Local Features

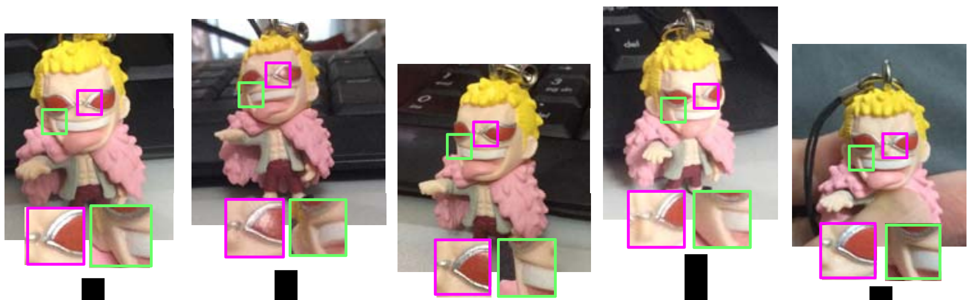
Global vs. Local Features

Feature extract from entire image = Global Features

Consider the global features extracted from these images

Global vs. Local Features

Feature extract from some parts of image = Local Features



Consider the local features extracted from these images

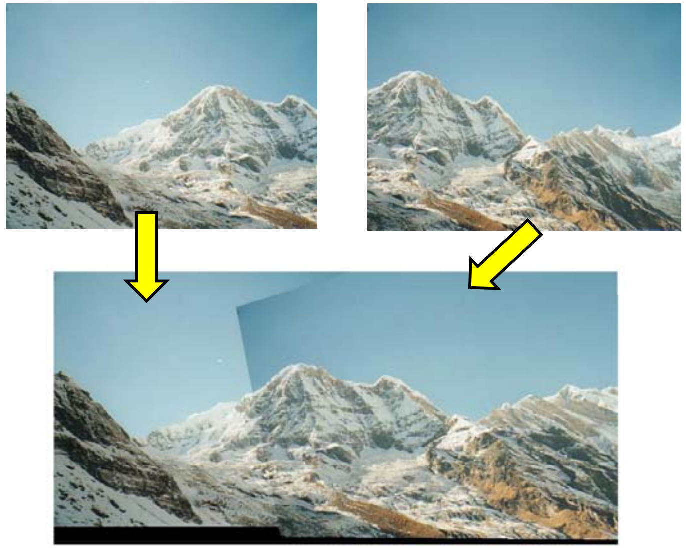
Advantages Local Features

- Locality:
  - Features are local, so robust to occlusion and clutter
- Distinctiveness:
  - Can differentiate a large database of objects
- Quantity
  - Hundreds or thousands in a single image
- Efficiency
  - Real-time performance achievable
- Generality
  - Exploit different types of features in different situations

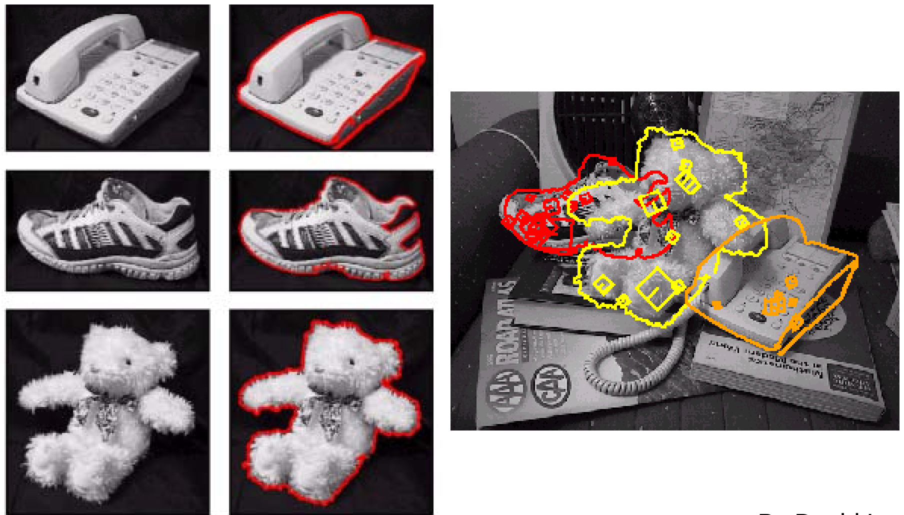
Applications Using Local Features

- Image alignment (e.g., image stitching)
- Object recognition
- Motion tracking
- 3D reconstruction
- Depth estimation stereo images
- Indexing and database retrieval
- Robot navigation
- … other

Image Stitching



Object Recognition

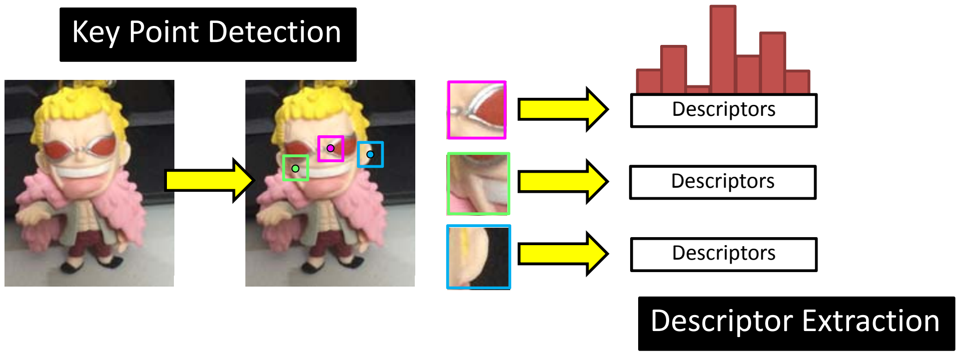


By David Lowe

Image Stitching

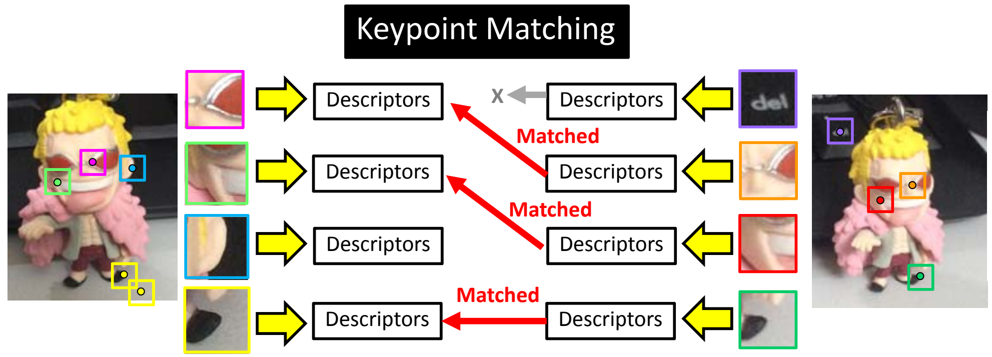
http://www.cs.toronto.edu/~kyros/courses/2503/Handouts/features.pdf

Local Feature Based Methods



- Feature Point Detection (Keypoint Detection)
- Feature Descriptor Extraction
- Feature Point Matching

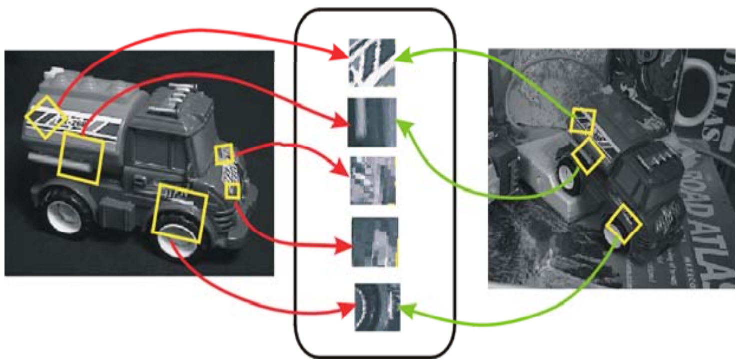
Local Feature Based Methods



- Feature Point Detection (Keypoint Detection)
- Feature Descriptor Extraction
- Feature Point Matching

Local Features

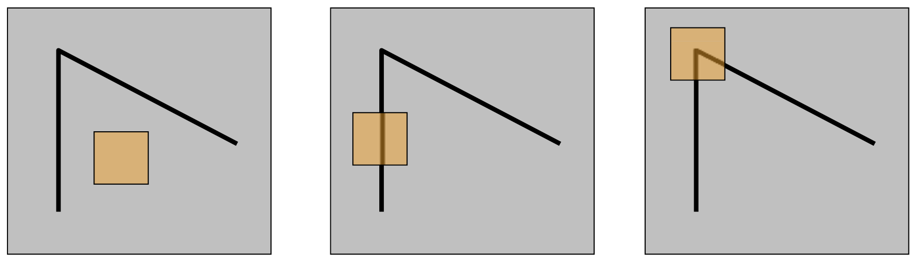
- Find features that are invariant to transformations
  - Geometric invariance: translation, rotation, scale
  - Photometric invariance: brightness, exposure,...



Feature Descriptors

Feature Points [Keypoints]

- Suppose we only consider a small window of pixels
  - What define whether a feature point is a good or bad candidate ?



Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature Points [Keypoints]

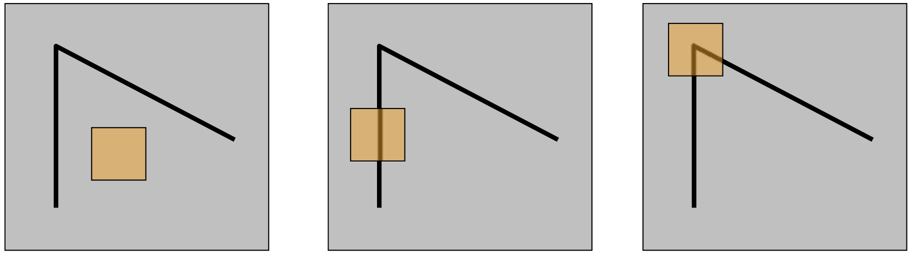
Local Measures of Uniqueness

- Local measure of "feature uniqueness"
  - How does the window change when you shift it?
  - Shifting the window in *any direction* causes a big change

"Flat" region: no change in all directions

"Edge": no change along the edge direction

"Corner": significant change in all directions

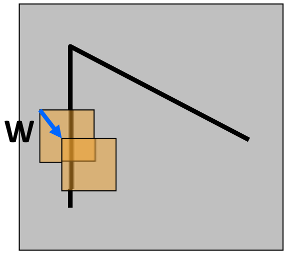


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Keypoint Detection

Consider shifting the window **W** by **(*u*,*v*)**

- How do the pixels in **W** change?
- Compare each pixel before and after by summing up the squared differences (SSD)
- This defines an SSD "error" of $E(u,v)$



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

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Keypoint Detection

- Taylor series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

- If the motion **(*u*,*v*)** is small, then first order approximation is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial u}v$$
$$\approx I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

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

- Plugging this into the formula on the previous slide...

Keypoint Detection

$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} \left[I(x,y) + [I_x \; I_y]\begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)\right]^2 \\
&\approx \sum_{(x,y)\in W} \left[[I_x \; I_y]\begin{bmatrix} u \\ v \end{bmatrix}\right]^2 \\
&\approx \sum_{(x,y)\in W} [u \; v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}
$$

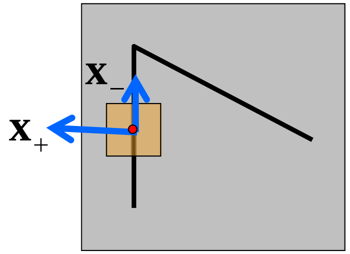
$$
E(u,v) \approx [u \;\; v] \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
$$

Keypoint Detection

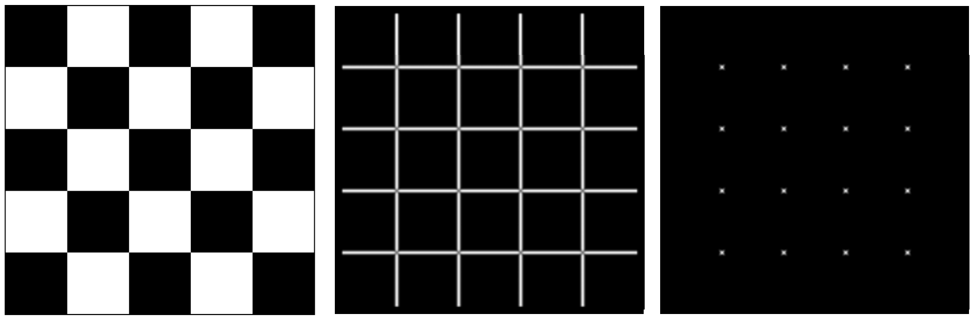
$$
H = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix} \Rightarrow \boxed{\begin{matrix} \mathbf{x}_+, \lambda_+ \\ \mathbf{x}_-, \lambda_- \end{matrix}} \quad (\lambda_+ \geq \lambda_-)
$$

Eigenvalues and eigenvectors of H

- $\mathbf{x}_+$ = Direction $\begin{bmatrix} u \\ v \end{bmatrix}$ of largest increase in $E(u,v)$
- $\mathbf{x}_-$ = Direction $\begin{bmatrix} u \\ v \end{bmatrix}$ of smallest increase in $E(u,v)$
- λ_+ = Amount of increasing in direction x_+
- λ_- = Amount of increasing in direction x_-



Keypoint Detection

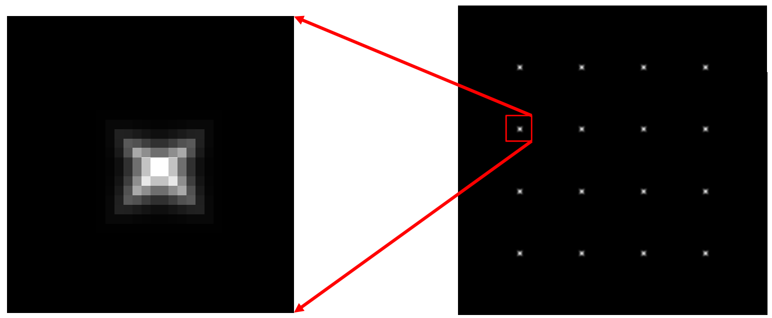


I $\qquad$ λ_+ $\qquad$ λ_-

Keypoint Detection

Here's what to do

- Compute the gradient at each point in the image
- Create the *H* matrix from the entries in the gradient
- Compute the eigenvalues
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as feature points



λ_-

Other Corner Measure Functions

Harris Detector

$$M_H = \det(H) - \alpha[\mathrm{trace}(H)]^2$$
$$= (1+2\alpha)\lambda_-\lambda_+ + \alpha(\lambda_-^2 + \lambda_+^2)$$

α is a tuning parameter (= 0.04 in the original paper)

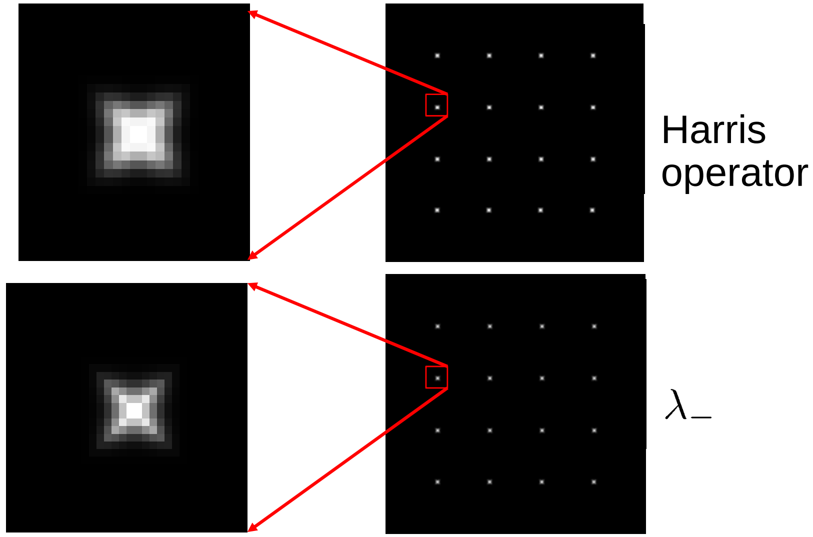
Noble Detector

$$M_N = \frac{\det(H)}{\mathrm{trace}(H)+\varepsilon} = \frac{\lambda_-\lambda_+}{\lambda_- + \lambda_+ + \varepsilon}$$

ε is a small number to aviod zero division

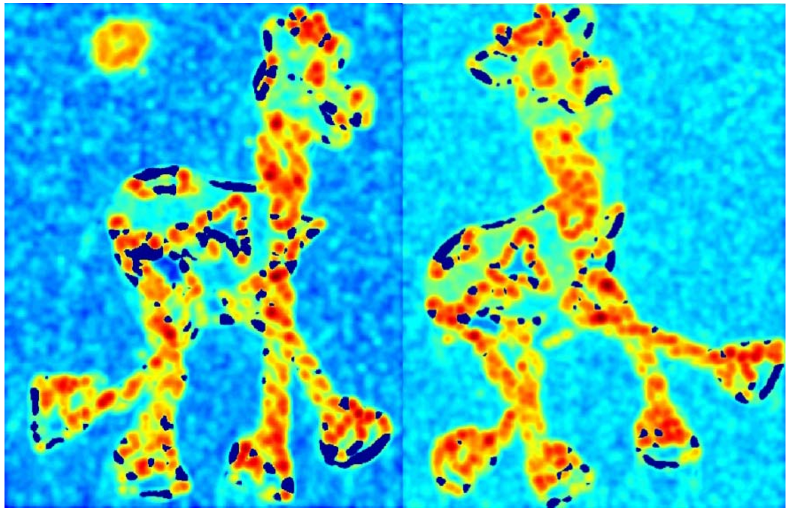
- $\mathrm{trace}(H) = h_{11} + h_{22}$
- Very similar to λ_- but less expensive (no square root)

Harris Operator

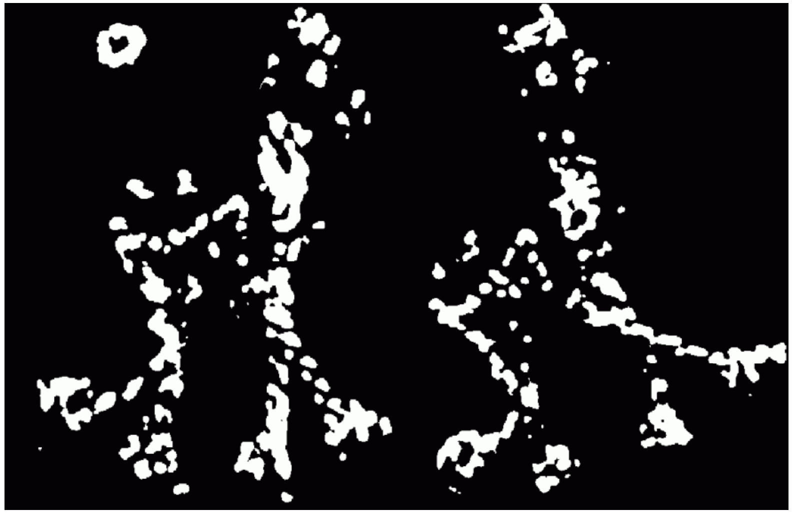


Keypoint Detection Example

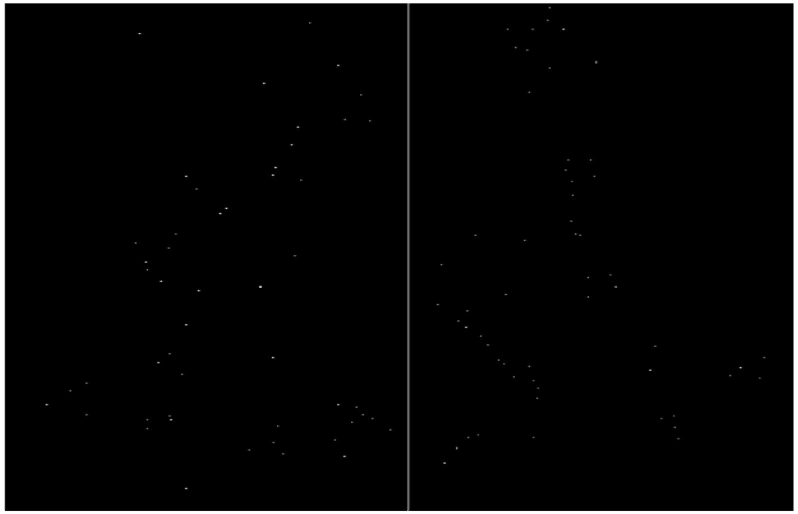
Keypoint Detection Example



Keypoint Detection Example



Keypoint Detection Example



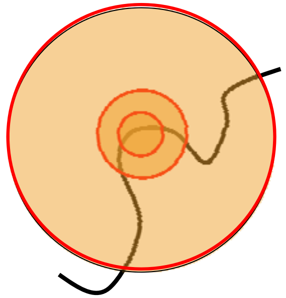
Keypoint Detection Example

Keypoint Detection Invariance?

- Suppose you **rotate** the image by some angle
  - Will you still get the same features ?
- What if you change the brightness?
- How about scale ?

Scale Invariant Detection

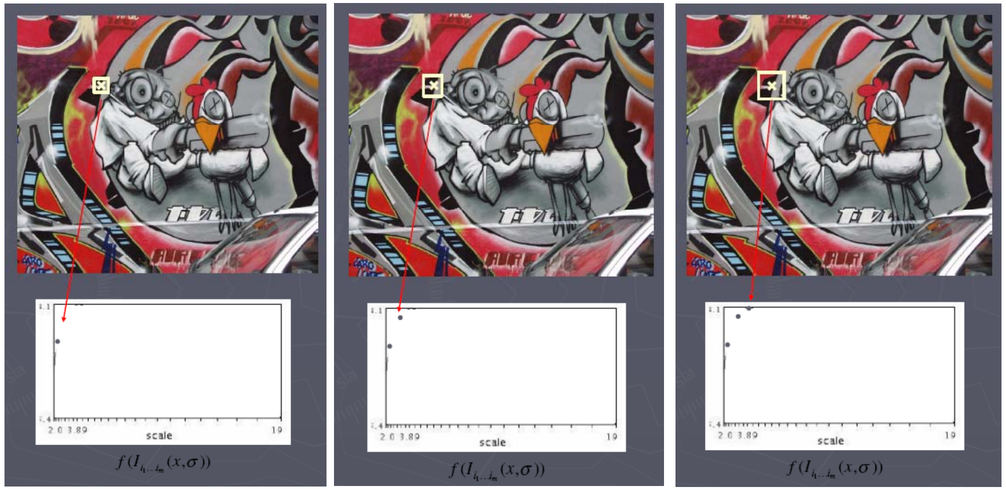
- Suppose you're looking for corners



- Key idea: find scale that gives local maximum of f
  - f is a local maximum in both position and scale
  - Common definition of f: Laplacian

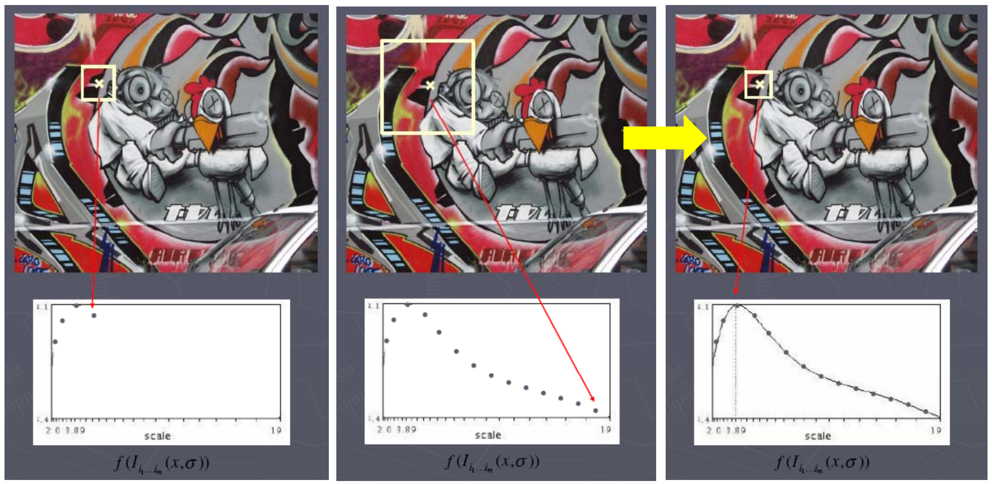
  (or difference between two Gaussian filtered images with different sigmas)

Automatic Scale Selection



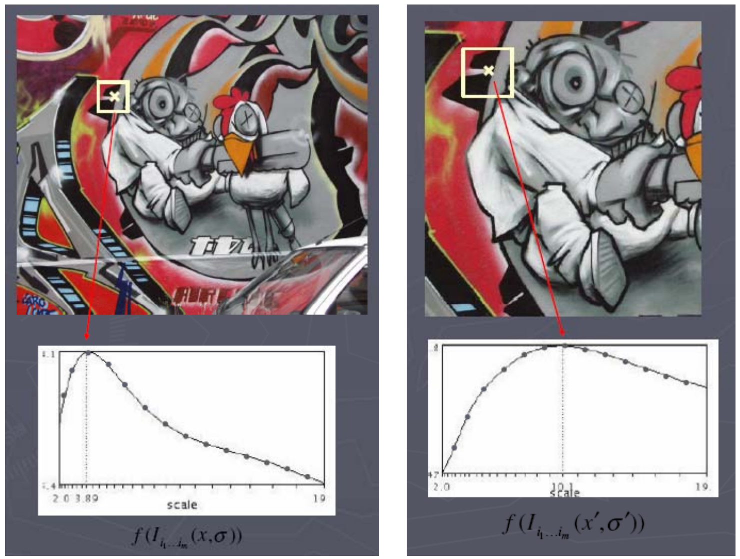
Lindeberg et al., 1996

Automatic Scale Selection



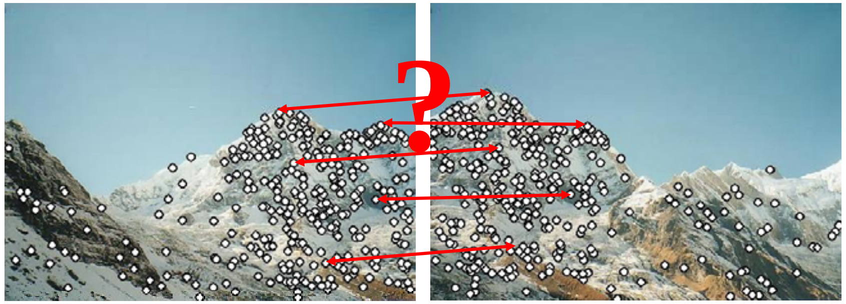
Lindeberg et al., 1996

Automatic Scale Selection



Keypoint Descriptors

- We know how to detect good points
- Next question: How to match them ?



- Lots of possibilities (this is a popular research area)
  - Simple option: match square windows around the point
  - State of the art approach: SIFT, SURF, BRIEF.

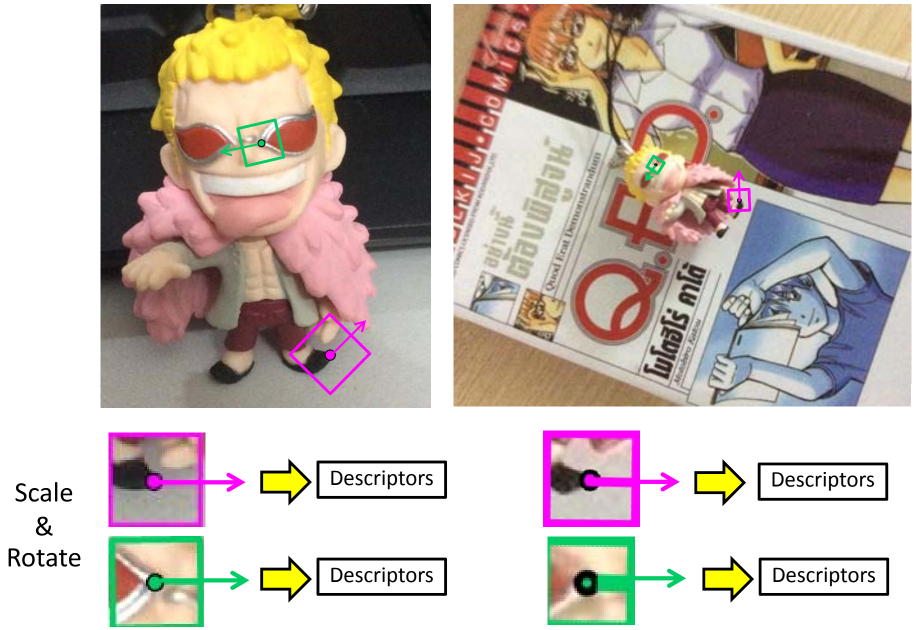
Invariant Descriptors

- Suppose we are comparing two images I_1 and I_2
  - I_2 may be a transformed version of I_1
  - What kinds of transformations are we likely to encounter in practice?
- We'd like to find the same feature regardless of the transformation
  - This is called transformational invariance
  - Most feature methods are designed to be invariant to
    - Translation, 2D rotation, scale
  - They can usually also handle
    - Limited 3D rotations (SIFT works up to about 60 degrees)
    - Limited affine transformations (some are fully affine invariant)
    - Limited illumination/contrast changes

Rotation Invariance for Feature Descriptors

- Find dominant orientation of the image patch. This is given by $\mathbf{x}_+$ (the eigenvector of H corresponding to the larger eigenvalue λ_+)
- Rotate the patch according to this angle
- Extract descriptor from the patch

Figure by Matthew Brown

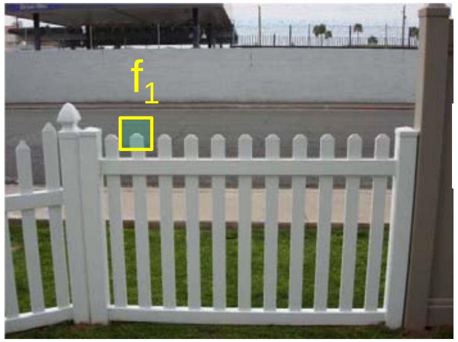


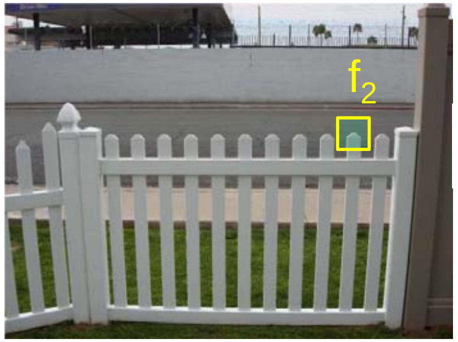
Keypoint Matching

- Given a keypoint in I_1, how to fine the best match in I_2 ?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with minimum distance

Distance Function

- How to define the difference between two features f_1 and f_2 ?
  - Simple approach is $SSD(f_1, f_2)$
    - Sum of square differences between entries of the two descriptors
    - Can give good scores to very ambiguous (bad) matches





Distance Function

- How to define the difference between two features f_1 and f_2 ?
  - Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
    - f_2 is the best SSD match to f_1 in I_2
    - f_2' is the 2nd best SSD match to f_1 in I_2
    - Gives small values for ambiguous mactches