in
all directions

"edge":
no change
along the edge
direction

"corner":
significant
change in all
directions

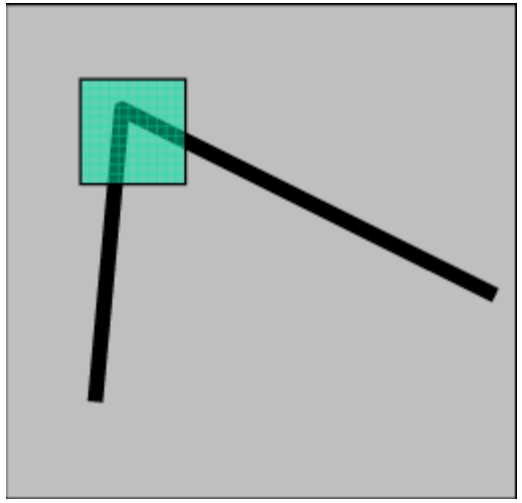
Harris Corner Detector

C. Harris and M.Stephens. "A Combined Corner and Edge Detector.“ Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988

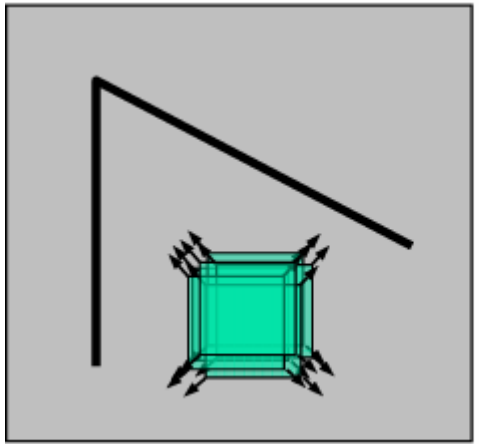


Harris Corner Detector

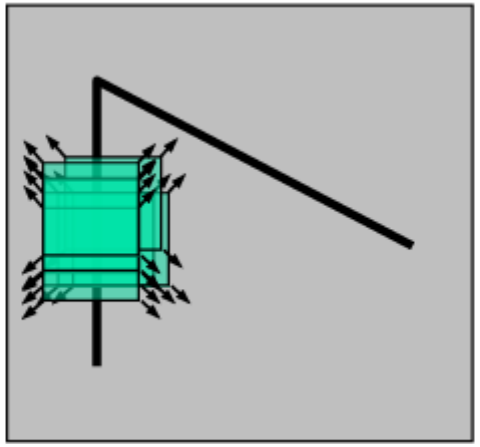
- Corner point can be recognized in a window
- Shifting a window in any direction should give a large change in intensity



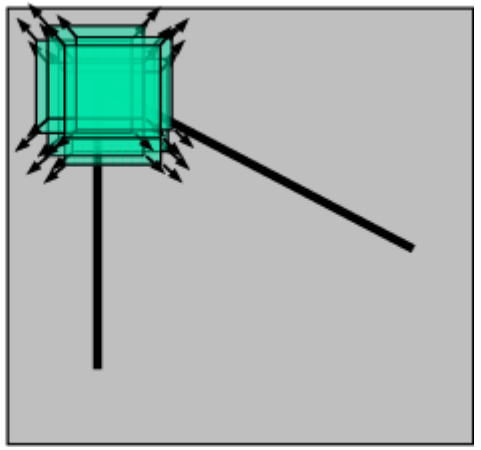
Basic Idea



“flat” region: no change in all directions



“edge”: no change along the edge direction

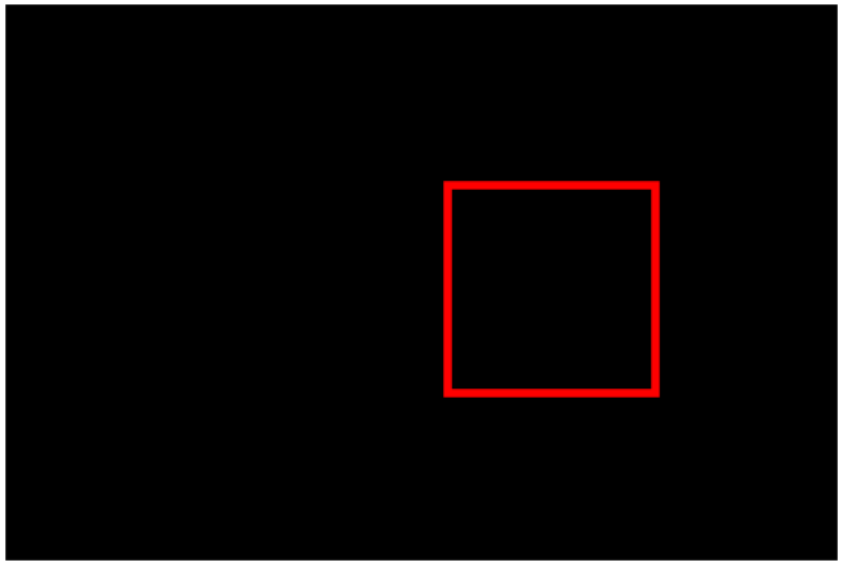


“corner”: significant change in all directions

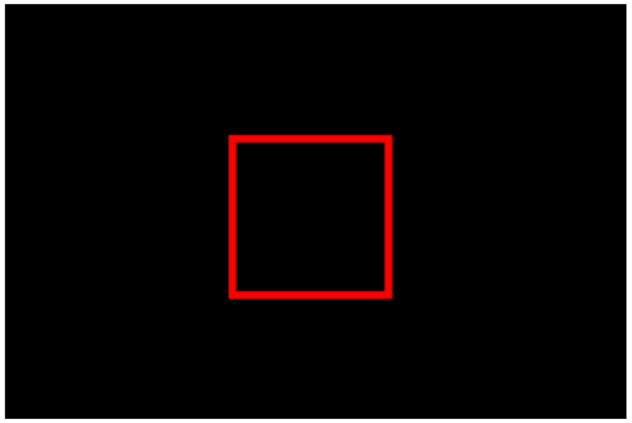
The Aperture Problem

- When observing motion through a small aperture (like a small window), only the component of motion **perpendicular to an edge** can be detected.
- Motion **parallel to the edge** is ambiguous and cannot be determined from that local view alone.

The Aperture Problem



The Aperture Problem



Corner Detection: Mathematics

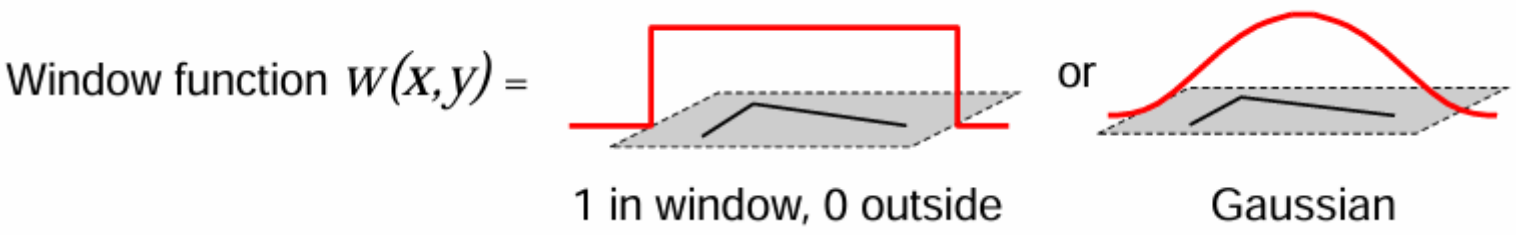
Change in appearance of window $w(x, y)$ for the shift $[u, v]$:

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

Window function

Shifted Intensity

Intensity



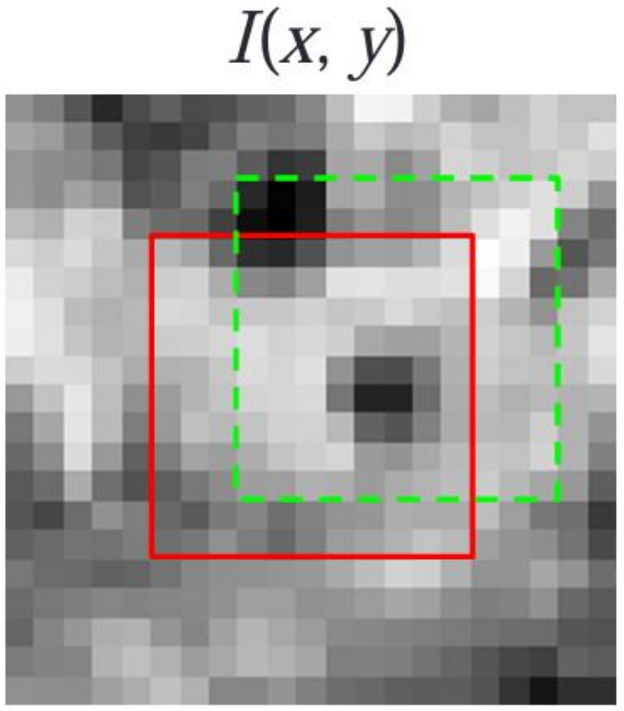
w determines weights of each pixel

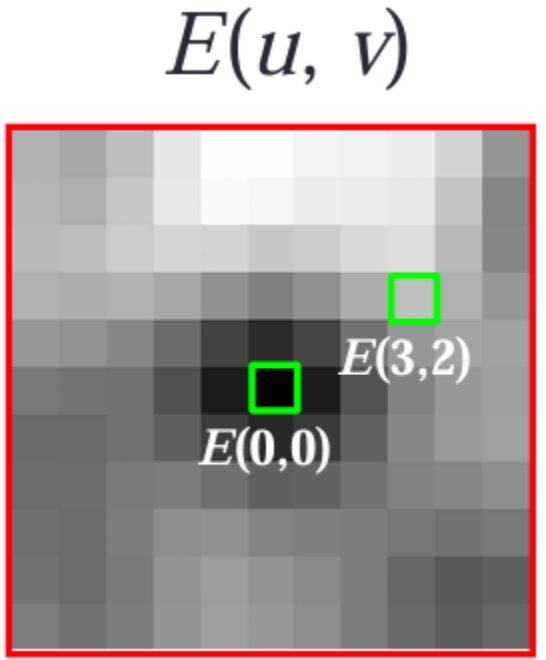
Corner Detection: Mathematics

- Suppose we have a window (small image) $w(x, y)$ placed at pixel (x, y) of input image
- If we move window by (u, v) then change in appearance of window $w(x, y)$ for the shift (u, v)

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

- **If the image region is flat:** shifting the window produces almost no change $\rightarrow E(u, v) \approx 0$
- **If the region is an edge:** shifting along the edge $\rightarrow$ small change; shifting perpendicular $\rightarrow$ large change
- **If the region is a corner:** shifting in any direction $\rightarrow$ large change $\rightarrow$ high $E(u, v)$





Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)\, h(k,l)$$

f *is image and* h *is kernel*

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$f \otimes h = f_1h_1 + f_2h_2 + f_3h_3 +$$
$$f_4h_4 + f_5h_5 + f_6h_6 +$$
$$f_7h_7 + f_8h_8 + f_9h_9$$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)\, h(k,l)$$

Cross correlation

- A measure of similarity between **two different signals (or images)** as one is shifted relative to the other.
- Used for Template matching in images.

$$f \otimes f = \sum_k \sum_l f(k,l)\, f(k,l)$$

Auto correlation

- A special case of cross-correlation where the **signal is compared with itself**.
- Used for detecting repeating patterns.

Correlation Vs SSD

- **Sum of Squared Differences (SSD)**
- We define SSD between an image patch $f(k,l)$ and a template $h(k,l)$ as:

$$SSD = \sum_k \sum_l (f(k,l) - h(k,l))^2$$

- This measures how different the template is from the image patch.
- The **smaller the SSD**, the more similar the two patches are.

$$SSD = \sum_k \sum_l \left(f(k,l)^2 - 2f(k,l)h(k,l) + h(k,l)^2\right)$$

Correlation Vs SSD…

$$SSD = \sum_k \sum_l f(k,l)^2 + \sum_k \sum_l h(k,l)^2 - 2 \sum_k \sum_l f(k,l)h(k,l)$$

- The first two terms do **not depend on correlation** (they are constants for a given patch & template).

$$\sum_k \sum_l f(k,l)^2$$

$$\sum_k \sum_l h(k,l)^2$$

- The only variable term is:

$$-2 \sum_k \sum_l f(k,l)h(k,l)$$

Correlation Vs SSD…

The last term is exactly the **cross-correlation** between f and h.

$$\sum_k \sum_l f(k,l)h(k,l)$$

Minimizing SSD is the same as **Maximizing correlation** (since the -2 coefficient makes them inversely related).

Correlation Vs SSD…

$$SSD = \sum_k \sum_l f^2 + \sum_k \sum_l h^2 - 2.Correlation\,(f, h)$$

- If you minimize SSD, you are effectively maximizing correlation.
- That's why correlation is often used as a similarity measure (easier than computing all SSD terms).

- SSD → measures dissimilarity (smaller = better match).
- Correlation → measures similarity (larger = better match).
- Both are equivalent, just expressed differently.

Corner Detection by Auto Correlation

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

This is computationally very expensive

Taylor Series

- The **Taylor series expansion** is a way to approximate a smooth function around a point.
- A function $f(x)$ can be represented at point a in terms of its derivatives.

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \frac{1}{3!}f'''(a)(x-a)^3 + \cdots$$

- The first term $f(a) = function\ value\ at\ a$
- The linear term $f'(a)(x-a) = local\ slope\ (rate\ of\ change)$.
- Higher-order terms capture curvature and more subtle variations.

In computer vision, we often use **first-order Taylor expansion** to approximate intensity changes in images.

Taylor Series

Express $I(x+u, y+v)$ at (x, y):

$$I(x+u, y+v) = I(x, y) + I_x(x+u-x) + I_y(y+v-y)$$

$$I(x+u, y+v) = I(x, y) + I_x u + I_y v$$

- I_x, I_y are the **image gradients** (changes in intensity along x and y directions).

Optical Flow Constraint Equation

- Suppose an image intensity is $I(x, y, t)$
- If a pixel moves slightly by (u, v) in time, intensity should remain constant:

$$I(x + u, y + v, t + 1) = I(x, y, t)$$

- Apply **first-order Taylor expansion**:

$$I(x + u, y + v, t + 1) = I(x, y, t) + I_x u + I_y v + I_t$$

$$I_x u + I_y v + I_t = 0$$

- This is the **optical flow constraint equation**.
- Taylor series is used to linearize image changes for motion analysis.

Auto-Correlation

- To find **good features to track** (like corners), we analyze how much the intensity changes under small shifts (u, v).
- We measure the **auto-correlation function** (second-moment matrix):

$$E(u,v) = \sum_{x,y} [I(x+u, y+v) - I(x,y)]^2$$

- Apply **Taylor expansion** for $I(x+u, y+v)$:

$$I(x+u, y+v) = I(x,y) + I_x u + I_y v$$

Substitute into $E(u,v)$

$$E(u,v) = \sum_{x,y} \left[I(x,y) + I_x u + I_y v \ - I(x,y)\right]^2$$

Auto-Correlation

$$E(u,v) = \sum_{x,y} \left[I_x u + I_y v\right]^2$$

$$E(u,v) = \sum_{x,y} \left[\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix}\right]^2$$

$$E(u,v) = \sum_{x,y} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) = \sum_{x,y} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Matrix M in the middle is the **auto-correlation matrix** (a.k.a. **second-moment matrix** or **structure tensor**).

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$E(u,v)$ tells us how much intensity changes when we shift by (u,v). The eigenvalues of M tell us the local structure:

- Both small → flat region.
- One large, one small → edge.
- Both large → corner.

Example-1

Input Image

0	0	1	4	9
1	0	5	7	11
1	4	9	12	16
3	8	11	14	16
8	10	15	16	20

$\frac{d}{dx}$

-1	0	1

$\frac{d}{dy}$

-1
0
1

I_x

	4	7	6	
	8	8	7	
	8	6	5	

$$I_x^2 = 4^2 + 7^2 + 6^2 + 8^2 + 8^2 + 7^2 + 8^2 + 6^2 + 5^2 = 403$$

I_y

	4	8	8	
	8	6	7	
	6	6	4	

$$I_y^2 = 4^2 + 8^2 + 8^2 + 8^2 + 6^2 + 7^2 + 6^2 + 6^2 + 4^2 = 381$$

$$I_x I_y = 4 \times 4 + 7 \times 8 + 6 \times 8 + 8 \times 8 + 8 \times 6 + 7 \times 7 + 8 \times 6 + 6 \times 6 + 5 \times 4 = 385$$

Example-1

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} 403 & 385 \\ 385 & 381 \end{bmatrix}$$

- Determinant $|M| = \lambda_1 \lambda_2$ (product of eigenvalues)
- $\text{trace}(M) = \lambda_1 + \lambda_2$ (sum of eigenvalues)
- $k =$ empirical constant (usually between $0.04 - 0.06$)

$$R = |M| - k.(trace(M))^2$$
$$R = (403 \times 381 - 385 \times 385) - 0.04.(403 + 381)^2$$
$$R = (153543 - 148225) - 0.04.(784)^2$$
$$R = (5318) - 0.04.(614656)$$
$$R = (5318) - 24586.24$$
$$R = -19268.24$$

- Interpretation:
  - If R is **large positive** → corner
  - If R is **negative** → edge
  - If R is **small** → flat region

We need to find whether pixel at location (3, 3) is a corner or not?
Gradients are provided

Example 2 – Step 1

$$E(u,v) = [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) = [u \quad v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

I_x

1	3	6	5
12	2	11	3
9	2	4	10
3	7	1	2
7	2	7	8

I_x^2

1	9	36	25
144	4	121	9
81	4	16	100
9	49	1	4
49	4	49	64

$I_x I_y$

7	9	6	35
108	4	55	15
63	2	8	20
15	63	3	6
28	6	7	16

I_y

7	3	1	7
9	2	5	5
7	1	2	2
5	9	3	3
4	3	1	2

I_y^2

49	9	1	49
81	4	25	25
49	1	4	4
25	81	9	9
16	9	1	4

Example 2 – Step 2

Select the kernel size w (3, 3)

W		
(x-1, y-1)	(x-1, y)	(x-1, y+1)
(x, y-1)	(x, y)	(x, y+1)
(x+1, y-1)	(x+1, y)	(x+1, y+1)

Example 2 – Step 3

I_x^2

1	9	36	25
144	4	121	9
81	4	16	100
9	49	1	4
49	4	49	64

I_y^2

49	9	1	49
81	4	25	25
49	1	4	4
25	81	9	9
16	9	1	4

$I_x I_y$

7	9	6	35
108	4	55	15
63	2	8	20
15	63	3	6
28	6	7	16

Calculate

$$\sum I_x^2 = 4 + 121 + 9 + 4 + 16 + 100 + 49 + 1 + 4 = 308$$

$$\sum I_y^2 = 4 + 25 + 25 + 1 + 4 + 4 + 81 + 9 + 9 = 162$$

$$\sum I_x I_y = 4 + 55 + 15 + 2 + 8 + 20 + 63 + 3 + 6 = 176$$

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} 308 & 176 \\ 176 & 162 \end{bmatrix}$$

Example 2 – Step 4

Find points with large response

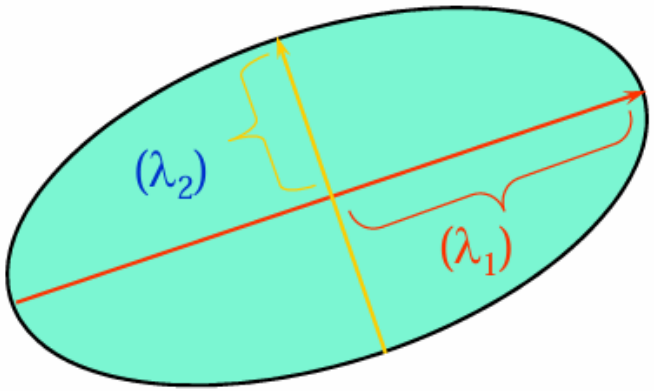
$$R = |M| - k.(trace(M))^2$$
$$R = \lambda_1\lambda_2 - k.(\lambda_1 + \lambda_2)^2$$

- Interpretation:
  - If R is **large positive** $\rightarrow$ corner
  - If R is **negative** $\rightarrow$ edge
  - If R is **small** $\rightarrow$ flat region

$$M = \begin{bmatrix} 308 & 176 \\ 176 & 162 \end{bmatrix}$$
$$|M| = (308 \times 162) - (176 \times 176) = 49896 - 30976 = 18920$$
$$trace(M) = 308 + 162 = 470$$

$$R = 18920 - 0.04.(470)^2$$
$$R = 18920 - 8836 = 10084$$

Mathematics of Harris Detector



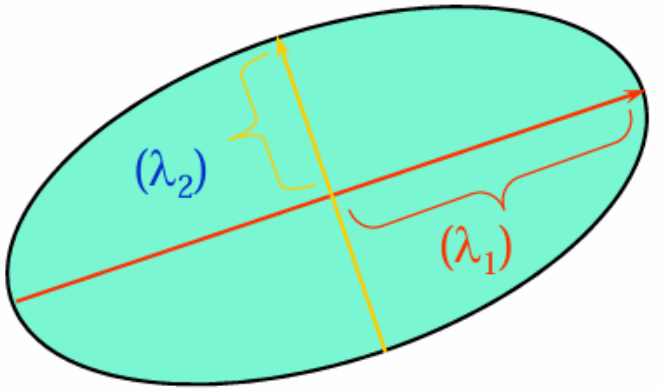
$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$E(u,v) = [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

- $E(u,v)$ is equation of an ellipse
- This is a **quadratic form**, which is the general equation of an ellipse (or ellipse-like curve) in 2D.
- The "shape" of the ellipse depends on M.
- If you draw **all** (u,v) **for which** $E(u,v)$ **is constant**, the result is an ellipse.

- Let λ_1, λ_2 be the eigenvalues of M.
- Eigenvalues tell us **how much** $E(u,v)$ **grows in a certain direction**.
- Eigenvectors give the **directions** of the ellipse's principal axes.
- So:
- The **long axis** of the ellipse corresponds to the larger eigenvalue.
- The **short axis** corresponds to the smaller eigenvalue.
- That's why the ellipse in the picture is labeled with λ_1 (long axis) and λ_2 (short axis).

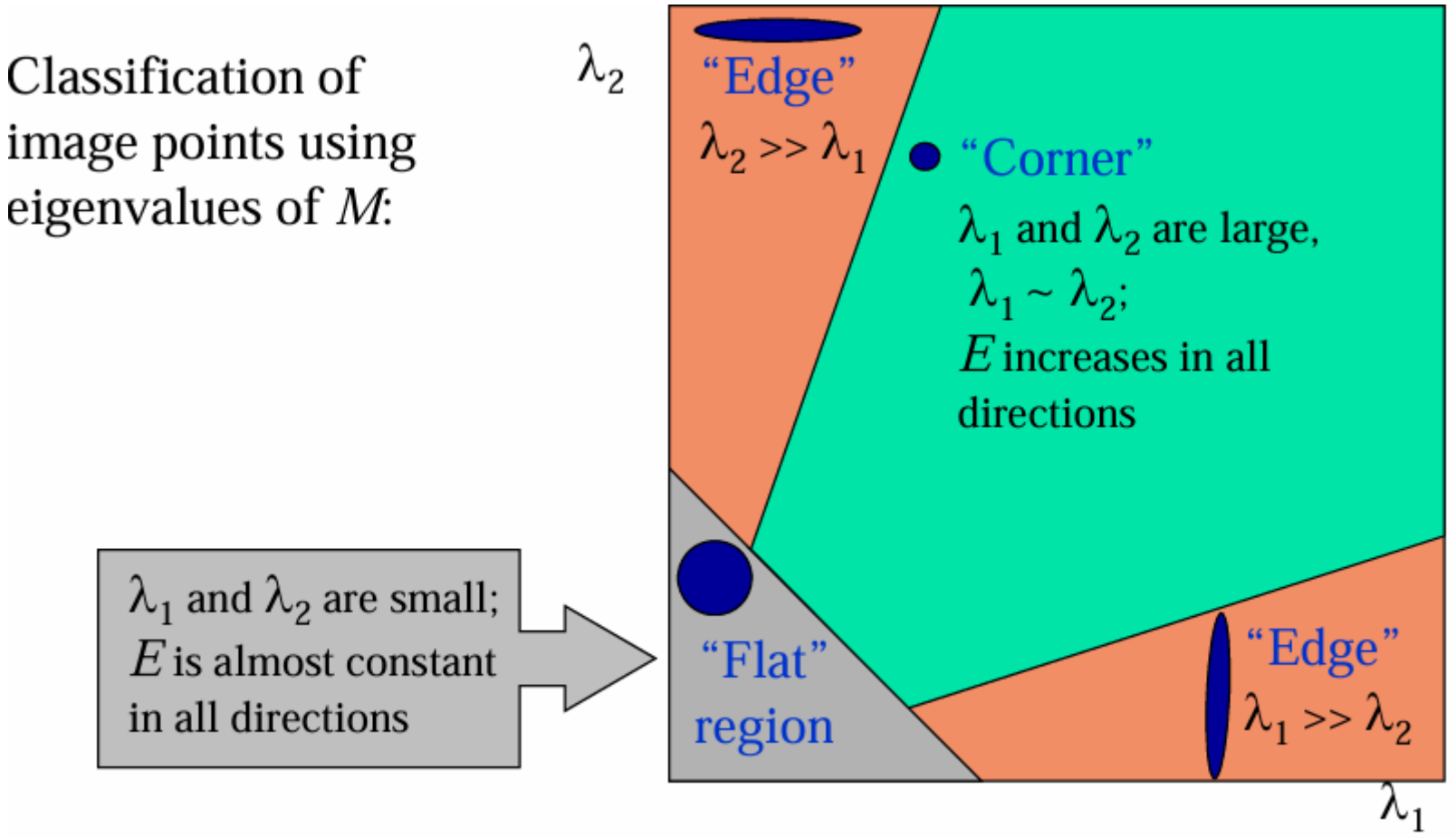
Mathematics of Harris Detector



Interpretation in terms of image features

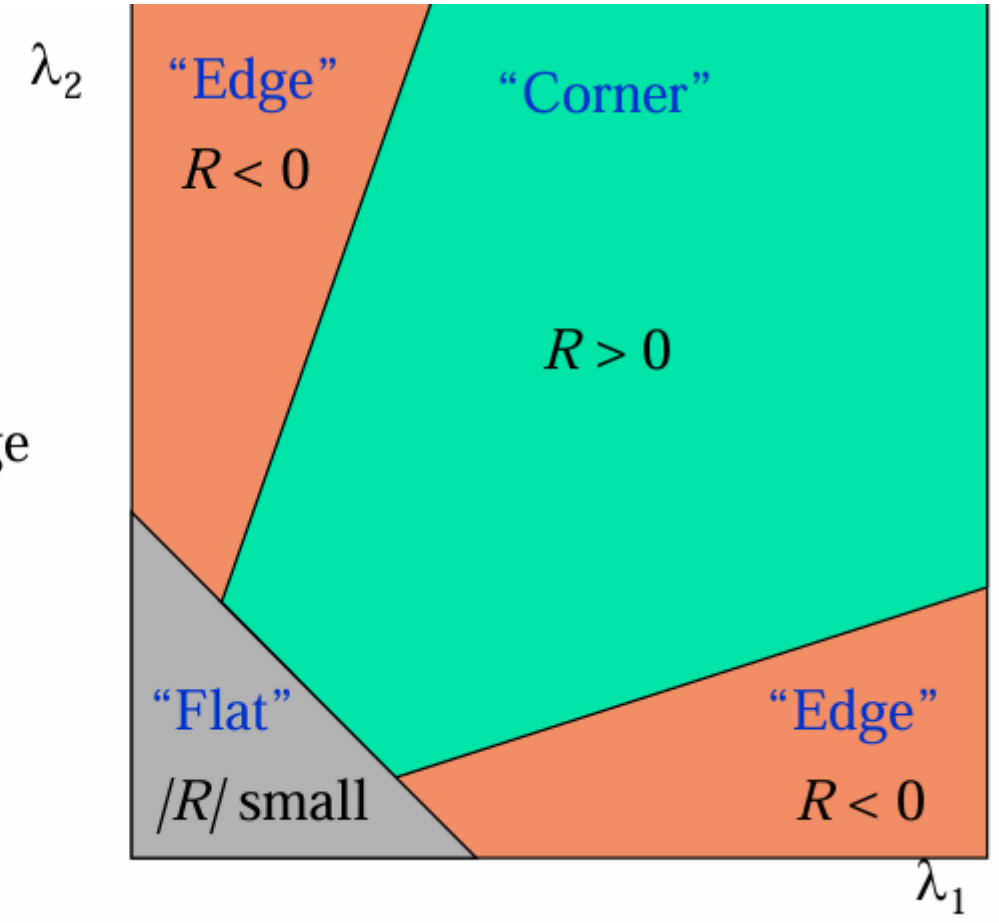
- **Flat region**: both λ_1, λ_2 small $\rightarrow$ ellipse is almost a circle of tiny radius $\rightarrow$ little change in any direction.
- **Edge**: one eigenvalue large, the other small $\rightarrow$ ellipse is very elongated $\rightarrow$ strong change in one direction only.
- **Corner**: both eigenvalues large $\rightarrow$ ellipse is big in both directions $\rightarrow$ strong change in all directions.

Mathematics of Harris Detector



Mathematics of Harris Detector

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Eigen Vectors and Eigenvalues

- If A is a square matrix, an **eigenvector** x is a special non-zero vector such that

$$Ax = \lambda x$$

Here:

  - x= eigenvector
  - λ= eigenvalue (a scalar)
- Multiplying the matrix A by x does **not change its direction**, only scales it by λ.

Finding Eigenvalues

$$Ax = \lambda x \Rightarrow (A - \lambda I)x = 0$$

- This is a **homogeneous linear system**.
- For a non-trivial solution ($x \neq 0$), the matrix $(A - \lambda I)$ must be **singular** (not invertible).
- Condition for singularity: $\det(A - \lambda I) = 0$
- This is the **characteristic equation**.
- Solving it gives us the eigenvalues λ.

Finding Eigenvectors

- Once you know λ, substitute it back into:

$$(A - \lambda I)x = 0$$

- Solve this system to find the corresponding eigenvector(s).

- Eigenvalues tell us the "strength" of scaling in special directions.
- Eigenvectors tell us the "directions" that remain unchanged (except for scaling).
- In **Harris detector context**: the eigenvalues of the second-moment matrix (M) tell us how much intensity changes in different directions.

Numerical Example

$$A = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix}$$

A is **upper triangular**, so its eigenvalues are the diagonal entries:
$\lambda_1 = 7, \lambda_2 = 3, \lambda_3 = -1$

Eigenvectors

For $\lambda_1 = 7$: solve $(A - 7I)x = 0$

$$A - 7I = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} - 7\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A - 7I = \begin{bmatrix} -8 & 2 & 0 \\ 0 & -4 & 4 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{cases} -8x + 2y = 0 \\ -4y + 4z = 0 \end{cases} \Rightarrow y = 4x,\ z = y = 4x$$

Choose $x = 1 \Rightarrow (\mathrm{x}, \mathrm{y}, \mathrm{z}) = (1, 4, 4)$. $\quad x_1 = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix}$

Numerical Example

For $\lambda_2 = 3$: solve $(A - 3I)x = 0$

$$A - 3I = \begin{bmatrix} -4 & 2 & 0 \\ 0 & 0 & 4 \\ 0 & 0 & 4 \end{bmatrix}, \quad \begin{cases} -4x + 2y = 0 \\ 4z = 0 \end{cases} \Rightarrow y = 2x,\ z = 0$$

Choose $x = 1 \Rightarrow (x, y, z) = (1, 2, 0)$ $\qquad x_2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

For $\lambda_3 = -1$: solve $(A + I)x = 0$

$$A + I = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 8 \end{bmatrix}, \quad \begin{cases} 2y = 0 \\ 4y + 4z = 0 \Rightarrow y = 0,\ z = 0,\ x \text{ free} \\ 8z = 0 \end{cases}$$

Choose $x = 1 \Rightarrow (x, y, z) = (1, 0, 0)$ $\qquad x_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Eigenvalues

$$\det(A-\lambda I)=0$$

$$\det\left(\begin{bmatrix}-1&2&0\\0&3&4\\0&0&7\end{bmatrix}-\lambda\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)=0$$

$$\det\left(\begin{bmatrix}-1-\lambda&2&0\\0&3-\lambda&4\\0&0&7-\lambda\end{bmatrix}\right)=0$$

$$(-1-\lambda)((3-\lambda)(7-\lambda)-0)=0$$

$$(-1-\lambda)(3-\lambda)(7-\lambda)=0$$

$$\lambda=-1,\quad \lambda=3,\quad \lambda=7$$

$\lambda=-1$

$$(A-\lambda I)x=0$$

$$\left(\begin{bmatrix}-1&2&0\\0&3&4\\0&0&7\end{bmatrix}+\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}0&2&0\\0&4&4\\0&0&8\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$0+2x_2+0=0$$

$$0+4x_2+4x_3=0$$

$$0+0+8x_3=0$$

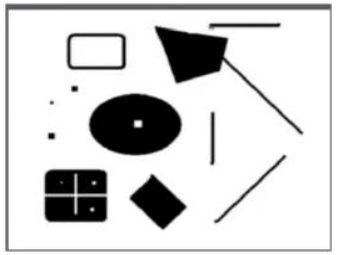
$$x_1=1,\quad x_2=0,\quad x_3=0$$

$$\mathbf{x_1}=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

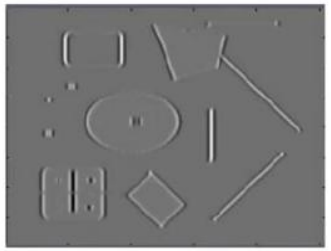
Corners as Distinctive Interest Points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives
(averaged in neighborhood of a point)

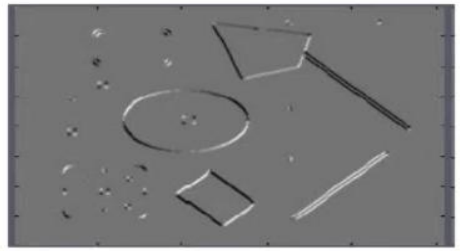


Notation:



$$I_x \Leftrightarrow \frac{\partial I}{\partial x}$$

$$I_y \Leftrightarrow \frac{\partial I}{\partial y}$$



$$I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

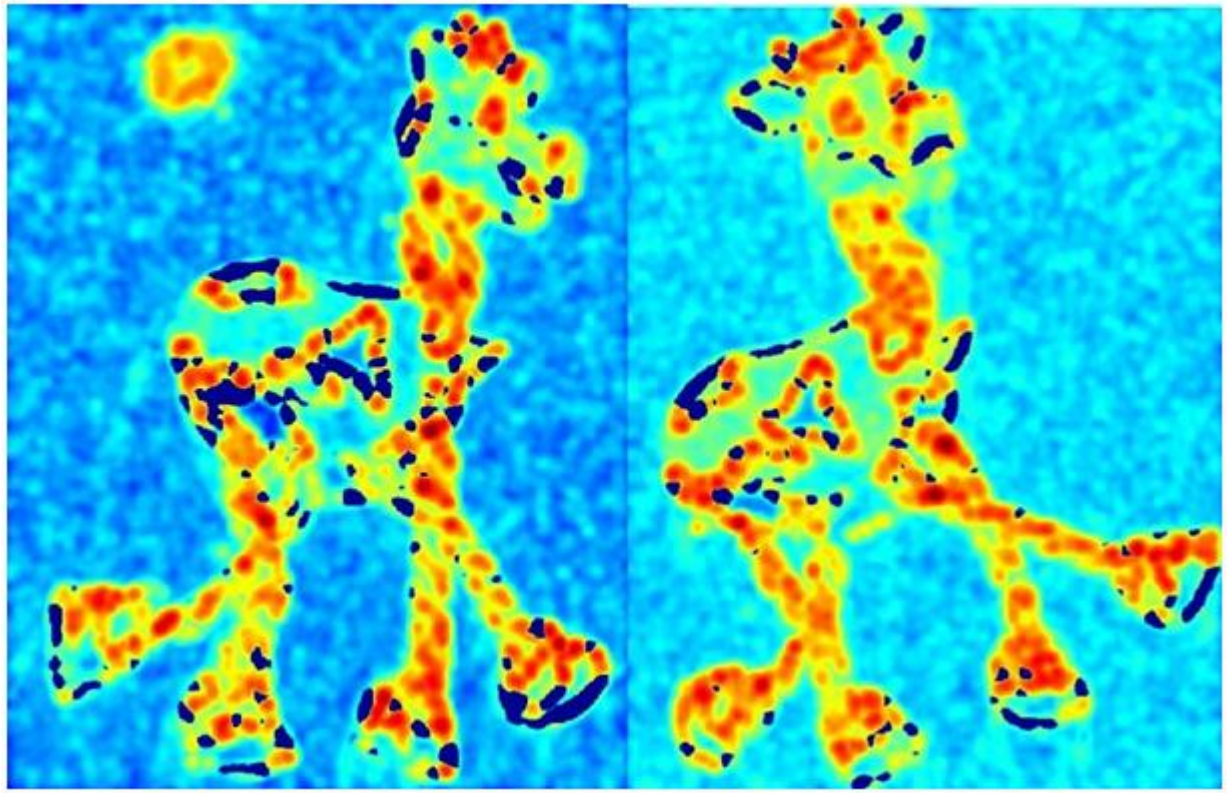
Harris Corner Detector Algorithm

- Compute horizontal and vertical derivatives of image I_x and I_y.
- Compute three images corresponding to three terms in matrix *M*.
- Convolve these three images with a larger Gaussian (window).
- Compute scalar cornerness value using one of the *R* measures.
- Find local maxima above some threshold as detected interest points.

Example: Harris Detector

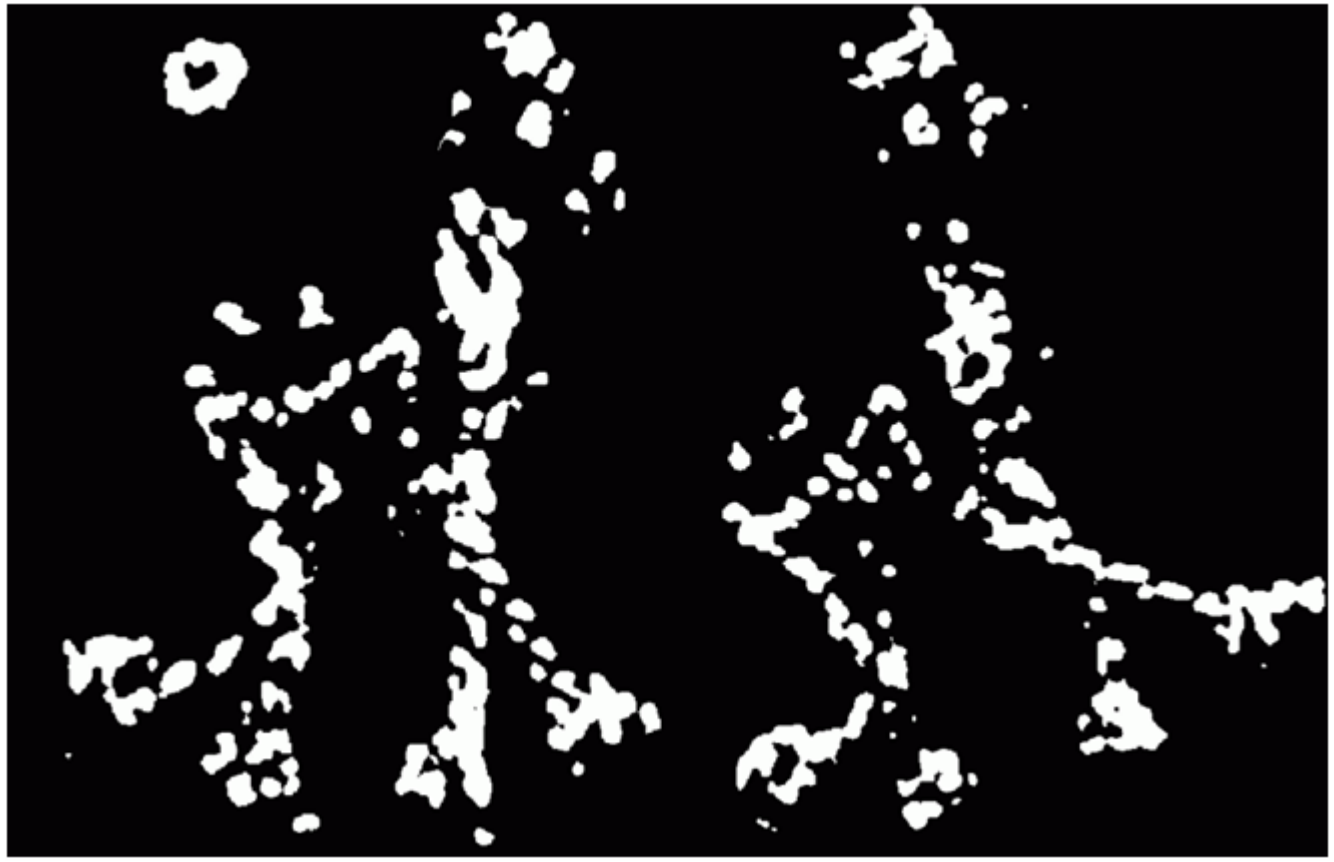
Example: Harris Detector

Compute corner response R



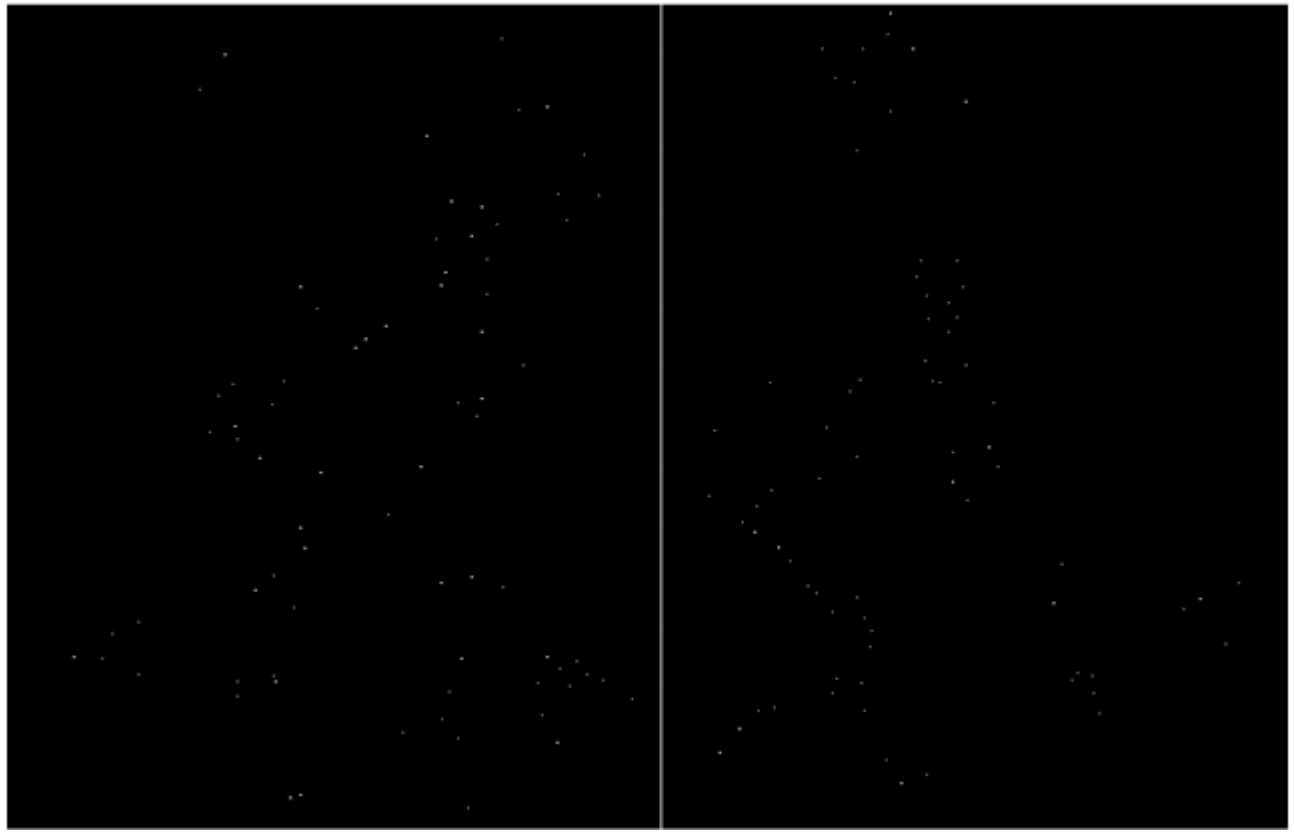
Harris Detector: Steps

Find points with large corner response:
R>threshold



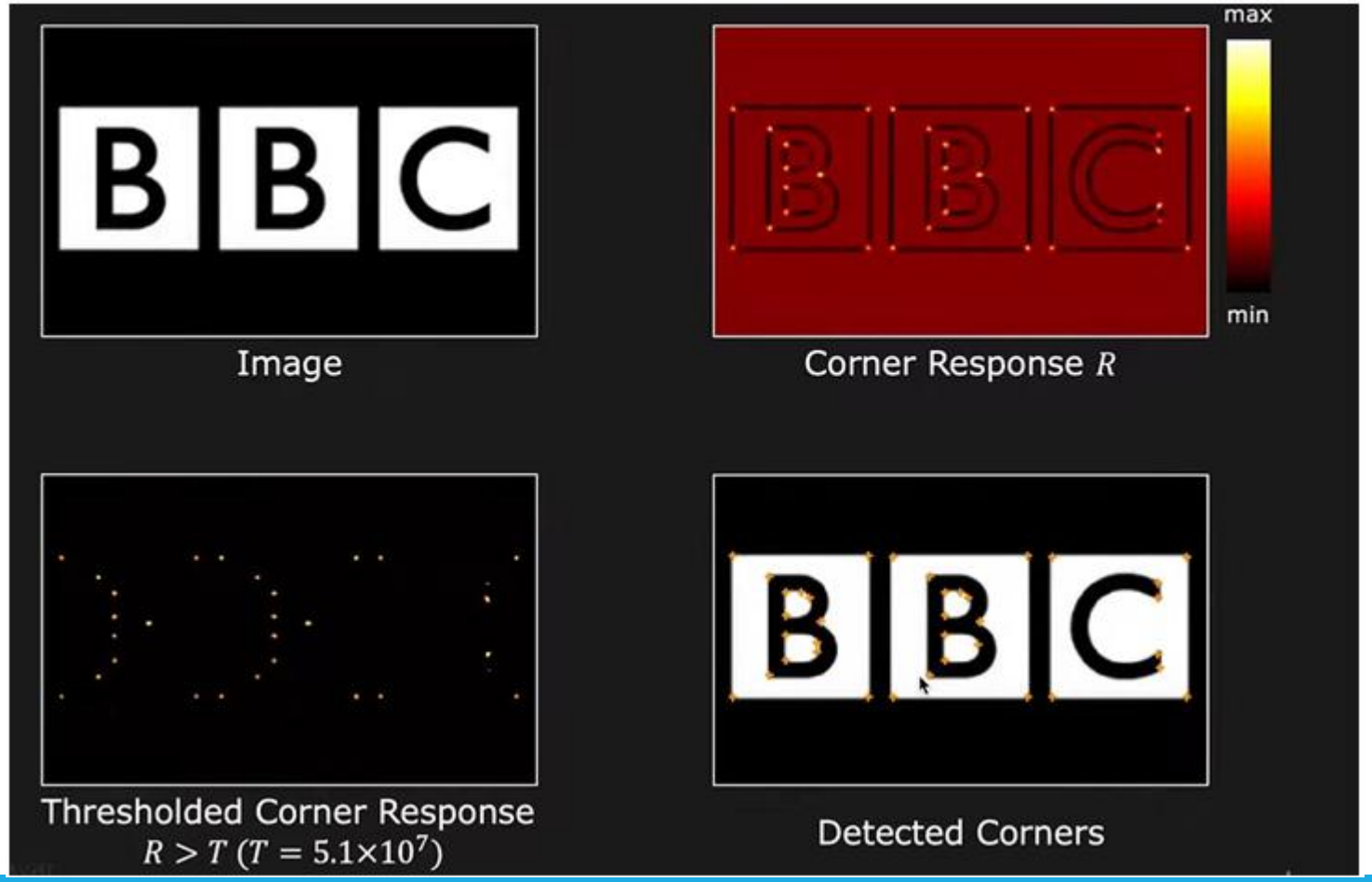
Example: Harris Detector

Take only the points of local maxima of R

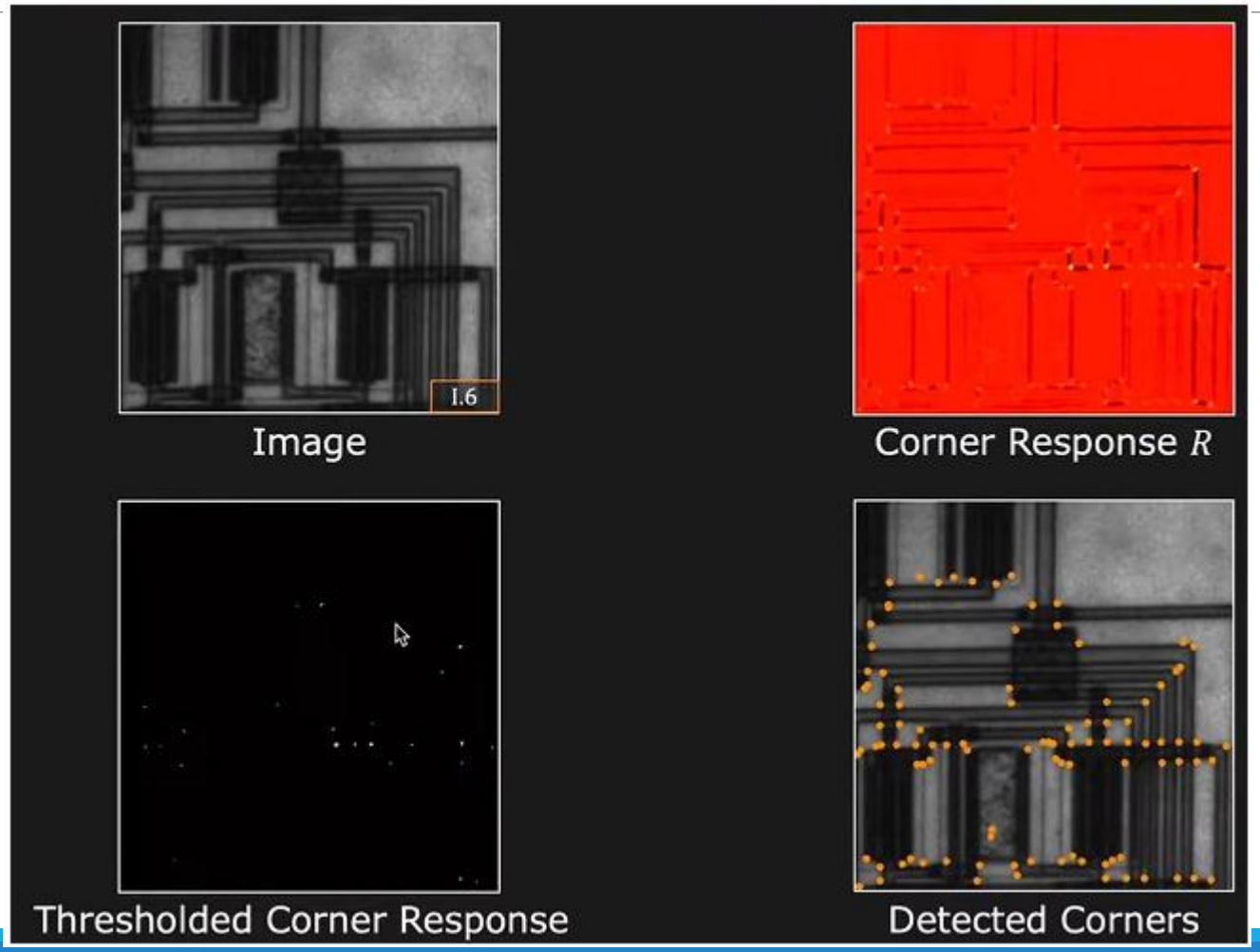


Example: Harris Detector

Example: Harris Detector



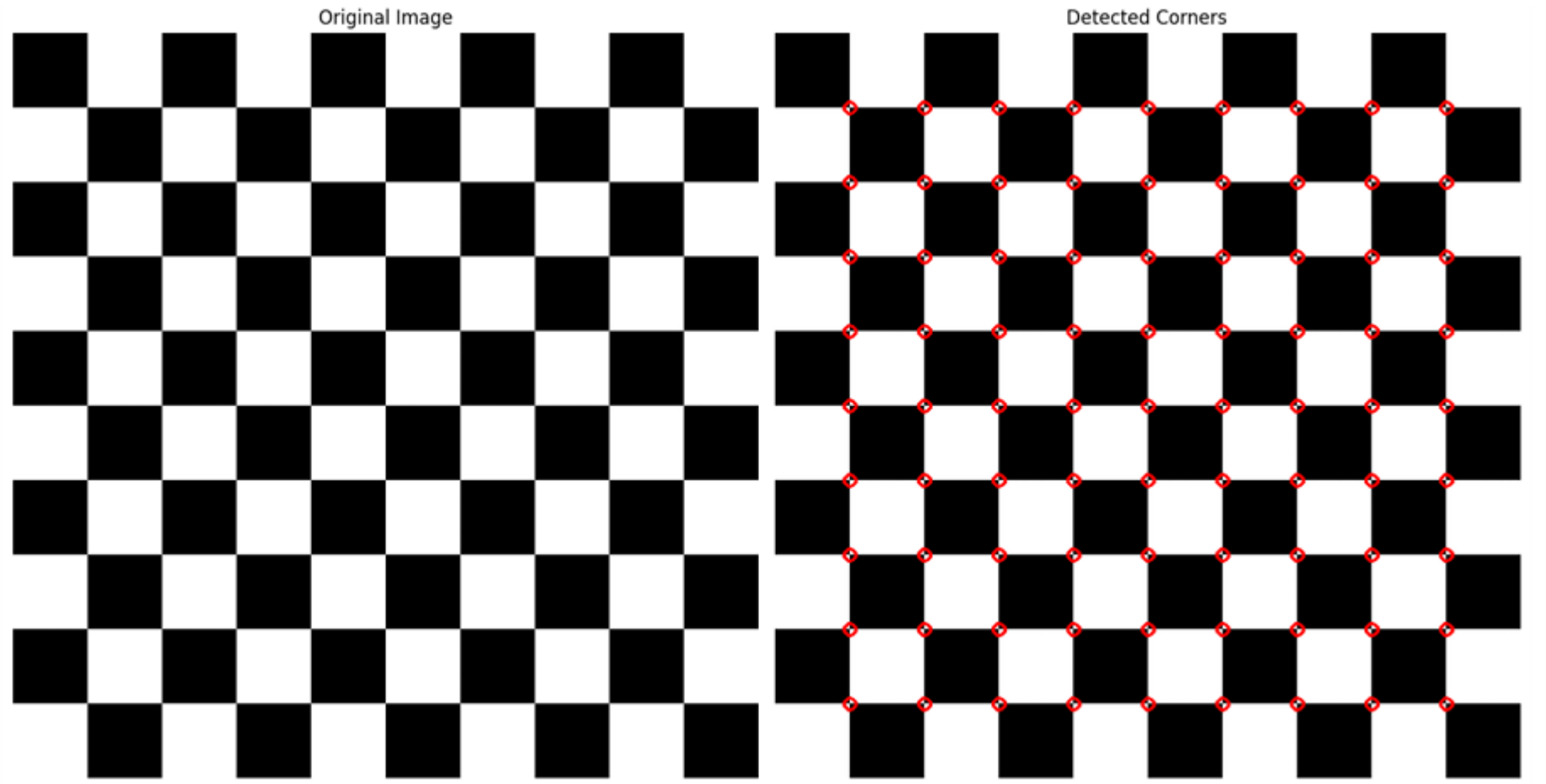
Example: Harris Detector



Harris Corner Detection Demo Code

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Read the image
image = cv2.imread('checker.png', cv2.IMREAD_GRAYSCALE)
image_with_corners = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
# Perform Harris corner detection
dst = cv2.cornerHarris(image, 2, 3, 0.04)
# Threshold the corner response
dst_thresh = 0.1 * dst.max()
corner_image = np.zeros_like(image)
corner_image[dst > dst_thresh] = 255
# Find centroids of corners
ret, labels, stats, centroids = cv2.connectedComponentsWithStats(corner_image)
# Draw red circles at corner locations
for centroid in centroids[1:]:
    x, y = int(centroid[0]), int(centroid[1])
    cv2.circle(image_with_corners, (x, y), 5, (255, 0, 0), 2)
```

Harris Corner Detection Demo Code



How Invariant are Harris Corners?

Location invariant to Photometric Transformations

Covariant to Geometric Transformations

Invariance: Image is transformed, and corner locations do not change

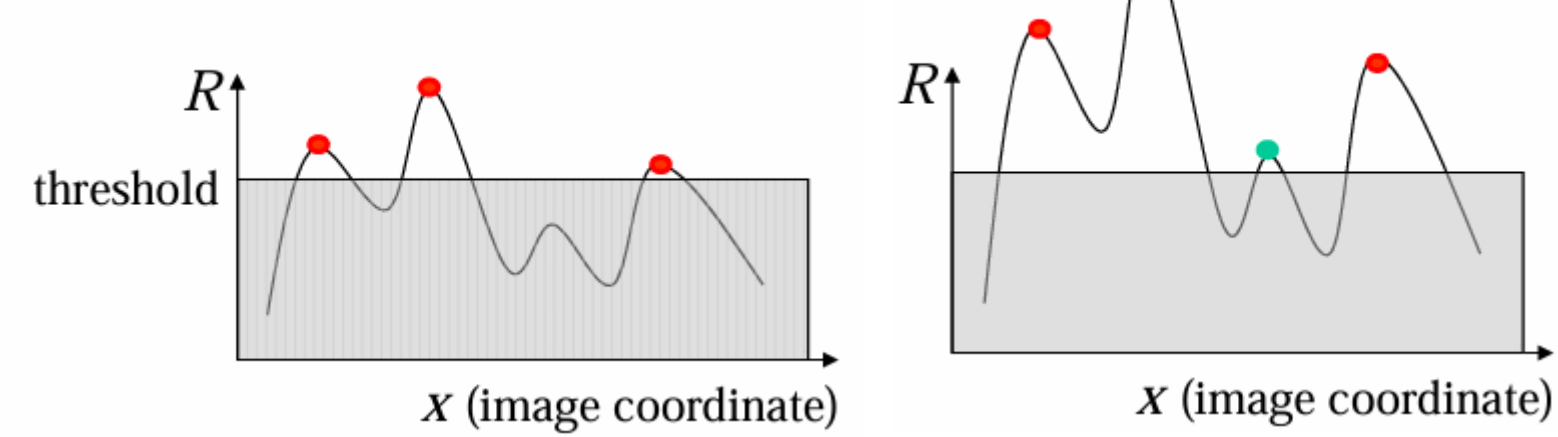
Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations

Affine intensity change

$$I \rightarrow a\,I + b$$

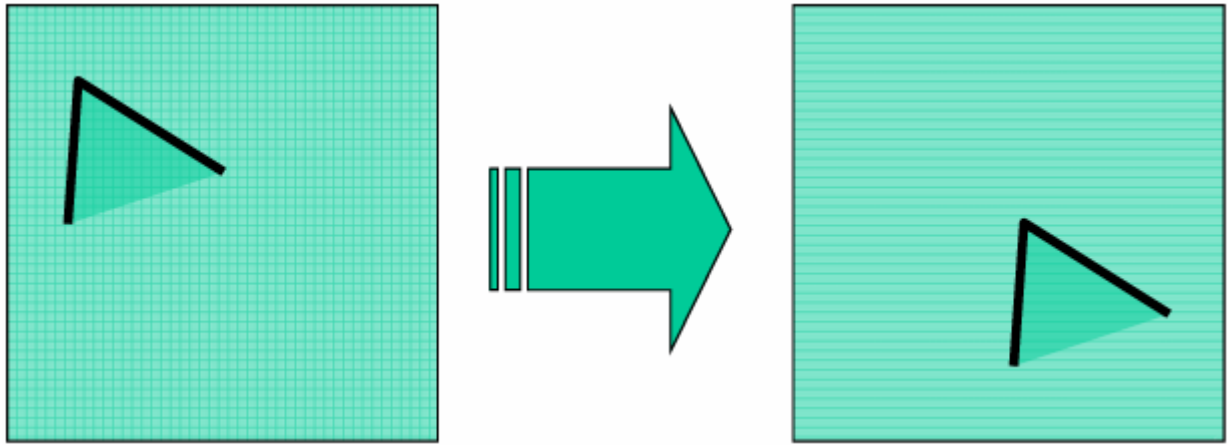
Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

Intensity scaling: $I \rightarrow a\,I$



Partially invariant to affine intensity change

Image translation

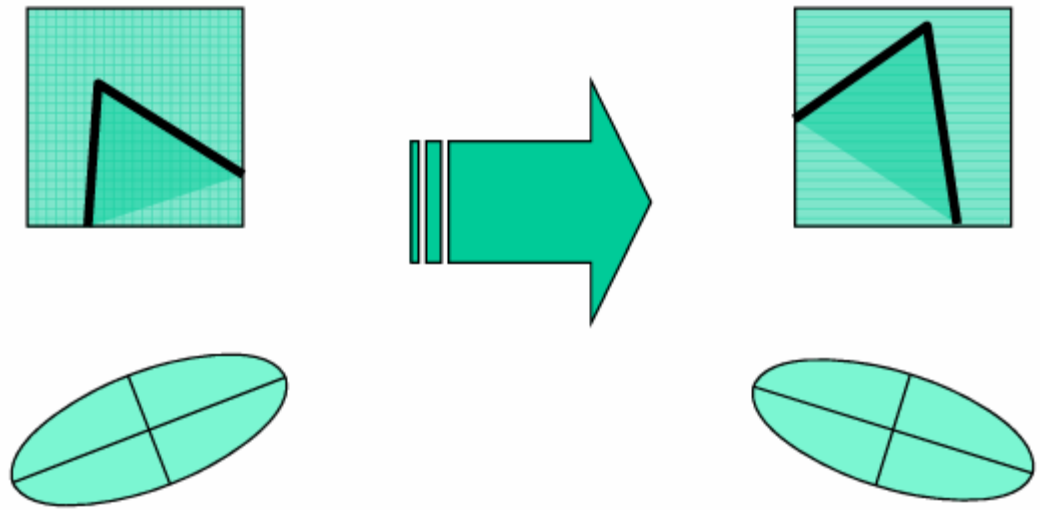


Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

Image rotation

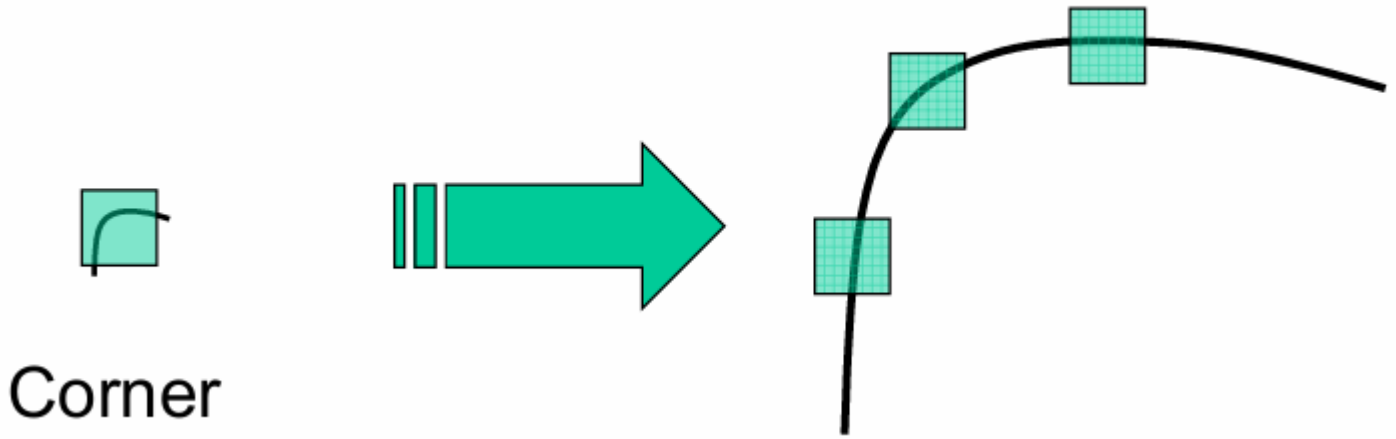
Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same



Corner location is covariant w.r.t. rotation

Scaling

All points will be classified as edges



Corner location is not covariant to scaling!

Shi-Tomasi Corner Detector

Shi-Tomasi Corner Detector

- Often called the Good Features to Track algorithm, Shi-Tomasi is an efficient and fast method that detects prominent corners well-suited for tracking across image frames.
- It focuses on identifying the best corners to track, which is beneficial for real-time applications.

Shi-Tomasi Corner Detector

Shi–Tomasi Corner Detector (1994)

Improvement: Instead of using the Harris "R" score, Shi–Tomasi looks directly at the eigenvalues $\lambda_1, \lambda_2$of M.

Criterion:

A good corner exists if both eigenvalues are large (so intensity varies in both directions).

Advantage: More robust, doesn't require tuning of k.

Used in practice: It's the basis of cv2.goodFeaturesToTrack in OpenCV.

Thank you