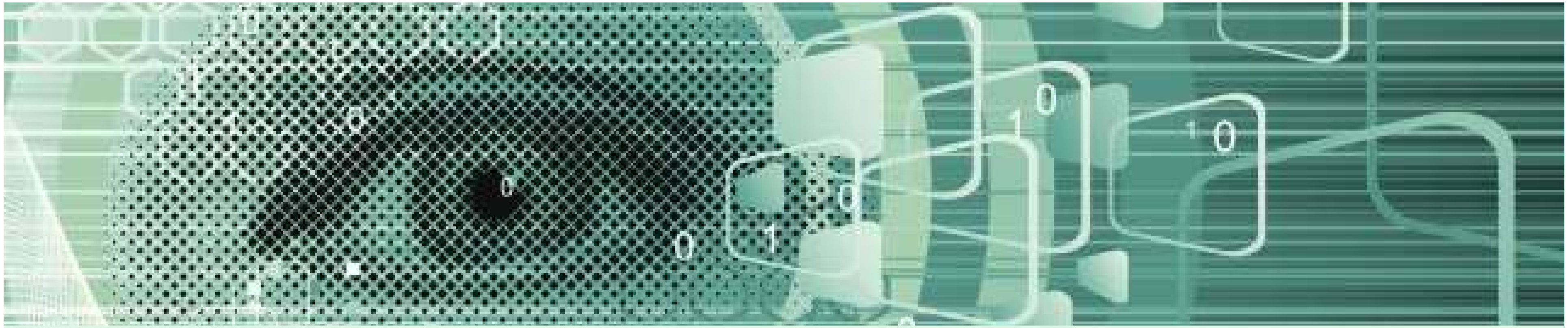


# Lecture 03- Feature Detection and Tracking

## EE382-Visual localization & Perception

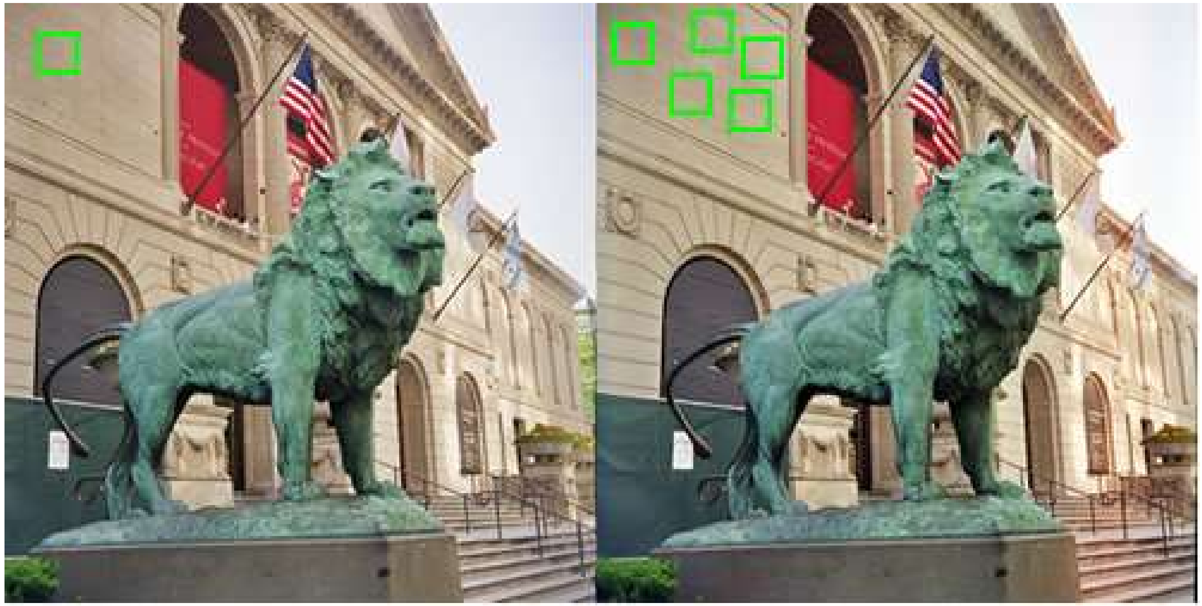
Danping Zou @Shanghai Jiao Tong University

# Outline

- Feature points
  - Feature detector & descriptor
  - Feature matching & tracking
- Optical flow
  - Harris / Kanade-Tomasi detector
  - Lucas-Kanade tracker
  - Coarse-to-fine
- FAST detector
- BRIEF/ORB descriptor
- Line features
  - Hough transform
  - LSD detector

# Feature points: What are they?

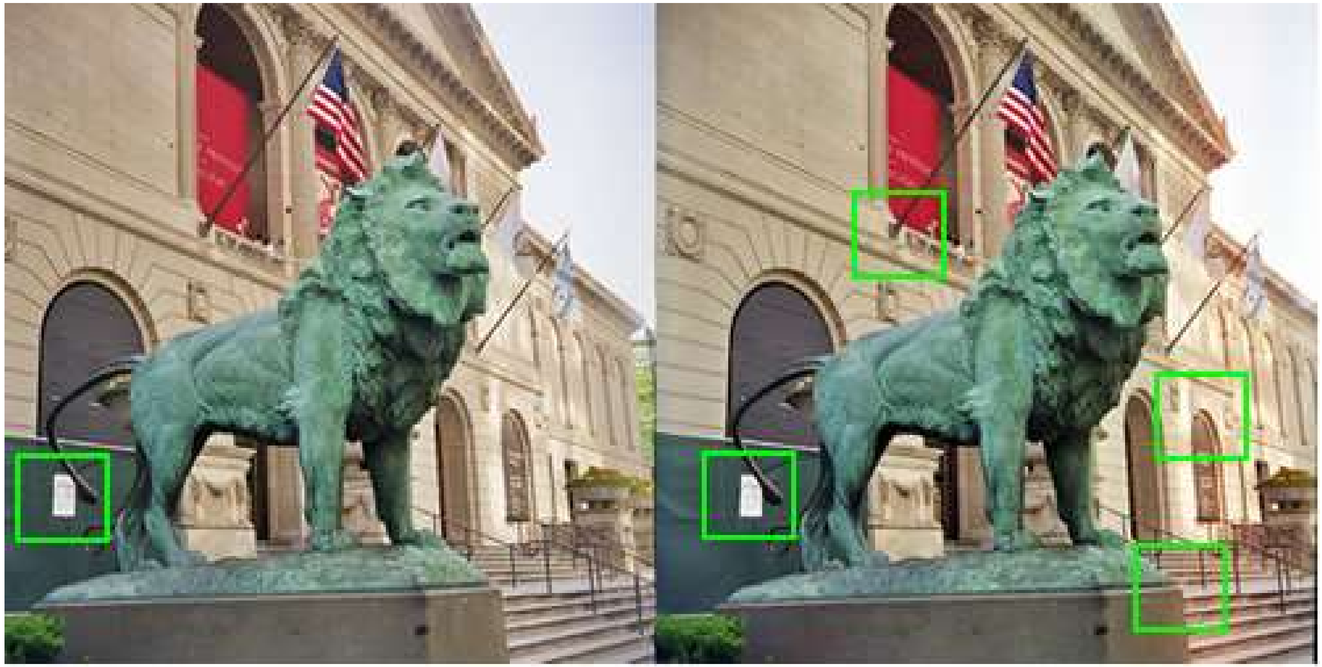
- Feature points are the backbone of a lost computer vision algorithms. But what is a feature point?
  - A feature is defined as an "**interesting**" part of an image.



Feature points do not locate at positon with unrecognizable appearance.

# Features points: What are they?

- Intuitively, a patch with ***unique appearance***



But there could be a few of such patches in an image.

# Features points: What are they?

- Three properties a good feature point should have:
  - 1) **Stable** : The feature point can be detected repeatedly in different views.
  - 2) **Recognizable** : It has rich local information that make it easy to recognized.
  - 3) **Accurate** : Feature point can be located in sub-pixel accuracy.

# Feature detector

- Extract interesting points from the image
- List of feature detectors

| Corner detector |
|---|
| Harris detector |
| Kanade-Tomasi detector (Good feature to track detector) |
| FAST detector |

| Blob detector |
|---|
| Laplacian of Gaussian (LoG) detector |
| Difference of Gaussian (DoG) detector (SIFT detector) |
| MSER detector |

# Feature descriptors

- Feature descriptor – a vector describes the appearance of the feature point.
- List of feature descriptors

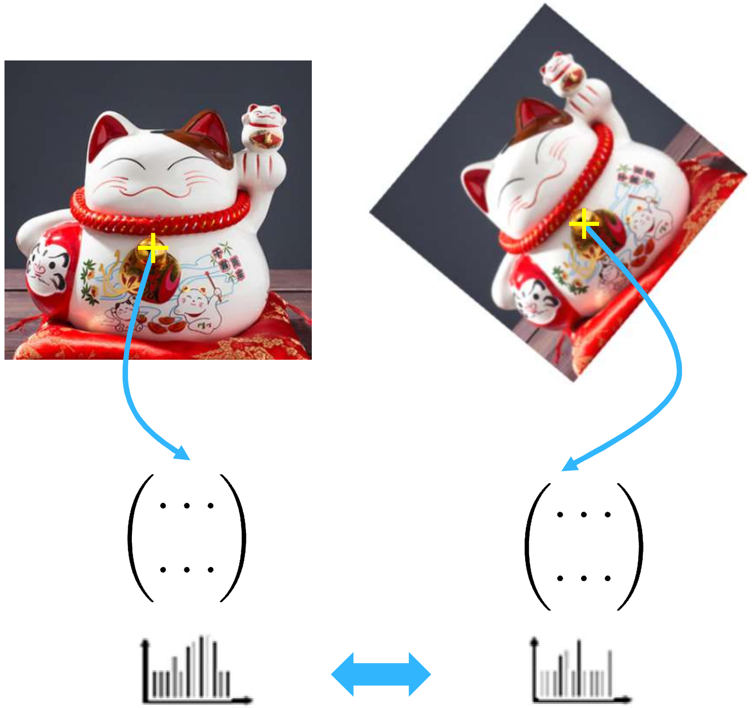
| Binary Descriptors: (suitable for real time processing) |
|---|
| Binary Robust Independent Elementary Features (**BRIEF**) |
| Oriented FAST and Rotated BRIEF (**ORB**) |
| Binary Robust Invariance Scalable Key points (**BRISK**) |
| Fast Retina Key point (**FREAK**) |

| Spectra Descriptors: (Robust to viewpoint change or illumination change but slow!) |
|---|
| Scale Invariant Feature Transform (**SIFT**) |
| Speed-Up Robust Feature (**SURF**) |
| Histogram of Oriented Gradient (**HOG**) |

# Feature matching

- Finding the corresponding feature points is the key problem in computer vision

- General approach
  - Find the interest points
  - Compute a local descriptor from the local region around the point
  - Match the feature descriptor



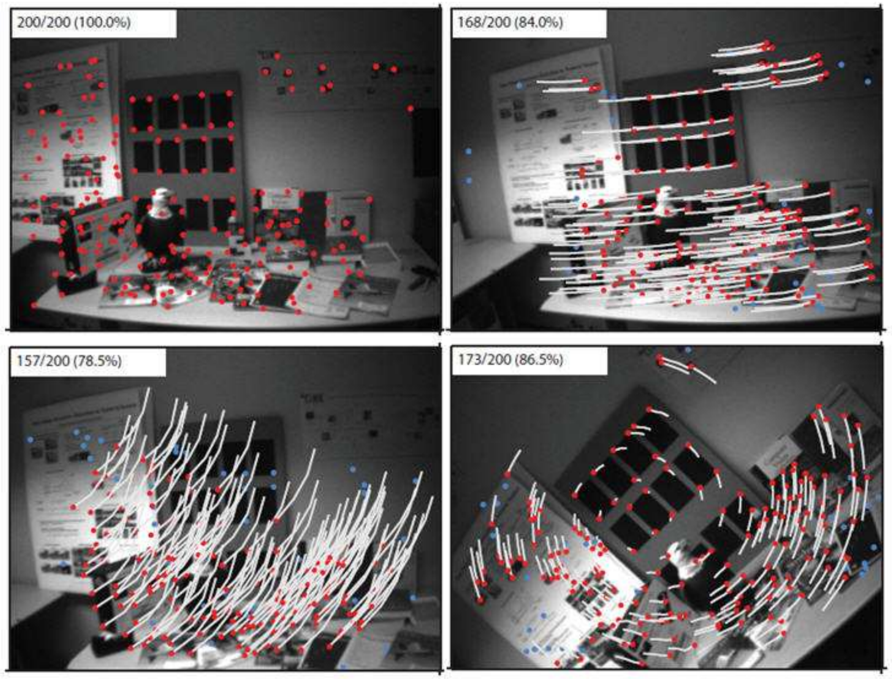
# Feature matching

- Matching by tracking
  - Find the interest points
  - Track the points using the local patch information
  - Requires small changes between two images

Optical flow

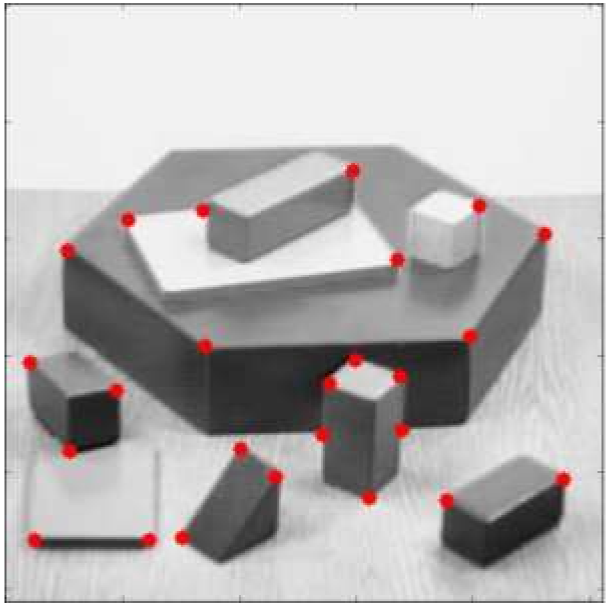
# Optical flow

- Here we introduce a classic optical flow algorithm:
  - **K**anade-**L**ucas-**T**omasi (**KLT**) feature tracker
    - Sparse flow estimation
    - **Feature extraction** – Kanade-Tomasi features
    - **Tracking** – Lucas-Kanade optical flow

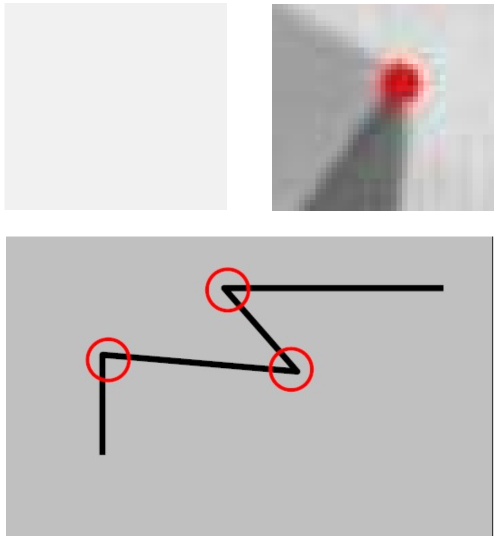


# Feature detection

- We introduce a classic feature point - Harris corner



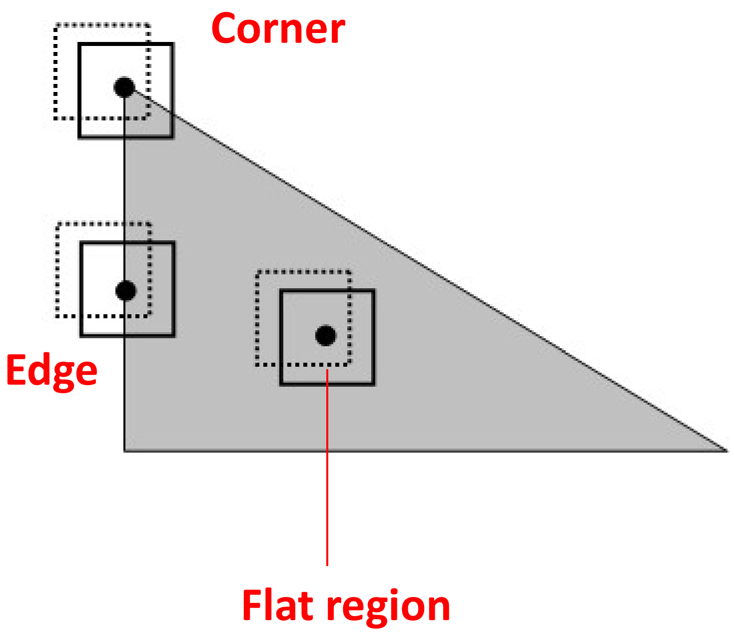
How do we define corner mathematically?



*Harris, Chris, and Mike Stephens. "A combined corner and edge detector."Alvey vision conference. Vol. 15. 1988.*

# Feature detection

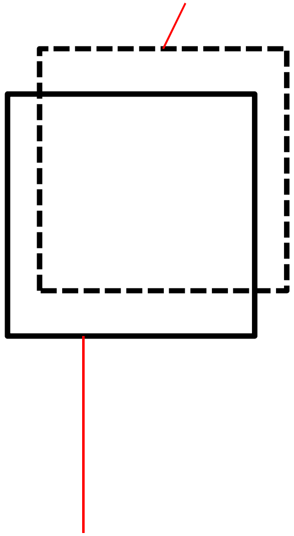
- Basic idea :
  – move the rectangle slightly around the original position and check the changes




- **Flat region:** No change in all direction
- **Edge :** No change along the edge direction
- **Corner:** Significant changes in all directions

# Feature detection

- Comparing the original image patch and the shifted image patch can be mathematically written as

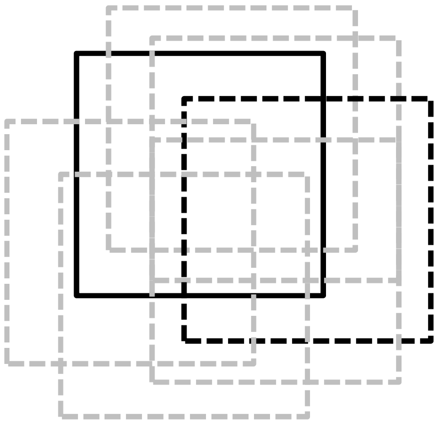



$$\underbrace{D(u,v)}_{\text{Sum of Squared Difference (SSD)}} = \sum_{u,v\in\Omega}(\underbrace{I(x,y)}_{\text{Original patch}} - \underbrace{I(x+u,y+v)}_{\text{Shifted patch}})^2$$

# Feature detection

- Maravec corner detector

Compare all nearby patches one by one to get the minimum SSD $D(u,v)$ as the corner response.



Problems:
1. high computational cost
2. Threshold of SSD is difficult to be set (depends on the image content)

Moravec H P. Obstacle avoidance and navigation in the real world by a seeing robot rover[R]. STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, **1980**.

# Feature detection

- Harris corner detector:
- Using first-order Taylor expansion, we get

$$I(x+\delta x, y+\delta y) \approx I(x,y) + \nabla I(x,y) \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = I(x,y) + [I_x, I_y] \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

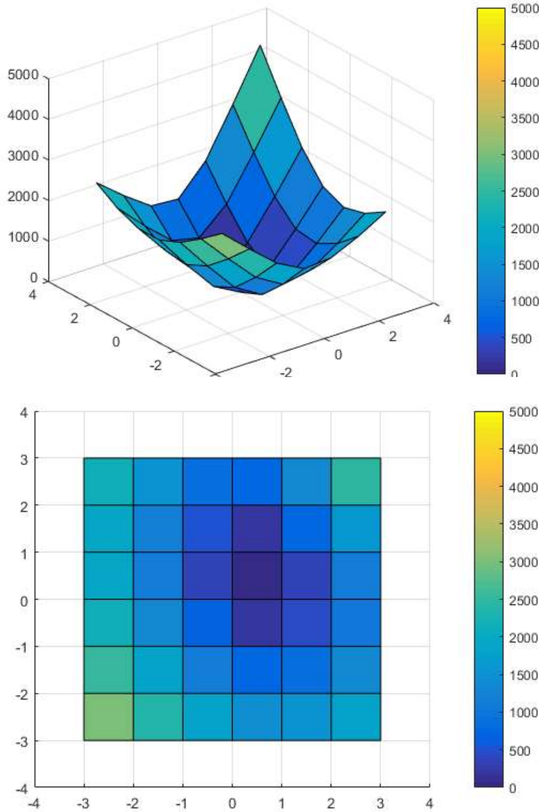
- Therefore, we can approximate the SSD function as

$$D(\delta x, \delta y) = \textstyle\sum_{u,v \in \Omega} (I(x,y) - I(x+\delta x, y+\delta y))^2$$

$$\approx [\delta x, \delta y] \begin{pmatrix} \sum_{u,v\in\Omega} I_x(\delta x, \delta y) I_x(\delta x, \delta y) & \sum_{u,v\in\Omega} I_x(\delta x, \delta y) I_y(\delta x, \delta y) \\ \sum_{u,v\in\Omega} I_x(\delta x, \delta y) I_y(\delta x, \delta y) & \sum_{u,v\in\Omega} I_y(\delta x, \delta y) I_y(\delta x, \delta y) \end{pmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

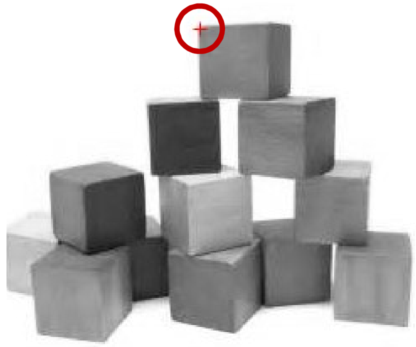
$$\boxed{D(\delta x, \delta y) = [\delta x, \delta y] M \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}}$$

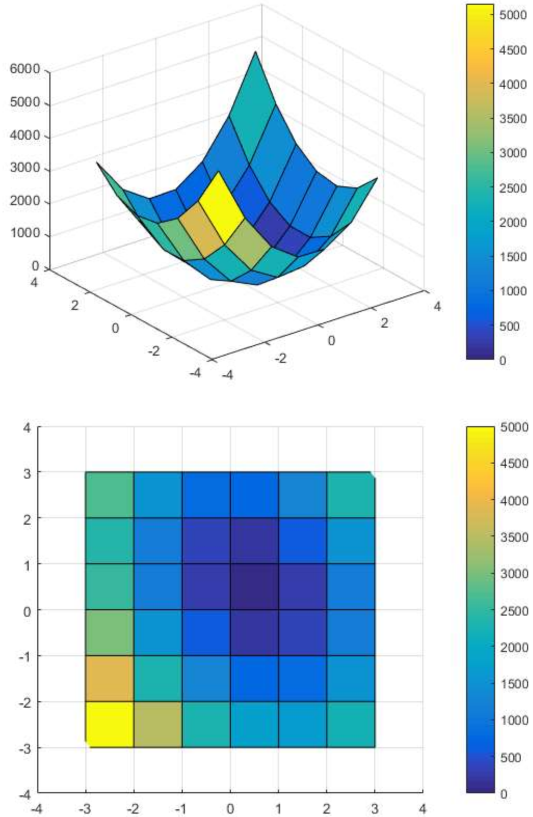
# Feature detection



Original SSD function

$$D(\delta x, \delta y) = \sum_{\delta x, \delta y \in \Omega} (I(x,y) - I(x+\delta x, y+\delta y))^2$$

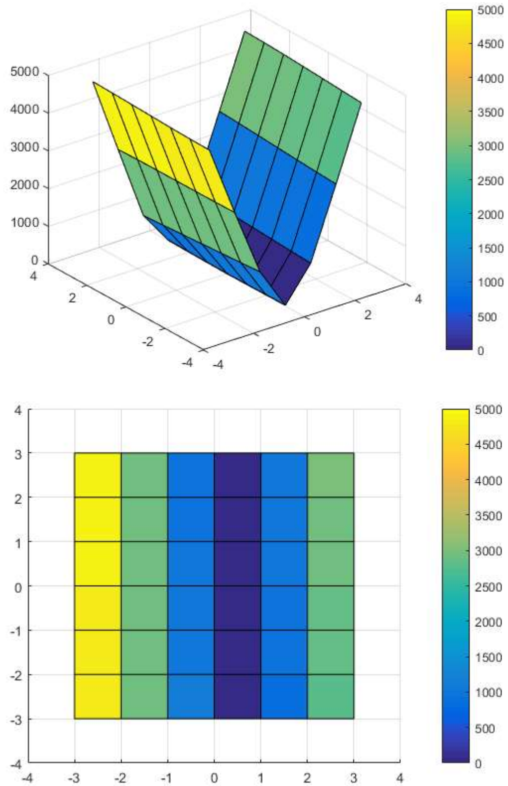


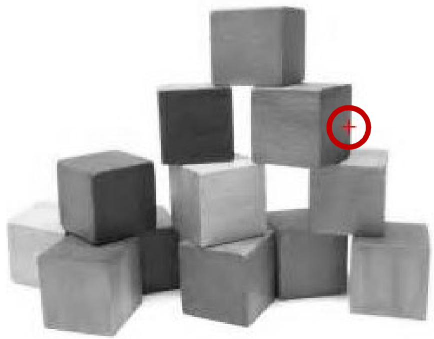


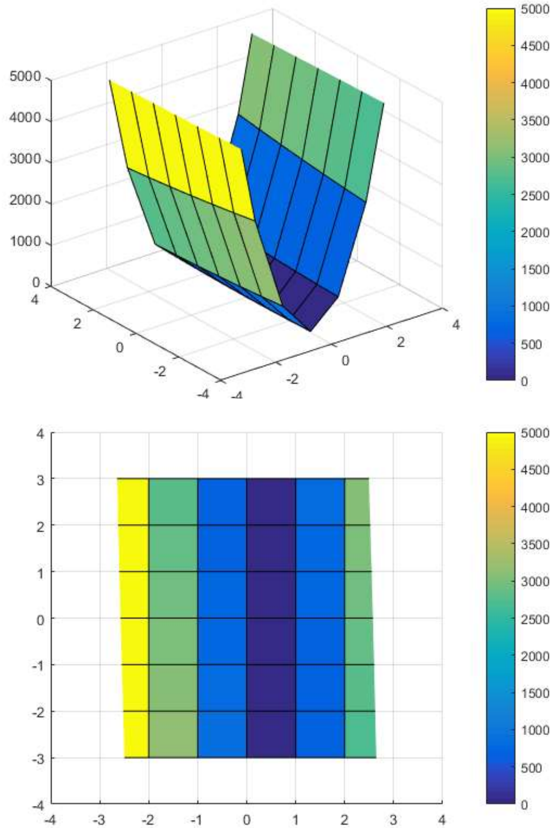
Approximated SSD function

$$\tilde{D}(\delta x, \delta y) = [\delta x, \delta y] M \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

# Feature detection







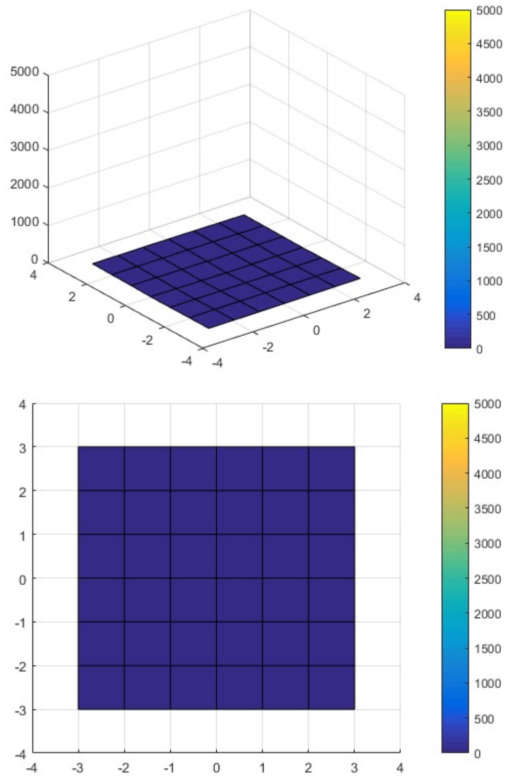
Original SSD function

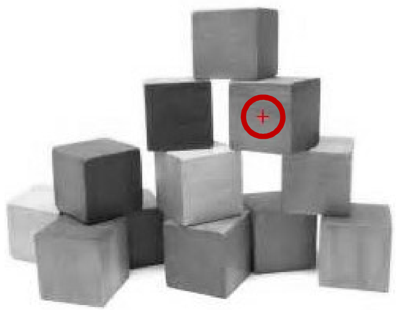
$$D(\delta x, \delta y) = \sum_{\delta x, \delta y \in \Omega} (I(x,y) - I(x+\delta x, y+\delta y))^2$$

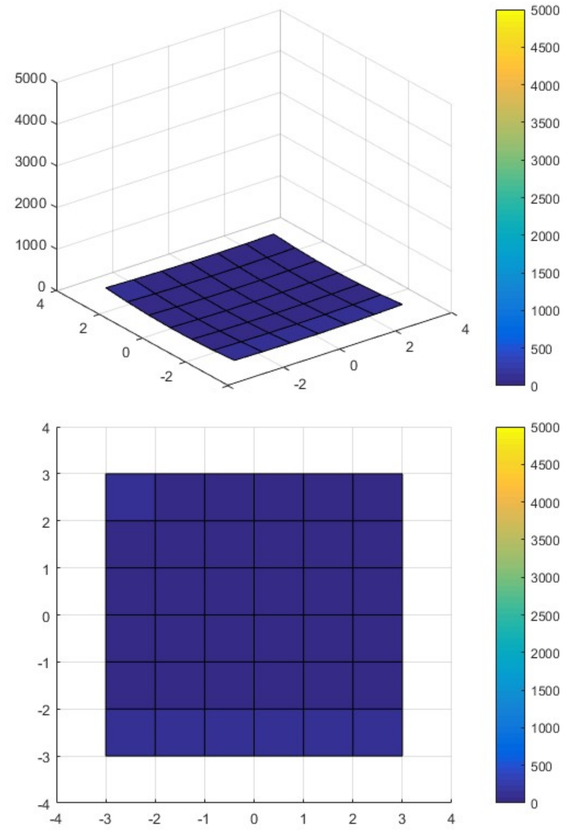
Approximated SSD function

$$\tilde{D}(\delta x, \delta y) = [\delta x, \delta y] M \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

# Feature detection






Original SSD function

$$D(\delta x, \delta y) = \sum_{\delta x, \delta y \in \Omega} (I(x, y) - I(x + \delta x, y + \delta y))^2$$

Approximated SSD function

$$\tilde{D}(\delta x, \delta y) = [\delta x, \delta y] M \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

# Feature detection

- Approximated SSD function is very close to the real one locally.
- The auto-correlation matrix $M$ plays a key role to detect the corners.

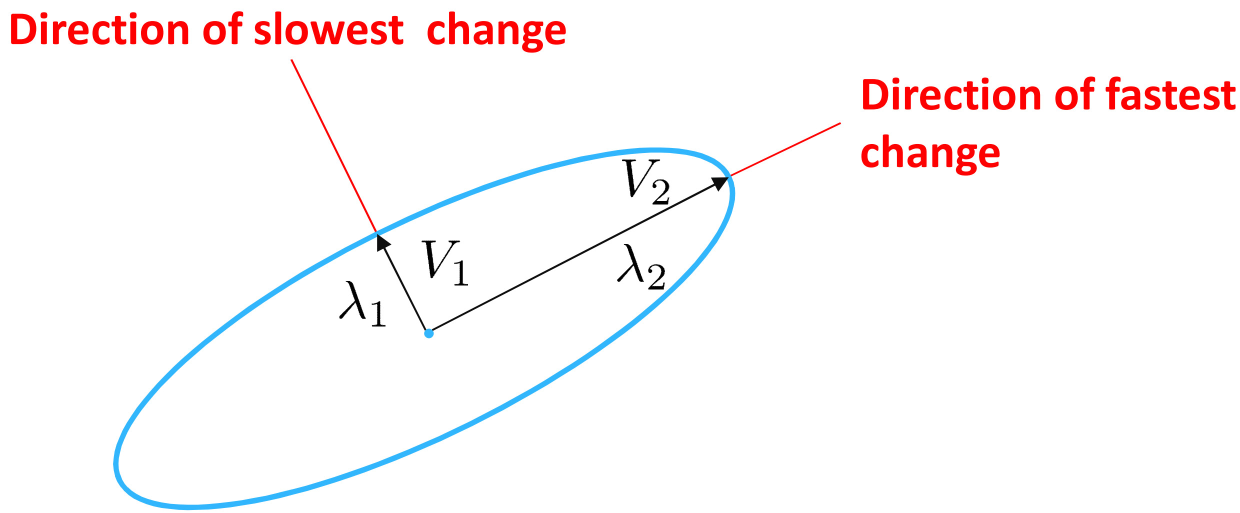
$$\tilde{D}(\delta x, \delta y) = [\delta x, \delta y] M \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$$

- $M$ is a $2 \times 2$ positive definite symmetric matrix.
- Let $\lambda_1, \lambda_2, V_1, V_2$ be its eigenvalues and the corresponding eigenvectors.

$$MV_1 = \lambda_1 V_1$$
$$MV_2 = \lambda_2 V_2$$

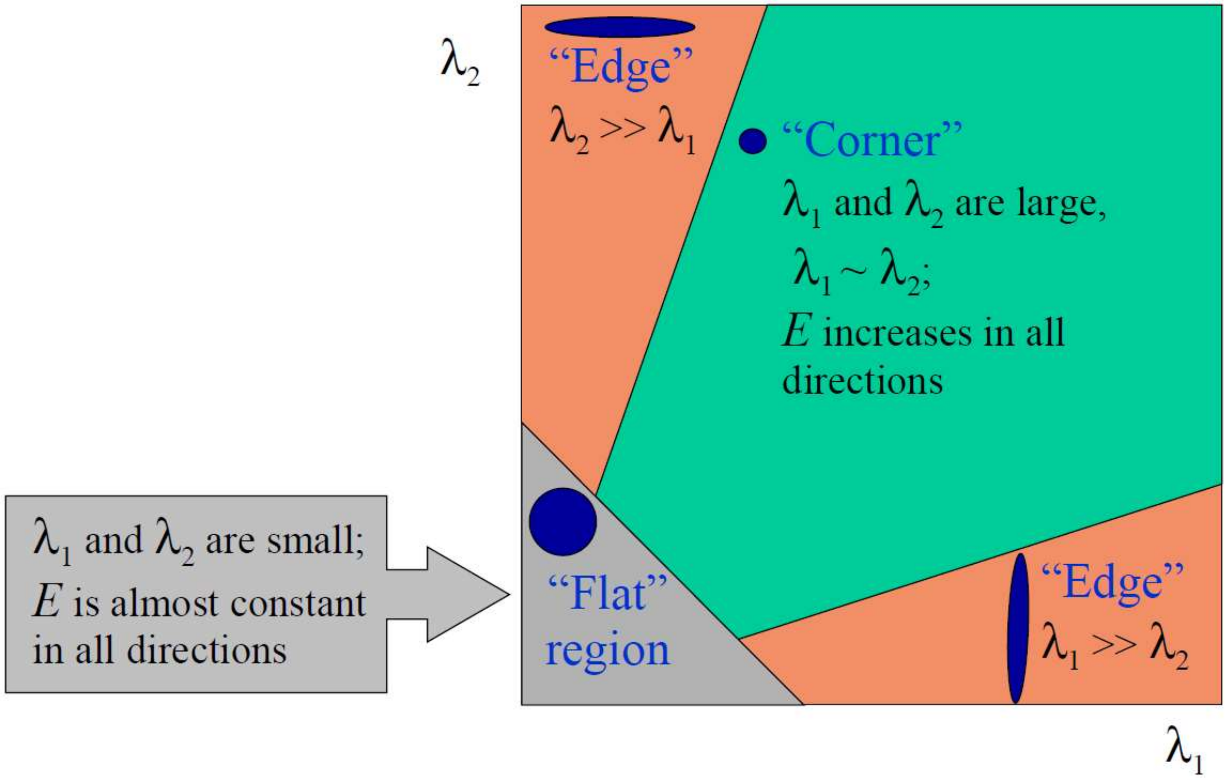
# Feature detection

- Eigenvalue indicates the change of SSD



# Feature detection

- Classify image points using eigenvalues of M:



# Feature detection

- Measure of corner response
  - Harris method

$$r = det(M) - k(trace(M))^2$$

$$det(M) = \lambda_1$$
$$trace(M) = \lambda_1 + \lambda_2$$

$k$ is an empirical constant ( 0.04 ~ 0.06)

  - Shi-Tomasi method

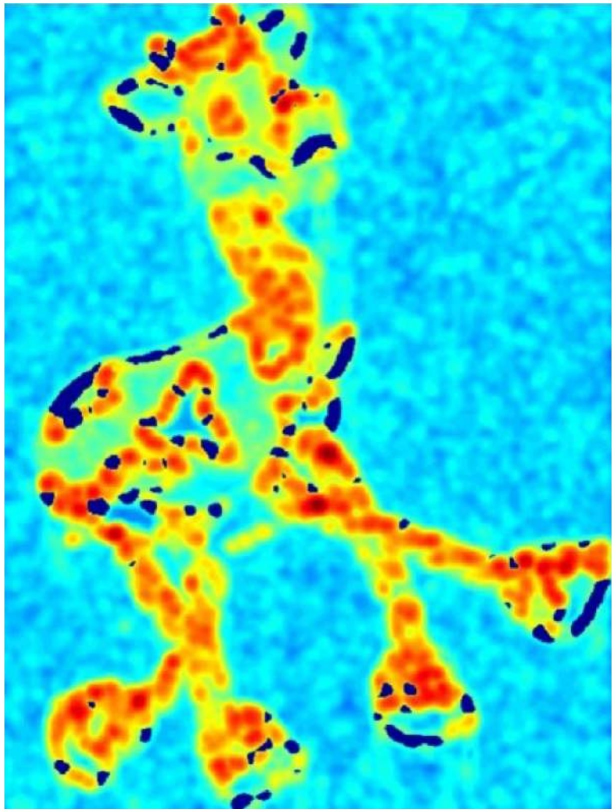
$$r = \min(\lambda_1, \lambda_2)$$

Shi, Jianbo, and Carlo Tomasi. "Good features to track." *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994.

# Feature detection

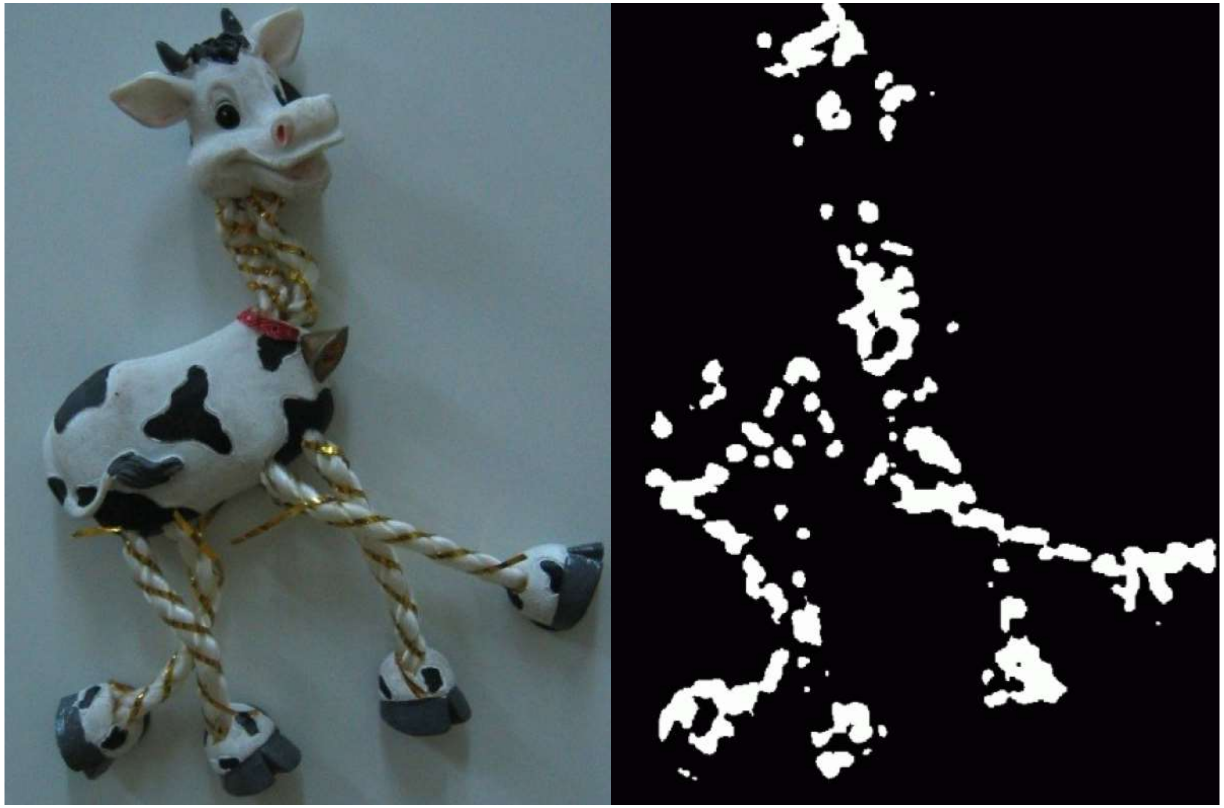
- Harris detector : work flow

compute corner response



# Feature detection

- Harris detector : work flow



compute corner response

↓

Find region
R > threshold

↓

# Feature detection

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

- Harris detector : work flow

compute corner response

↓

Find region R > threshold

↓

find local maximum in R

↓

# Feature detection

- Harris detector : work flow

compute corner response

↓

Find region R > threshold

↓

find local maximum in R

↓

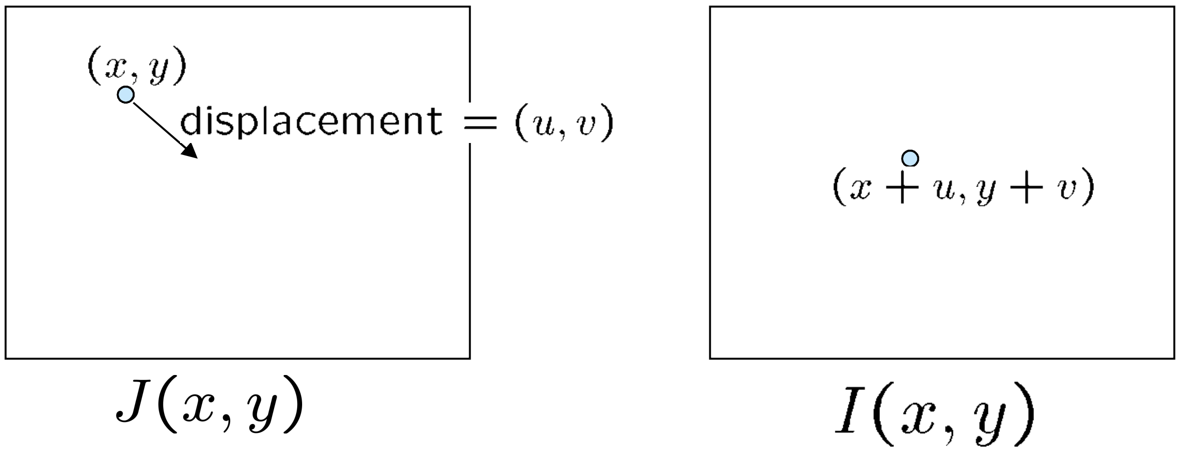
corners

# Summary

- Feature points should be recognizable, stable and accurate located.
- Harris corner is one kind of features that located on the position whose patch is very different from neighbor patches.
- The quality of corner is determined by the auto-correlation matrix $M$.
- The point with an auto-correlation matrix with two large Eigen values is detected as a corner.

# Feature tracking

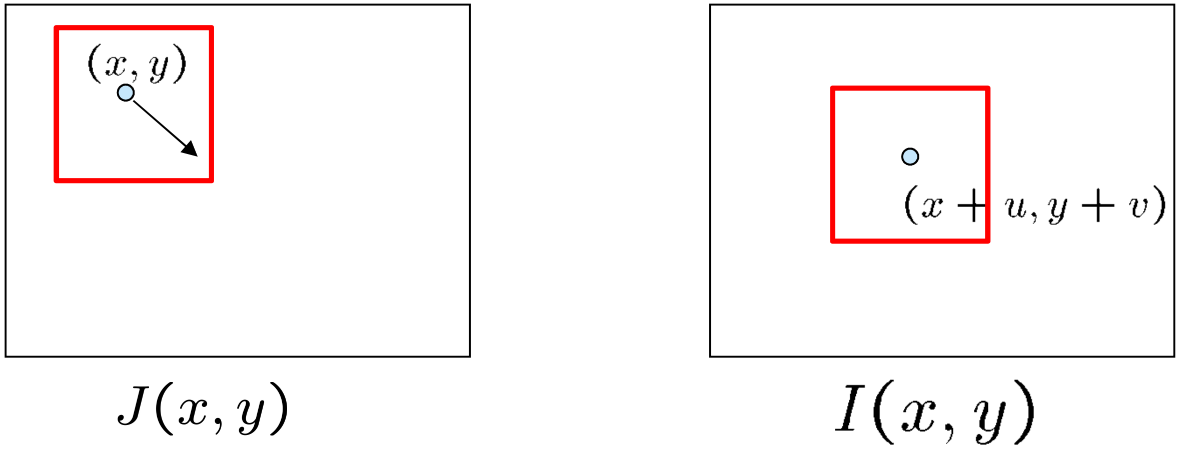
- Lucas-Kanade optical flow algorithm




Given a feature point in the previous frame $J$, how to infer its position in the current frame $I$ ?

# Feature tracking

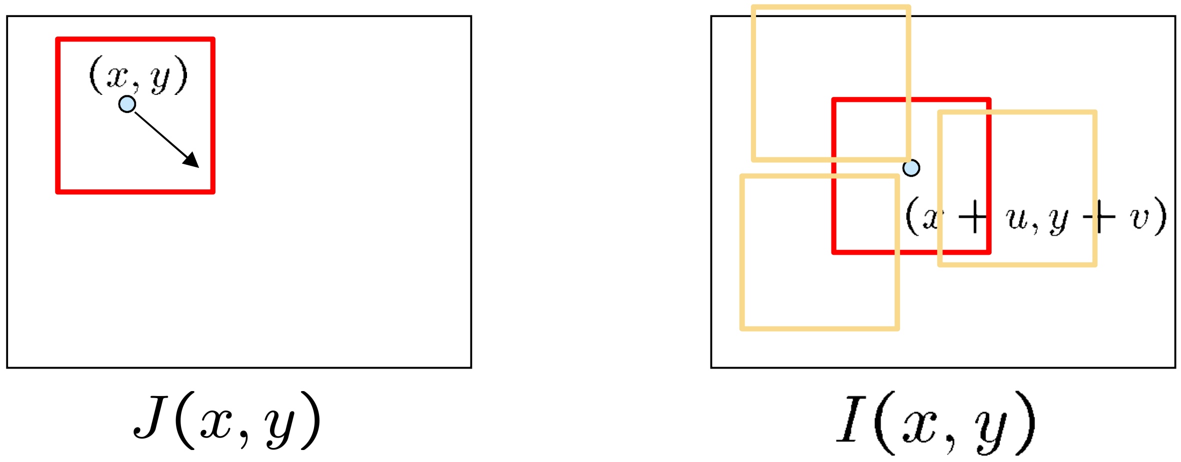
- Assumption
  - Displacement is small.
  - Color has little change.




Compare color difference between two patches.

# Feature tracking

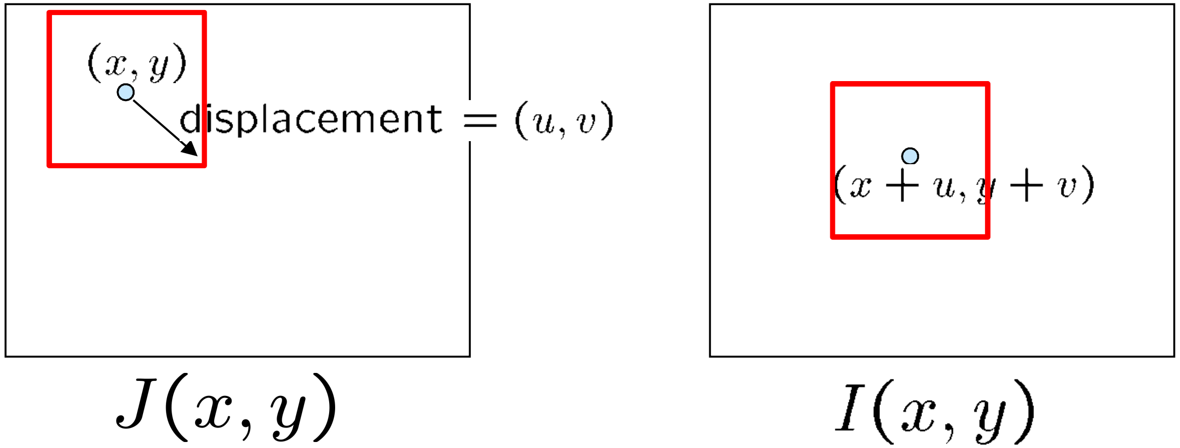
- A brute force searching method:
  - search the patch with the smallest color difference from original patch.




- **Inefficient:** searching in a 10x10 window requires 100 times of patch comparison.
- **Inaccurate:** operates on pixels

# Feature tracking

- Lucas-Kanade optical flow method




- We are trying to minimize the **S**um of **S**quared **D**ifference (**SSD**)

$$D(u,v) = \sum_{u,v\in\Omega}(I(x+u, y+v) - J(x,y))^2$$

# Feature tracking

- Local minimum: let the gradient be zero!

$$\nabla D = \begin{bmatrix} \partial D / \partial u \\ \partial D / \partial v \end{bmatrix} = 0$$

$$D(u,v) = \sum_{u,v \in \Omega} (\boxed{I(x+u, y+v)} - J(x,y))^2$$

We can approximate this term by Taylor series expansion (Lecture 2)

# Feature tracking

- First-order Taylor expansion

$$I(x+u, y+v) = I(x,y) + \nabla I(x,y) \begin{bmatrix} u \\ v \end{bmatrix} = I + I_x u + I_y v$$

$$S(u,v) = \sum_{x,y\in\Omega} (\boxed{I(x+u, y+v)} - J(x,y))^2$$

$$= \sum_{x,y\in\Omega} (I - J + I_x u + I_y v)^2$$

$$\nabla D = \begin{bmatrix} \partial D / \partial u \\ \partial D / \partial v \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{x,y} I_x (I - J) + \sum_{x,y} I_x^2 u + \sum_{x,y} I_x I_y v \\ \sum_{x,y} I_y (I - J) + \sum_{x,y} I_x I_y u + \sum_{x,y} I_y^2 v \end{bmatrix}$$

# Matching by tracking

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

- Finally we get the following equation

$$\boxed{\begin{bmatrix} \sum_{x,y} I_x^2 + \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y + \sum_{x,y} I_y^2 \end{bmatrix}} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{x,y} I_x (I - J) \\ \sum_{x,y} I_y (I - J) \end{bmatrix}$$

Note that it is the auto-correlation matrix in Harris corner detection (P9).

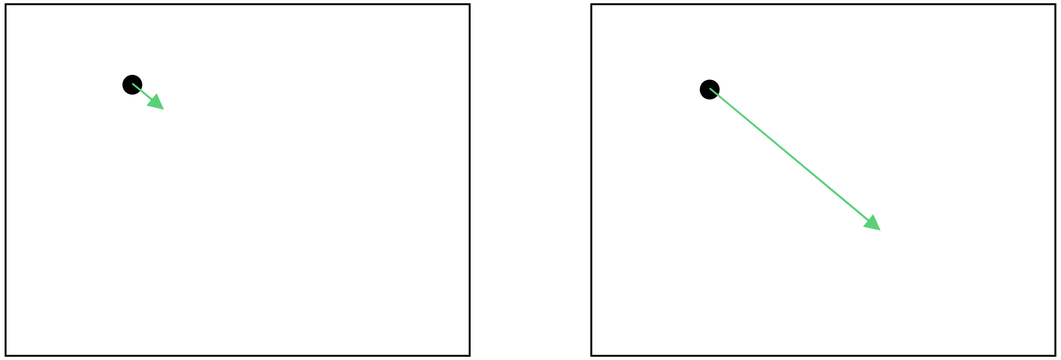
$$M \Delta X = b$$

$$\Rightarrow \Delta X = M^{-1} \mathbf{b}$$

Finally, we get the displacement (flow)!

# Feature tracking

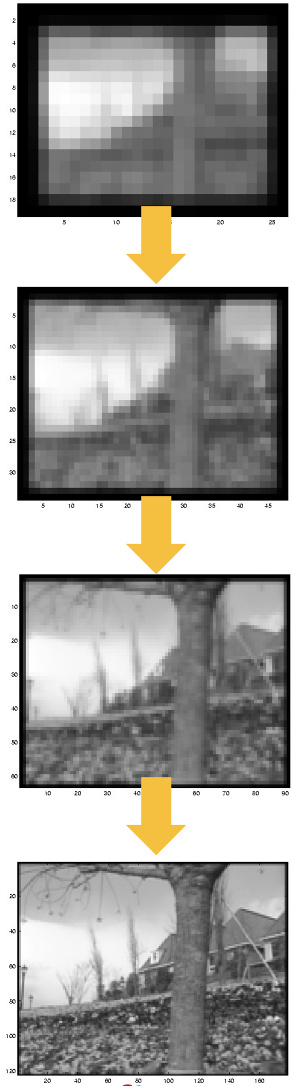
- Problem:
  - It assumes that the displacement is small ( ~ 1 pixel ) that Taylor series expansion holds.
- How about **large** displacement value?



Small displacement　　　　Large displacement

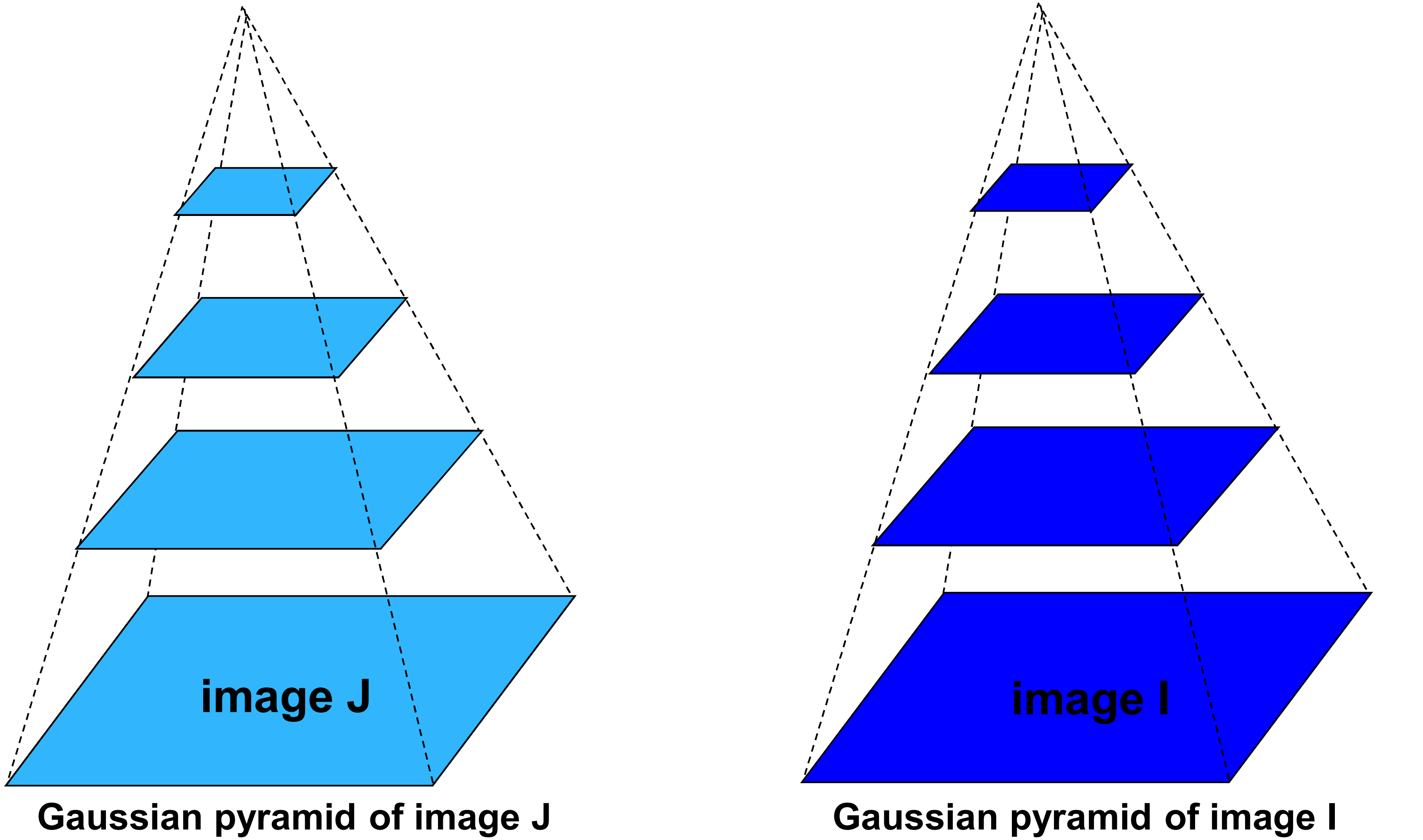
# Feature tracking

- Coarse-to-fine manner
  – Compute displacement in low resolution images (coarse)
  – Use the coarse displacement value as an initialization and refine the displacement in high resolution images( fine)
  – repeatedly run above steps until the finest level has been reached.



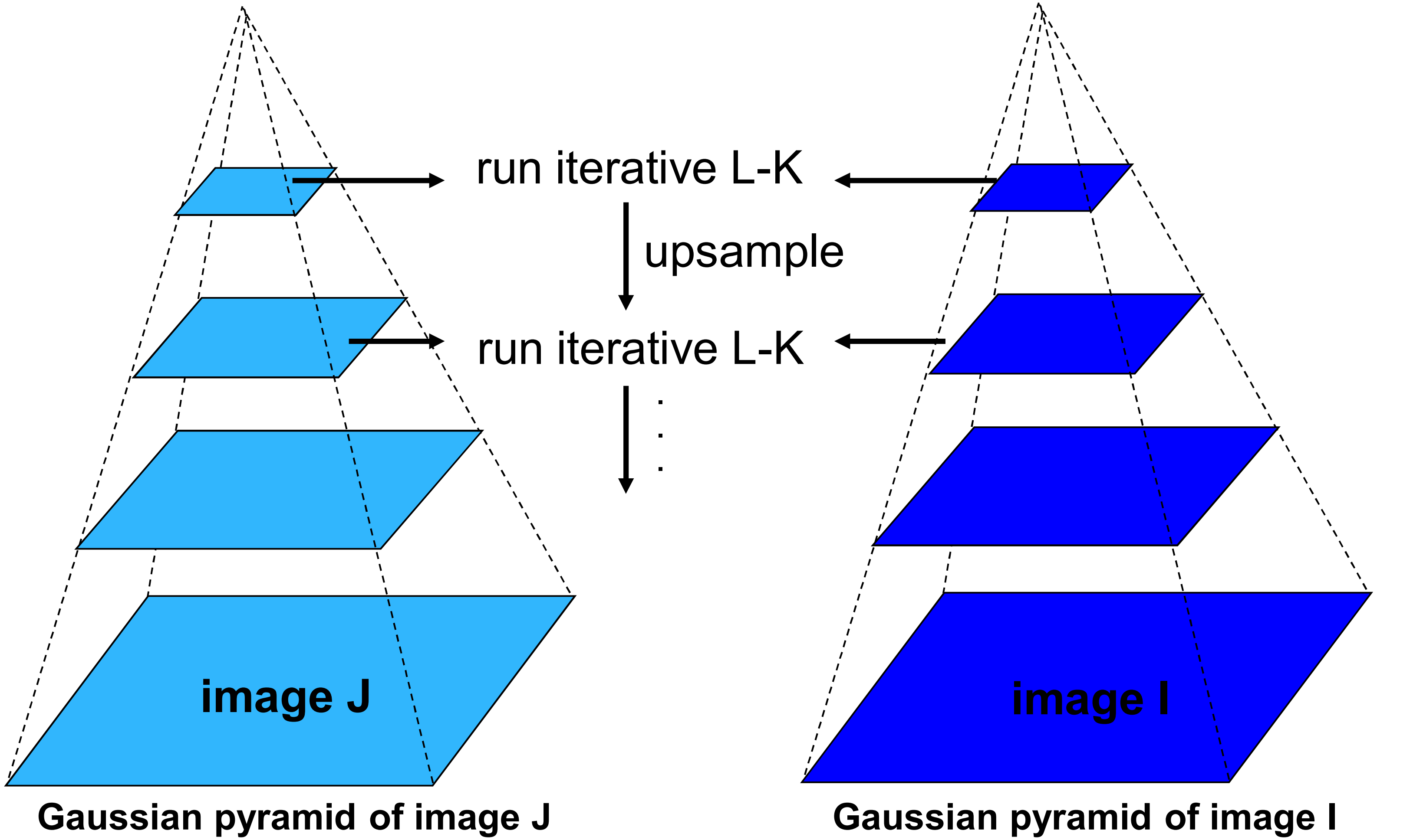
# Feature tracking

- Build image pyramid



# Feature tracking

- Run Lucuas-Kanade algorithm in coarse-to-fine manner



# KLT tracker

- OpenCV implementation

```
void goodFeaturesToTrack(InputArray image,
                         OutputArray corners,
                         int maxCorners,
                         double qualityLevel,
                         double minDistance,
                         InputArray mask=noArray(),
                         int blockSize=3,
                         bool useHarrisDetector=false,
                         double k=0.04 )

void calcOpticalFlowPyrLK(InputArray prevImg,
                          InputArray nextImg,
                          InputArray prevPts,
                          InputOutputArray nextPts,
                          OutputArray status,
                          OutputArray err,
                          Size winSize=Size(21,21),
                          int maxLevel=3,
                          TermCriteria   criteria,
                          int flags=0,
                          double minEigThreshold=1e-4 )
```

# Summary

- For video frames, we can use tracking to match the feature points in successive frames.

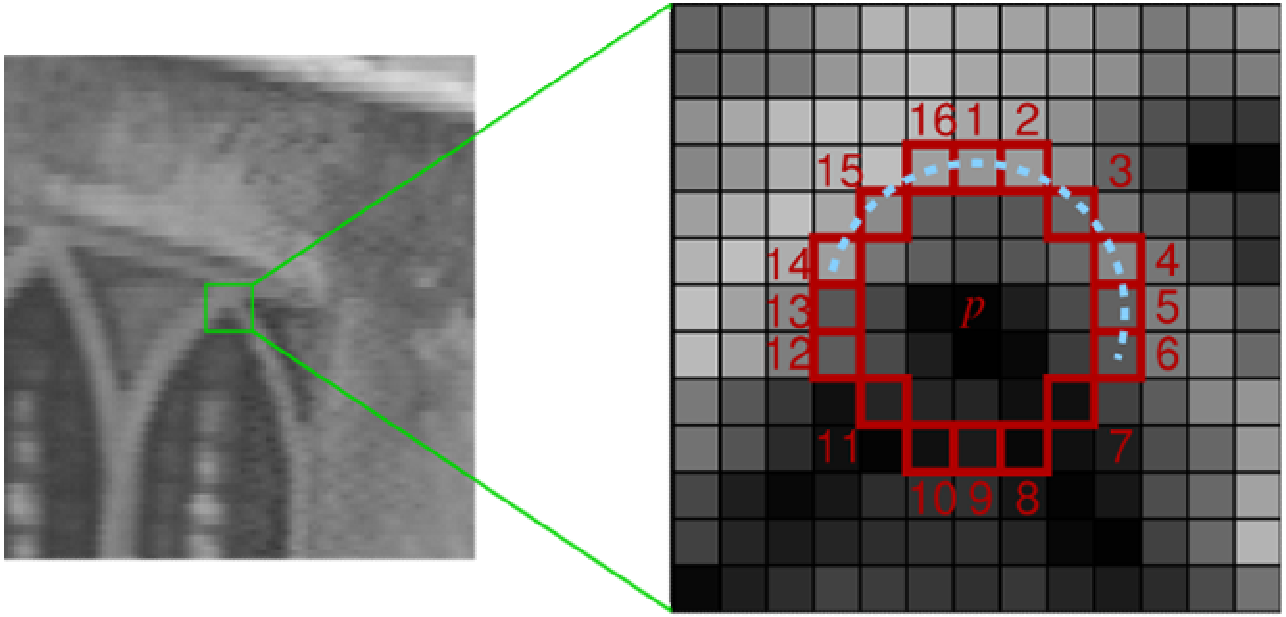
- When the displacement is small, matching becomes a problem of solving a linear equation.

$$M\Delta X = b$$

- If the displacement is large, we can use coarse-to-fine approach to compute displacement recursively.

# FAST detector

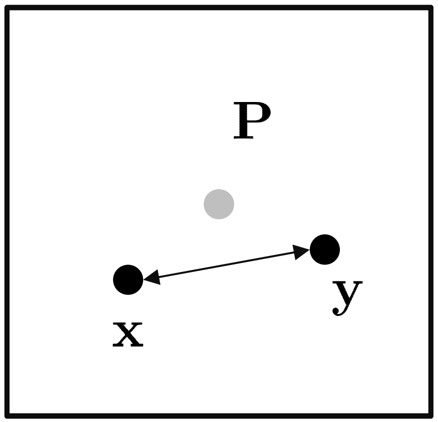
- Condition 1: A set of N contiguous pixels S, ∀ x ∈ S, the intensity of x (Ix) > Ip + threshold t
- Condition 2: A set of N contiguous pixels S, ∀ x ∈ S, Ix < Ip - t



# BRIEF descriptor

- Given a patch $\boldsymbol{P}$, randomly sample two pixels $\boldsymbol{x}$ and $\boldsymbol{y}$, compare their intensities by the following test:

$$\tau(\mathbf{P};\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{if} \quad \mathbf{P}(\mathbf{x}) < \mathbf{P}(\mathbf{y}) \\ 0 & \quad\;\;\, \text{otherwise} \end{cases}$$




Sample $n_d$ pixel-pairs and run the binary test, we get the BRIEF descriptor of $\boldsymbol{P}$ as:

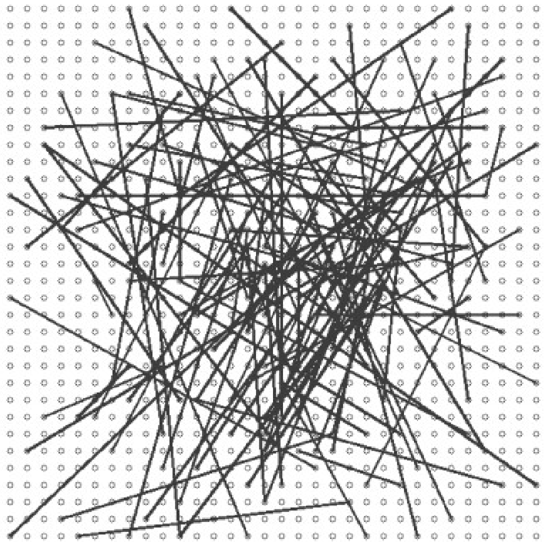
$$f_{n_d}(\mathbf{P}) = \sum_{i=1}^{n_d} 2^{i-1}\tau(\mathbf{P};\mathbf{x},\mathbf{y})$$

$$n_d = 128, 256, 512$$

Note that the descriptor is a bit string.

# BRIEF descriptor

- The approach  to choosing the sample positions:



$$(\mathbf{x},\mathbf{y}) \sim i.i.d. \quad \mathcal{N}(0, \tfrac{1}{25}S^2)$$

$S$ is the width of the window.

# BRIEF descriptor

- **Hamming distance** to measure the distance between two descriptors.
- The Hamming distance is the number of positions at which the corresponding symbols are different:

  - "karolin" and "kathrin" is 3.
  - "karolin" and "kerstin" is 3.
  - **1011101** and **1001001** is 2.
  - **2173896** and **2233796** is 3.

- Very fast - only bit operations are involved.

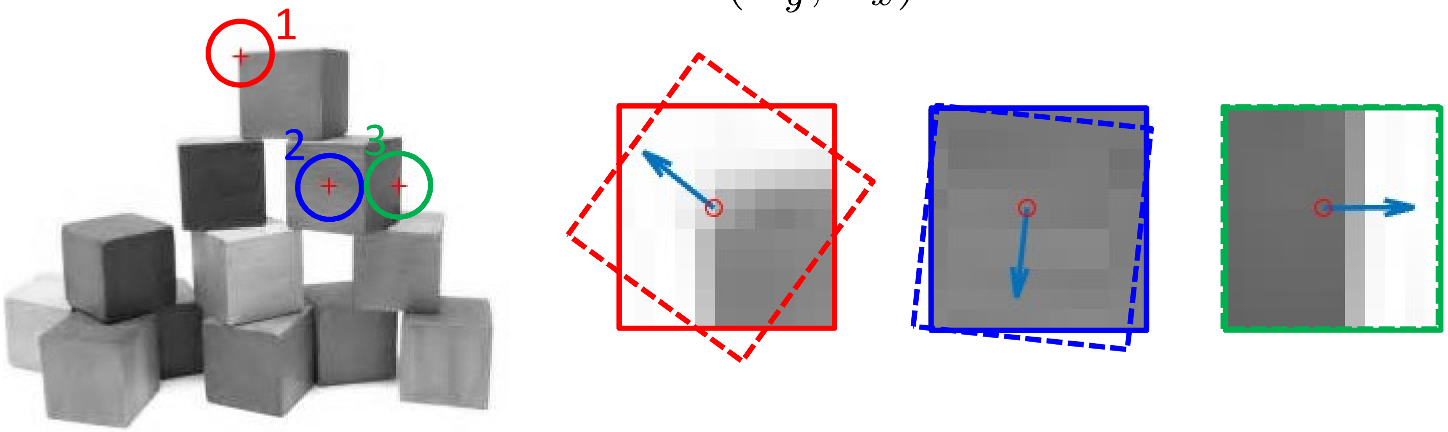
# ORB descriptor

- An improved version of BRIEF descriptor by adding **rotational invariance**.
- The corner orientation is defined by the intensity centroid:

$$C_x = \frac{\sum_{x,y} xI(x,y)}{\sum_{x,y} I(x,y)} \qquad C_y = \frac{\sum_{x,y} yI(x,y)}{\sum_{x,y} I(x,y)}$$

- The corner orientation is

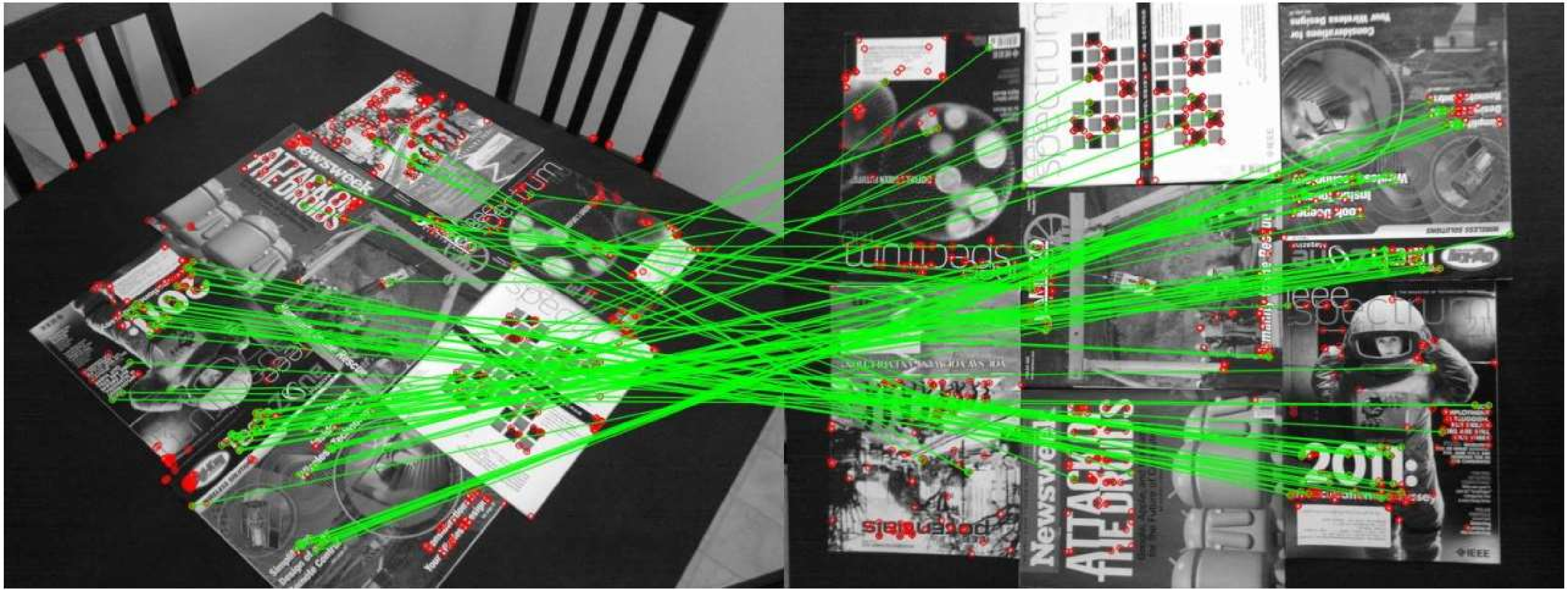
$$\theta = atan2(C_y, C_x)$$



# ORB descriptor

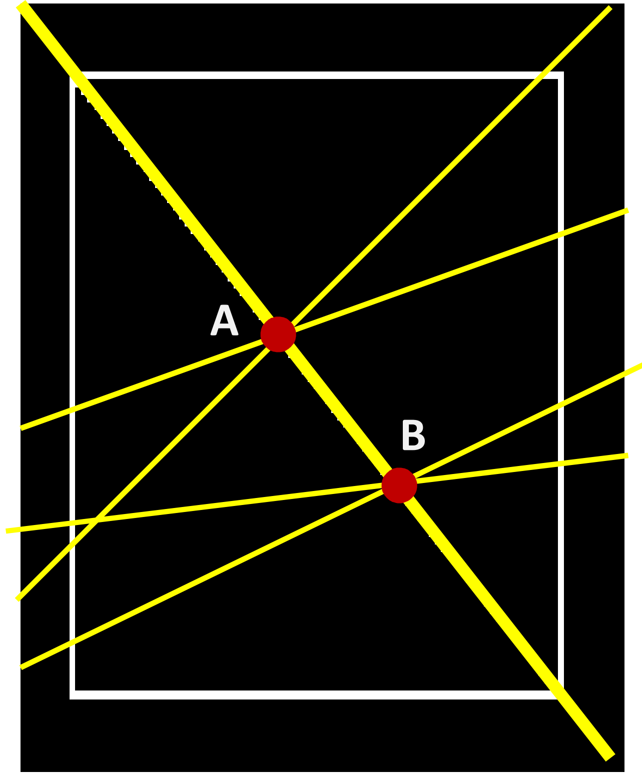
- ORB is much faster than other descriptors

| Descriptor | ORB | SURF | SIFT |
|---|---|---|---|
| Time per frame (ms) | **15.3** | 217.3 | 5228.7 |



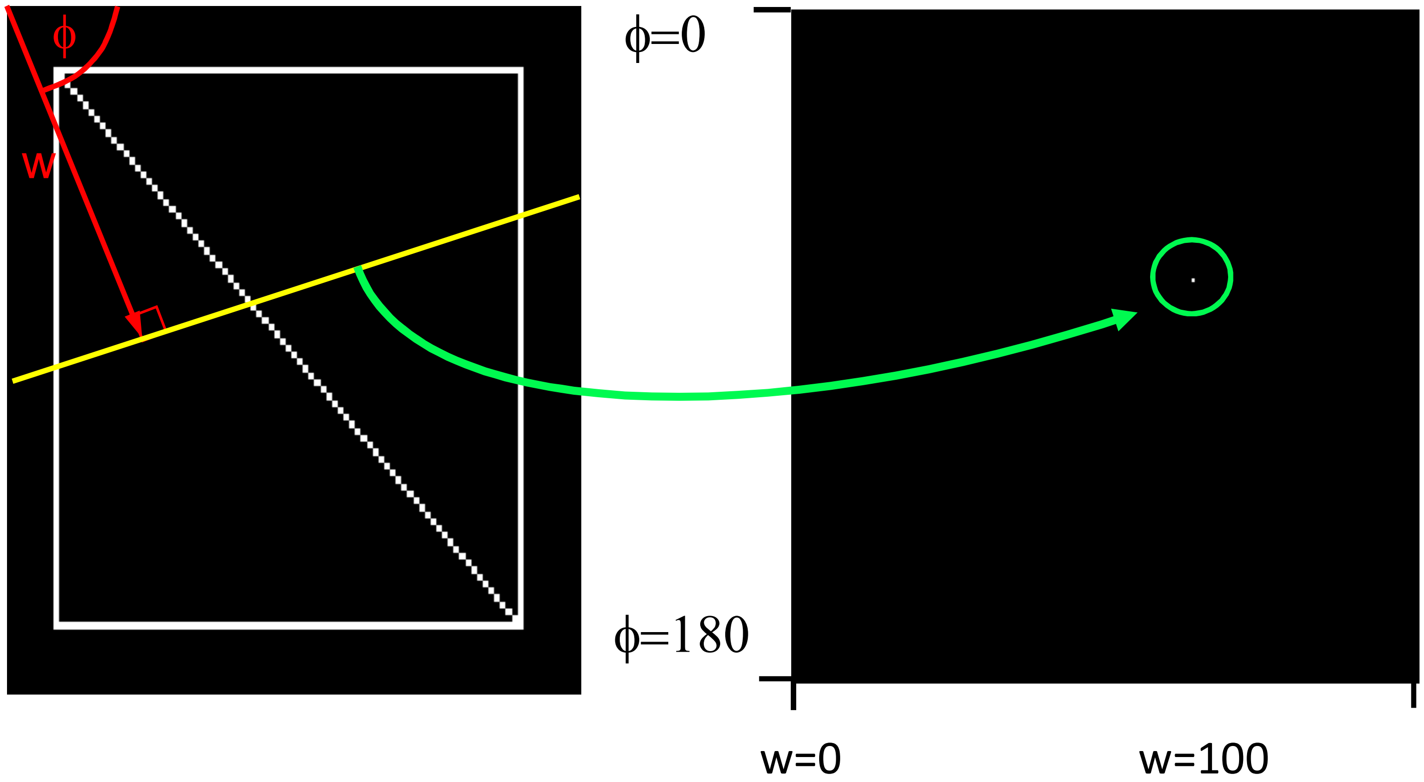
# Line detection

- Hough transformation
  - The basic idea:
    - Each point votes for a group of lines that across this point.
    - The lines that has been voted by many points are straight lines.



# Line detection

- A line can be represented by two parameters $(\emptyset, w)$.
- Each while pixel corresponds to a point in the parameter space (Hough space) .



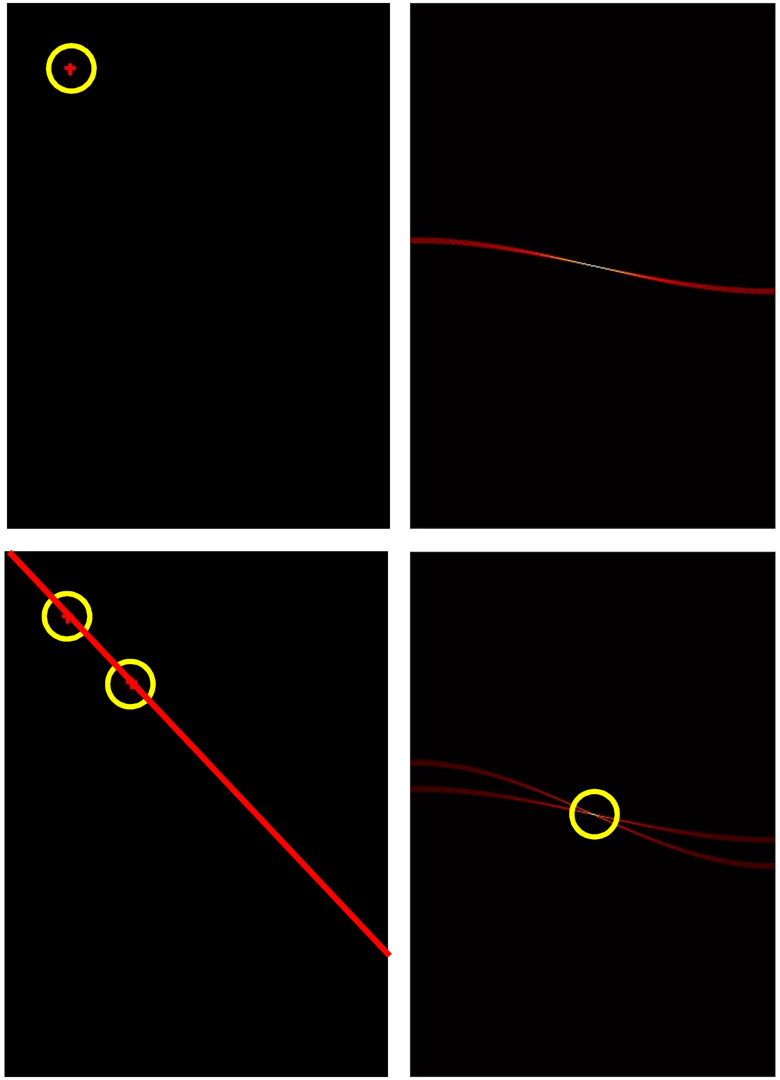
# Line detection

One point in image space corresponds to a sinusoidal curve in Hough space

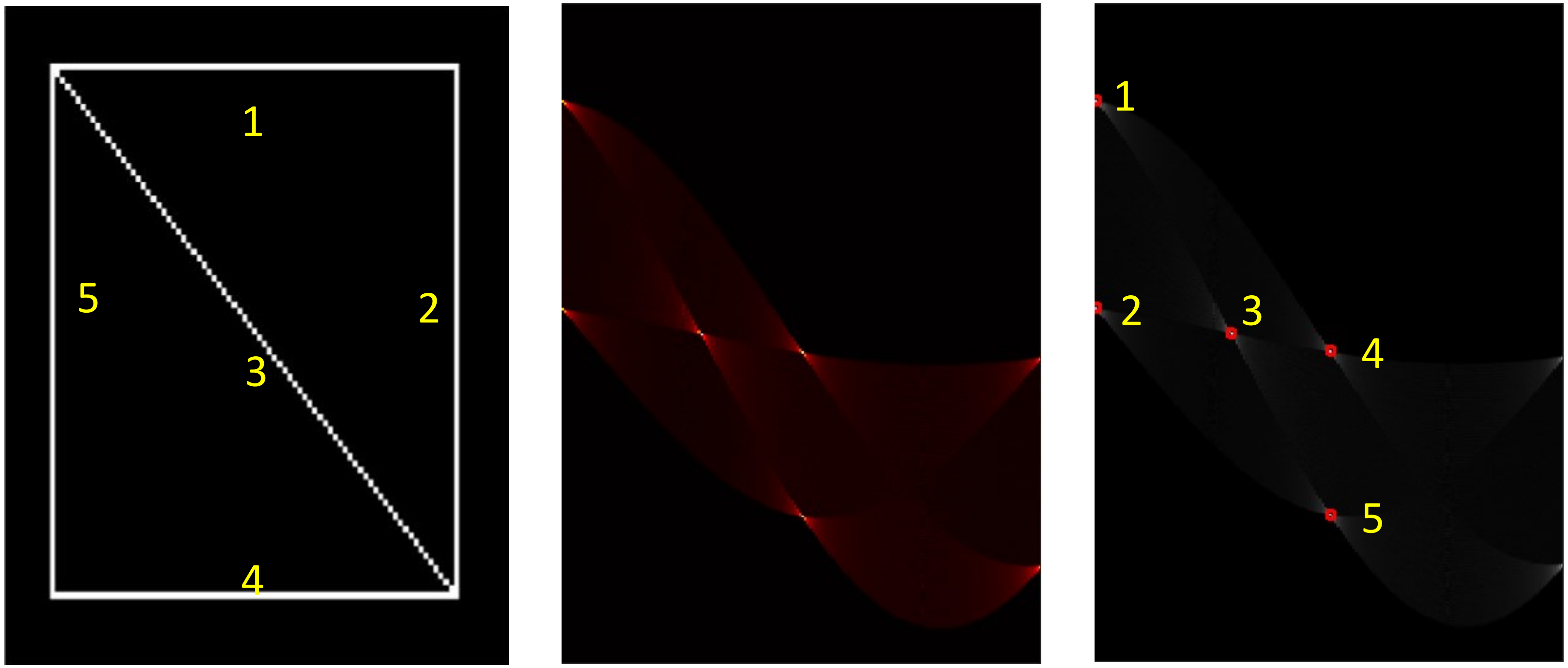
Two points correspond to two curves in Hough space

The intersection of those two curves has “two votes”.

This intersection represents the straight line in image space that passes through both points

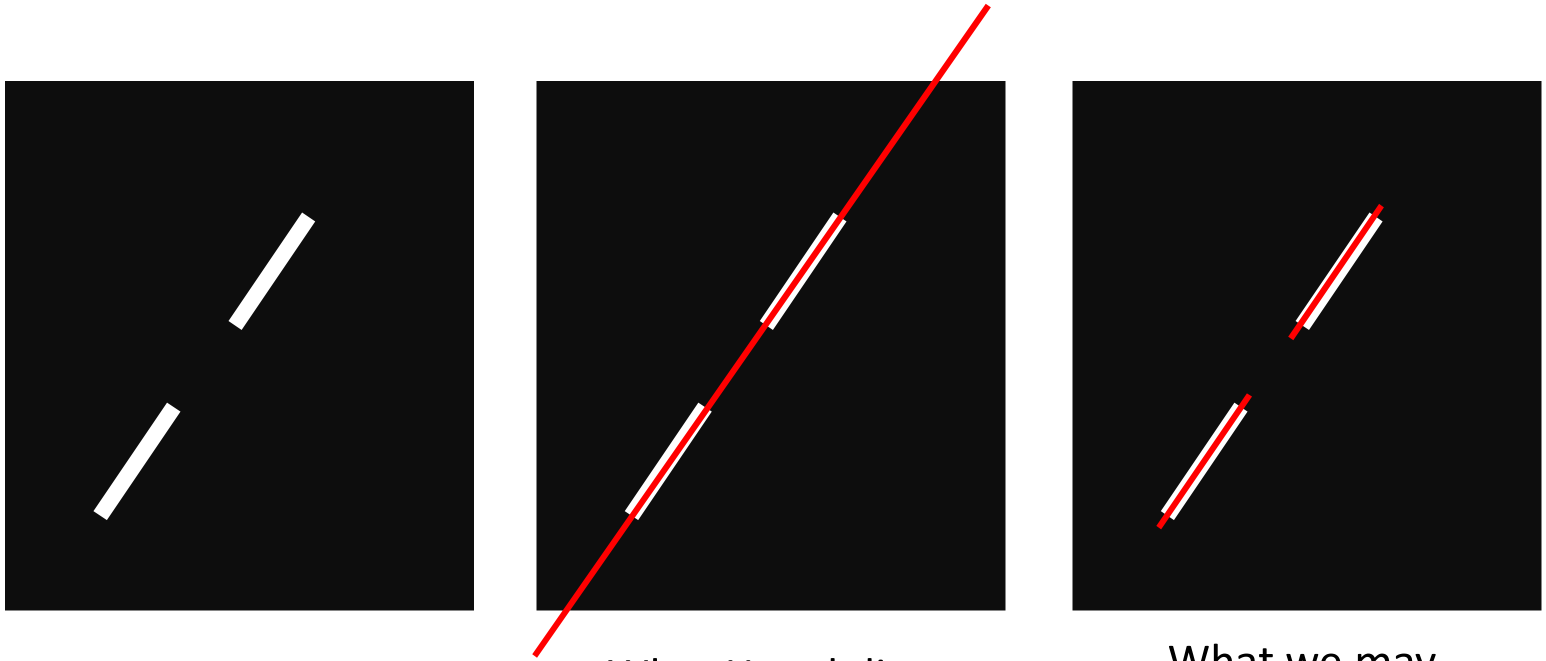


# Line detection



# Line detection

- One problem of Hough line detection is that it detects a line but not a line segment .
- In most situation, we want only line segments instead of infinite lines.



What Hough line detection obtains.

What we may want.

# Line detection

- Other line detector
  - LSD line detector [1]

Von Gioi, R. G., Jakubowicz, J., Morel, J. M., & Randall, G. (2010). LSD: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, *32*(4), 722-732.

# Summary

- BRIEF descriptor is a binary descriptor that compares a random pair of pixel intensities.
- ORB is an oriented version of BREIF descriptor by selecting the intensity centroid as the corner orientation.
- Hough transformation is transform a point into a sinuous curve in the parameter space (Hough space)
- Hough line detection detect only straight lines but not line segments.
- LSD line detector can be used to detect line segments