

# Feature Detection and Matching

CS4243 Computer Vision and Pattern Recognition

Leow Wee Kheng

Department of Computer Science  
School of Computing  
National University of Singapore



# Outline

- 1 Feature Detection
  - Harris Corner
  - Tomasi's Good Feature
  - Comparison
  - SIFT
- 2 Feature Descriptors
  - SIFT Descriptors
- 3 Feature Matching
- 4 Summary
- 5 Exercise
- 6 Further Reading
- 7 Reference

# Feature Detection

For image registration, need to obtain correspondence between images.  
Basic idea:

- detect feature points, also called **keypoints**
- match feature points in different images

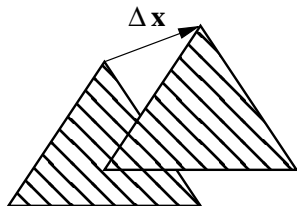
Want feature points to be detected consistently and matched correctly.

Many features available

- Harris corner
- Tomasi's "good features to track"
- SIFT: Scale Invariant Feature Transform
- SURF: Speeded Up Robust Feature
- GLOH: Gradient Location and Orientation Histogram
- etc.

# Harris Corner

Observation:



- A shifted corner produces some difference in the image.
- A shifted uniform region produces no difference.
- So, look for large difference in shifted image.

Suppose an image patch  $W$  at  $\mathbf{x}$  is shifted by a small amount  $\Delta\mathbf{x}$ . Then, the sum-squared difference at  $\mathbf{x}$  is

$$E(\mathbf{x}) = \sum_{\mathbf{x}_i \in W} [I(\mathbf{x}_i) - I(\mathbf{x}_i + \Delta\mathbf{x})]^2. \quad (1)$$

That is,

$$E(x, y) = \sum_{(x_i, y_i) \in W} [I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2. \quad (2)$$

This is also called the **auto-correlation function**.

Apply Taylor's series expansion to  $I(\mathbf{x}_i + \Delta\mathbf{x})$ :

$$I(x_i + \Delta x, y_i + \Delta y) = I(x_i, y_i) + \frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y \quad (3)$$

$$= I(x_i, y_i) + I_x \Delta x + I_y \Delta y \quad (4)$$

$$= I(\mathbf{x}_i) + (\nabla I)^\top \Delta\mathbf{x} \quad (5)$$

where  $\nabla I = (I_x, I_y)^\top$ .

Substituting Eq. 5 into Eq. 1 yields

$$E(\mathbf{x}) = \sum_W [I_x \Delta x + I_y \Delta y]^2 \quad (6)$$

$$= \sum_W [I_x^2 \Delta^2 x + 2I_x I_y \Delta x \Delta y + I_y^2 \Delta^2 y] \quad (7)$$

$$= (\Delta \mathbf{x})^\top \mathbf{A}(\mathbf{x}) \Delta \mathbf{x} \quad (8)$$

where the auto-correlation matrix  $\mathbf{A}$  is given by (Exercise)

$$\mathbf{A} = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}. \quad (9)$$

$\mathbf{A}$  captures intensity pattern in  $W$ .

Response  $R(\mathbf{x})$  of Harris corner detector is given by [HS88]:

$$R(\mathbf{x}) = \det \mathbf{A} - \alpha(\text{tr } \mathbf{A})^2. \quad (10)$$

Two ways to define corners:

(1) Large response

The locations  $\mathbf{x}$  with  $R(\mathbf{x})$  greater than certain threshold.

(2) Local maximum

The locations  $\mathbf{x}$  where  $R(\mathbf{x})$  are greater than those of their neighbors, i.e., apply **non-maximum suppression**.

Sample result (large response):



Many corners are detected near each other.  
So, better to find local maximum.

# Tomasi's Good Feature

Shi and Tomasi considered weighted auto-correlation [ST94]:

$$E(\mathbf{x}) = \sum_{\mathbf{x}_i \in W} w(\mathbf{x}_i) [I(\mathbf{x}_i) - I(\mathbf{x}_i + \Delta\mathbf{x})]^2 \quad (11)$$

where  $w(\mathbf{x}_i)$  is the weight.

Then,  $\mathbf{A}$  becomes

$$\mathbf{A} = \begin{bmatrix} \sum_W w I_x^2 & \sum_W w I_x I_y \\ \sum_W w I_x I_y & \sum_W w I_y^2 \end{bmatrix}. \quad (12)$$

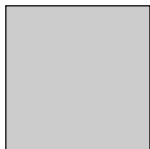
$\mathbf{A}$  is a  $2 \times 2$  matrix. This means there exist scalar values  $\lambda_1, \lambda_2$  and vectors  $\mathbf{v}_1, \mathbf{v}_2$  such that

$$\mathbf{A} \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, 2 \quad (13)$$

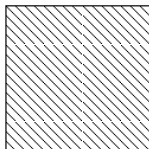
- $\mathbf{v}_i$  are the orthonormal eigenvectors, i.e.,

$$\mathbf{v}_i^\top \mathbf{v}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

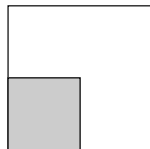
- $\lambda_i$  are the eigenvalues; expect  $\lambda_i \geq 0$ .



(1)



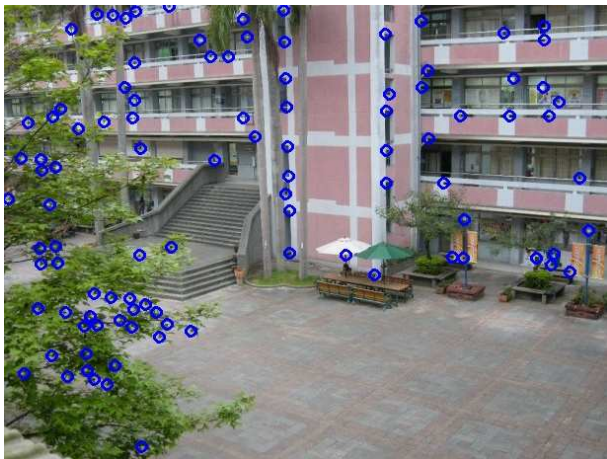
(2)



(3)

- (1) If both  $\lambda_i$  are small, then feature does not vary much in any direction.  $\Rightarrow$  uniform region (bad feature)
- (2) If the larger eigenvalue  $\lambda_1 \gg \lambda_2$ , then the feature varies mainly in the direction of  $\mathbf{v}_1$ .  $\Rightarrow$  edge (bad feature)
- (3) If both eigenvalues are large, then the feature varies significantly in both directions.  $\Rightarrow$  corner or corner-like (good feature)
- (4) In practice,  $I$  has a maximum value (e.g., 255).  
So,  $\lambda_1, \lambda_2$  also have an upper bound.  
So, only have to check that  $\min(\lambda_1, \lambda_2)$  is large enough.

Sample results (local maximum):



With non-maximum suppression, detected corners are more spread out.

# Comparison

- Tomasi's good feature uses smallest eigenvalue  $\min(\lambda_1, \lambda_2)$ .
- Harris corner uses  $\det \mathbf{A} - \alpha(\text{tr } \mathbf{A})^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$ .
- Brown et al. [BSW05] use the harmonic mean

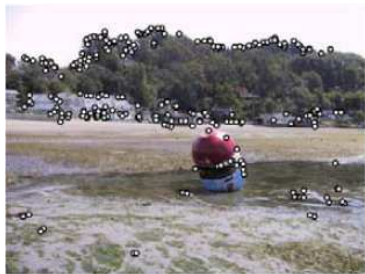
$$\frac{\det \mathbf{A}}{\text{tr } \mathbf{A}} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}. \quad (15)$$

# Subpixel Corner Location

- Locations are detected keypoints are typically at integer coordinates.
- To get more accurate real-number coordinates, need to run **subpixel** algorithm.
- General idea: starting with an approximate location of a corner, find the accurate location that lies at the intersections of edges.

# Adaptive Non-maximal Suppression

- Non-maximal suppression: look for local maximal as keypoints.
- Can lead to uneven distribution of detected keypoints.
- Brown et al. [BSW05] used adaptive non-maximal suppression:
  - local maximal
  - response value is significantly larger than those of its neighbors



(a) Strongest 250



(b) Strongest 500

(c) ANMS 250,  $r = 24$ (d) ANMS 500,  $r = 16$

# Scale Invariance

In many applications, the scale of the object of interest may vary in different images.



Simple but inefficient solution:

- Extract features at many different scales.
- Match them to the object's known features at a particular scale.

More efficient solution:

- Extract features that are invariant to scale.

# SIFT

Scale Invariant Feature Transform (SIFT) [Low04].

Convolve input image  $I$  with Gaussian  $G$  of various scale  $\sigma$ :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (16)$$

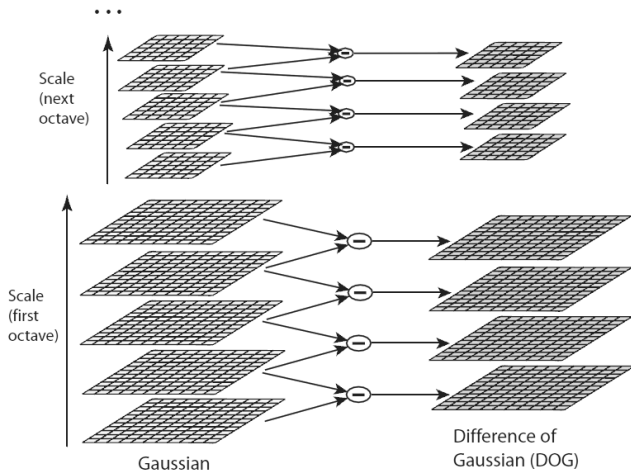
where

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \quad (17)$$

This produces  $L$  at different scales.

To detect stable keypoint, convolve image  $I$  with difference of Gaussian:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \quad (18)$$



- Have 3 different scales within each octave (doubling of  $\sigma$ ).
- Successive DOG images are subtracted to produce  $D$ .
- $D$  images in a lower octave are downsampled by factor of 2.

Detect local maximum and minimum of  $D(x, y, \sigma)$ :

- Compare a sample point with its 8 neighbors in the same scale and 9 neighbors in the scale above and below.
- Select it if it is larger or smaller than all neighbors.
- Obtain position  $x, y$  and scale  $\sigma$  of keypoint.

Orientation of keypoint can be computed as

$$\theta(x, y) = \tan^{-1} \frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)}. \quad (19)$$

Gradient magnitude of keypoint can be computed as

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}. \quad (20)$$

Keypoints detected may include edge points.

Edge points are not good because different edge points along an edge may look the same.

To reject edge points, form the Hessian  $\mathbf{H}$  for each keypoint

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (21)$$

and reject those for which

$$\frac{\text{tr } \mathbf{H}^2}{\det \mathbf{H}} > 10. \quad (22)$$

# Better Invariance

## Rotation invariance

- Estimate dominant orientation of keypoint.
- Normalize orientation.

## Affine invariance

- Fit ellipse to auto-correlation function or Hessian.
- Apply PCA to determine principal axes.
- Normalize according to principal axes.

For more details, refer to [Sze10] Section 4.1.1.

# Feature Descriptors

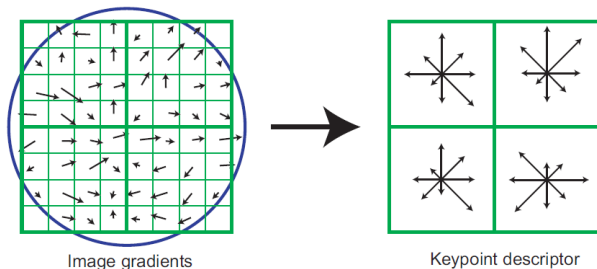
Why need **feature descriptors**?

- Keypoints give only the positions of strong features.
- To match them across different images, have to characterize them by extracting **feature descriptors**.

What kind of feature descriptors?

- Able to match corresponding points across images accurately.
- Invariant to scale, orientation, or even affine transformation.
- Invariant to lighting difference.

# SIFT Descriptors



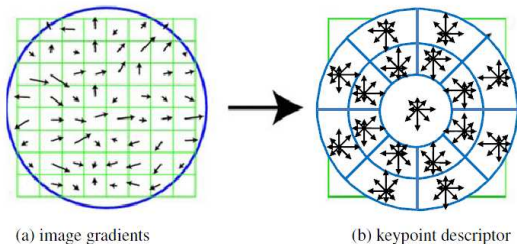
- Compute gradient magnitude and orientation in  $16 \times 16$  region around keypoint location at the keypoint's scale.
- Coordinates and gradient orientations are measured relative to keypoint orientation to achieve orientation invariance.
- Weighted by Gaussian window.
- Collect into  $4 \times 4$  orientation histograms with 8 orientation bins.
- Bin value = sum of gradient magnitudes near that orientation.

- Get  $4 \times 4 \times 8 = 128$  element feature vector.
- Normalize feature vector to unit length to reduce effect of linear illumination change.
- To reduce effect of nonlinear illumination change, threshold feature values to 0.2 and renormalize feature vector to unit length.

# Other Feature Descriptors

Variants of SIFT:

- PCA-SIFT [KS04]  
Use PCA to reduce dimensionality.
- SURF (Speeded Up Robust Features) [BTVG06]  
Use box filter to approximate derivatives.
- GLOH (Gradient Location-Orientation Histogram) [MS05]  
Use log-polar binning structure.

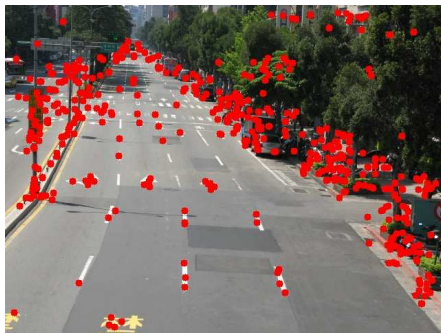


[MS05] found that GLOH performs the best, followed closely by SIFT.

Sample detected SURF keypoints (without non-maximal suppression):



(a)



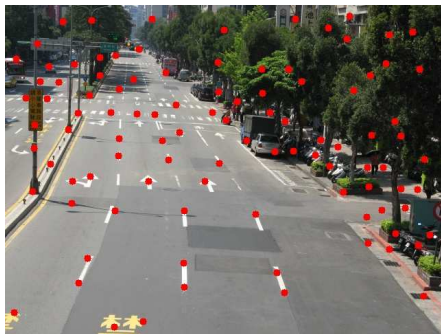
(b)

(a) Low threshold gives many cluttered keypoints.

(b) Higher threshold gives fewer keypoints, but still cluttered.

Sample detected SURF keypoints.

With adaptive non-maximal suppression, keypoints are well spread out.



(a)



(b)

(a) Top 100 keypoints.

(b) Top 200 keypoints

# Feature Matching

Measure difference as Euclidean distance between feature vectors:

$$d(\mathbf{u}, \mathbf{v}) = \left( \sum_i (u_i - v_i)^2 \right)^{1/2} \quad (23)$$

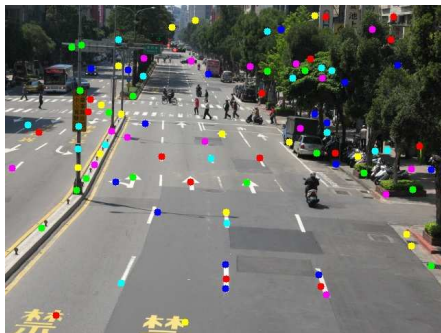
Several possible matching strategies:

- Return all feature vectors with  $d$  smaller than a threshold.
- Nearest neighbor: feature vector with smallest  $d$ .
- Nearest neighbor distance ratio:

$$\text{NNDR} = \frac{d_1}{d_2} \quad (24)$$

- $d_1, d_2$ : distances to the nearest and 2nd nearest neighbors.
- If NNDR is small, nearest neighbor is a good match.

Sample matching results: SURF, nearest neighbors with min. distance.



- Some matches are correct, some are not.
- Can include other info such as color to improve match accuracy.
- In general, no perfect matching results.

- Feature matching methods can give false matches.
- Manually select good matches.
- Or use **robust method** to remove false matches:
  - True matches are consistent and have small errors.
  - False matches are inconsistent and have large errors.
- Nearest neighbor search is computationally expensive.
  - Need efficient algorithm, e.g., using  $k$ -D Tree.
  - $k$ -D Tree is not more efficient than exhaustive search for large dimensionality, e.g.,  $> 20$ .

# Summary

- Harris corner detector and Tomasi's algorithm find corner points.
- SIFT keypoint: invariant to scale.
- SIFT descriptors: invariant to scale, orientation, illumination change.
- Variants of SIFT: PCA-SIFT, SURF, GLOH.

# Software Available

- SIFT code is available in Lowe's web site:  
[www.cs.ubc.ca/~lowe/keypoints](http://www.cs.ubc.ca/~lowe/keypoints)
- Available in OpenCV 2.1:
  - Corner detectors: Harris corner, Tomasi's good feature.
  - Subpixel corner location.
  - Feature descriptors: SURF, StarDetector.
- New in OpenCV 2.2:
  - Feature descriptors: FAST, BRIEF.
  - High-level tools for image matching.
- SciPy supports  $k$ -D Tree for nearest neighbor search.
- Fast nearest neighbor search library:  
[mloss.org/software/view/143/](http://mloss.org/software/view/143/)
- Approximate nearest neighbor search library:  
[www.cs.umd.edu/~mount/ANN/](http://www.cs.umd.edu/~mount/ANN/)





# Exercise

- (1) Derive the auto-correlation matrix  $\mathbf{A}$  given in Eq. 9.

# Further Reading

- Subpixel location of corner: [BK08] p. 319–321.
- Orientation and affine invariance: [Sze10] Section 4.1.1.
- SIFT: [Low04]
- SURF: [BTVG06]
- GLOH: [MS05]
- Adaptive non-maximal suppression: [Sze10] Section 4.1.1, [BSW05].

# Reference I

-  Bradski and Kaehler.  
*Learning OpenCV: Computer Vision with the OpenCV Library*.  
O'Reilly, 2008.
-  M. Brown, R. Szeliski, and S. Winder.  
Multi-image matching using multi-scale oriented patches.  
In *Proc. CVPR*, pages 510–517, 2005.
-  H. Bay, T. Tuytelaars, and L. Van Gool.  
SURF: Speeded up robust features.  
In *Proc. ECCV*, pages 404–417, 2006.
-  C. Harris and M. J. Stephens.  
A combined corner and edge detector.  
In *Alvey Vision Conference*, pages 147–152, 1988.

## Reference II



Y. Ke and R. Sukthankar.

PCA-SIFT: a more distinctive representation for local image descriptors.

In *Proc. CVPR*, pages 506–513, 2004.



D. G. Lowe.

Distinctive image features from scale-invariant keypoints.

*Int. J. Computer Vision*, 60:91–110, 2004.



K. Mikolajczyk and C. Schmid.

A performance evaluation of local descriptors.

*IEEE Trans. on PAMI*, 27(10):1615–1630, 2005.

# Reference III



J. Shi and C. Tomasi.

Good features to track.

In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

<http://citeseer.nj.nec.com/shi94good.html>,

<http://vision.stanford.edu/public/publication/>.



R. Szeliski.

*Computer Vision: Algorithms and Applications*.

Springer, 2010.