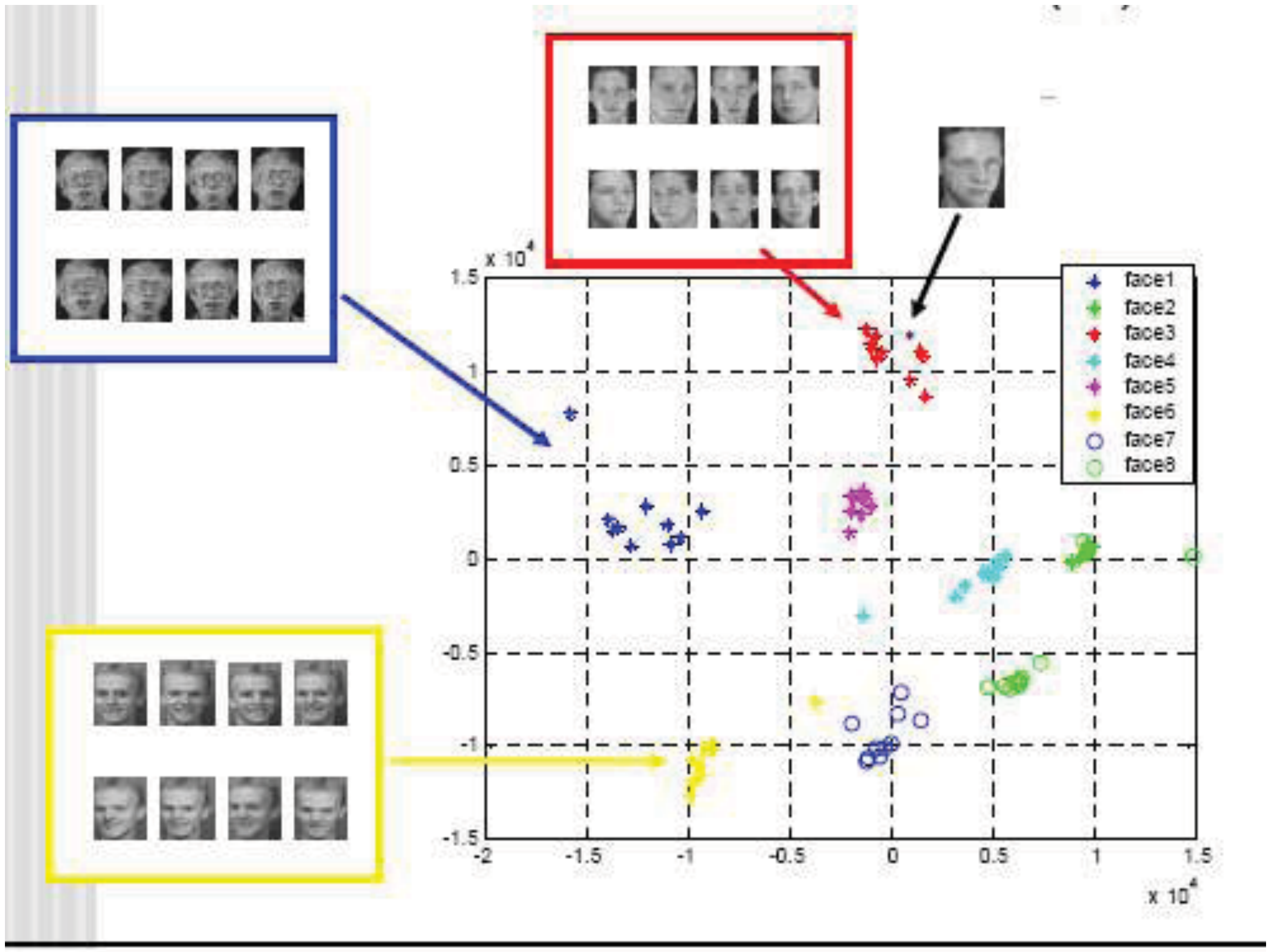
Outline

- Pattern recognition in computer vision
- Background on the development of SIFT
- SIFT algorithm and some of its variations
- Computational considerations (SURF)
- Potential improvement
- Summary

Pattern Recognition

- **Pattern recognition is a classical problem in computer vision. It consists of two crucial steps: feature extraction and pattern matching. Our focus today is on feature extraction.**
- **Features are effective and efficient representations of a pattern that are discriminative and invariant**
  - Discriminative: similar objects are close to each other in the feature space
  - Invariant: values remain stable under normal variation of conditions.
- **Scale (focal length) change is frequent and normal to any camera-based application. A person has no problem in identifying object with different focal length. A useful camera app needs to be scale-invariant.**
- **Ideally, features should also be photometric (illumination) invariant and geometrical (rotational, affine, or perspective).**

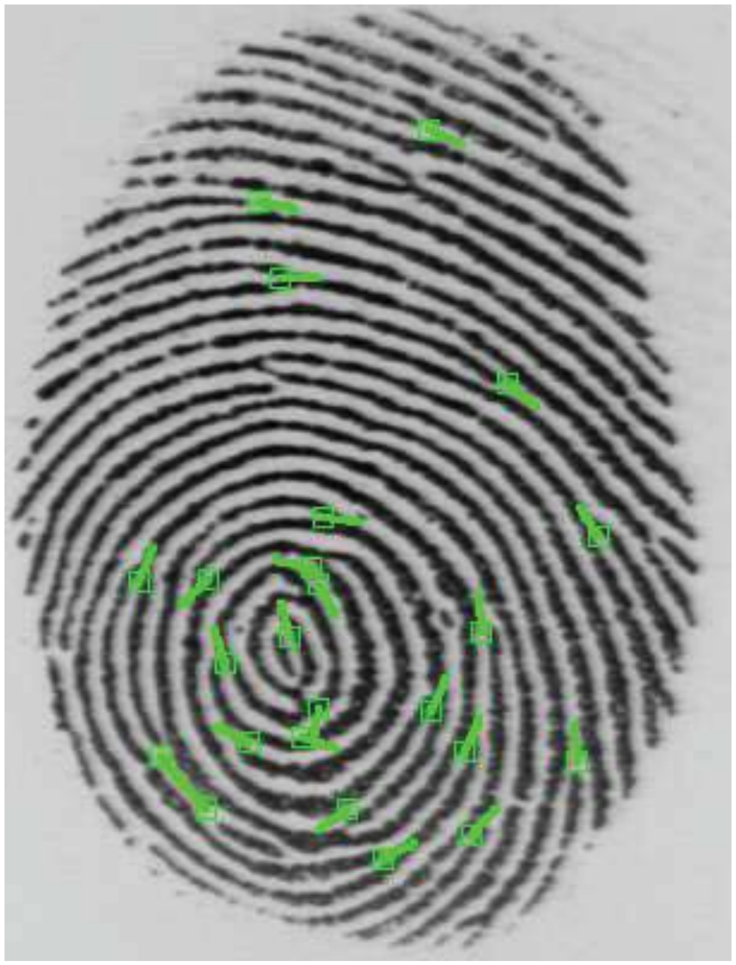
Example on Discriminative Features



Methods on Feature Extraction

- Features could be global or local. Global means features are derived based on the entire image. Eigen-face (or Fisher-face) is a good example of global feature.
- It has tractable mathematics with straightforward algorithm (although could be MIPS intensive). It is, however, subject to variations due to multiple factors: geometrical, photometrical, and partial occlusion to name a few. It only works well under well controlled conditions.
- Local features are features intrinsic to an object and can be computed using patches of images. The minutia (breaks and branching points), for example, are effective local features for fingerprint recognition.
- Local features could be difficult to compute; but has several advantages.

Fingerprint Recognition

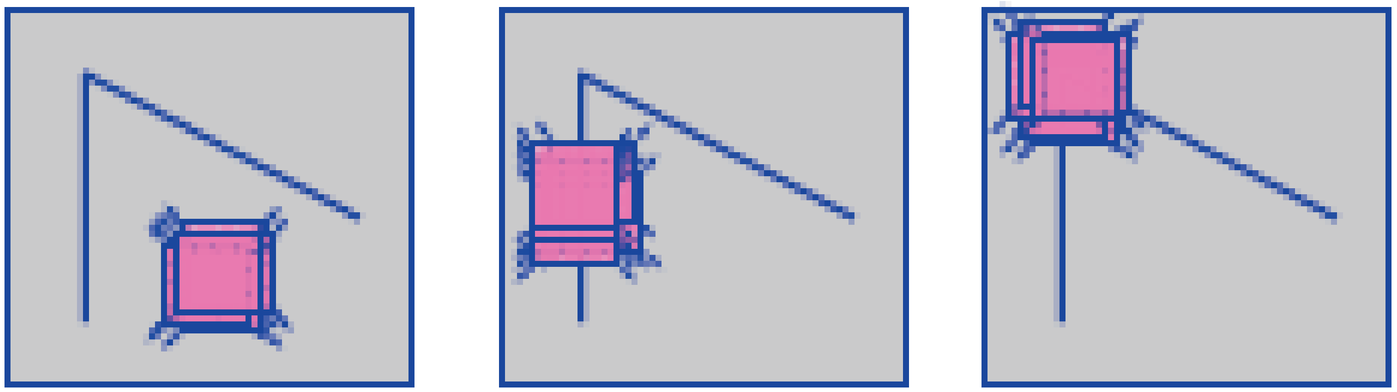


Additional Advantages of Local Features

- Locality: features are local, so robust to occlusion and clutter.
- Distinctiveness: individual features can be matched to a large database of objects.
- Quantity: many features can be generated for even a small object.
- Efficiency: close to real-time performance.
- Corners together with a description of its neighboring structure appear to be a excellent candidate as local feature for general object recognition.
- Corner provide a stable focal point whereas its neighboring structures provide some uniqueness of an object.

Harris Corner Detector

- SIFT is primarily originated from the Harris corner detector.
- Harris corner detector is correlation based:



"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Computing Harris Corner

- **Auto-SSD for a point (x,y) and a shift (Δx,Δy) is computed as**

$$f(x,y) = \sum_{(x_k,y_k)\in W} (I(x_k,y_k) - I(x_k+\Delta x, y_k+\Delta y))^2$$

$$\text{with} \quad I(x_k+\Delta x, y_k+\Delta y) = I(x_k,y_k) + (I_x(x_k,y_k) \quad I_y(x_k,y_k)) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$f(x,y) = \sum_{(x_k,y_k)\in W} \left((I_x(x_k,y_k) \quad I_y(x_k,y_k)) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right)^2$$

$$= (\Delta x \quad \Delta y) \begin{bmatrix} \sum_{(x_k,y_k)\in W} (I_x(x_k,y_k))^2 & \sum_{(x_k,y_k)\in W} I_x(x_k,y_k) I_y(x_k,y_k) \\ \sum_{(x_k,y_k)\in W} I_x(x_k,y_k) I_y(x_k,y_k) & \sum_{(x_k,y_k)\in W} (I_y(x_k,y_k))^2 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

$$= [\Delta x \ \Delta y] C(x,y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

where *C(x,y)* is an effective representation for corner determination.

Corner Determination

Let λ_1, λ_2 be the eigenvalues of matrix $C(x, y)$. The eigenvalues form a rotationally invariant description. There are three cases to be considered:

1. If both λ_1, λ_2 are small, so that the local auto-correlation function is flat (i.e., little change in $c(x, y)$ in any direction), the windowed image region is of approximately constant intensity.

2. If one eigenvalue is high and the other low, so the the local auto-correlation function is ridge shaped, then only local shifts in one direction (along the ridge) cause little change in $c(x, y)$ and significant change in the orthogonal direction; this indicates an edge.

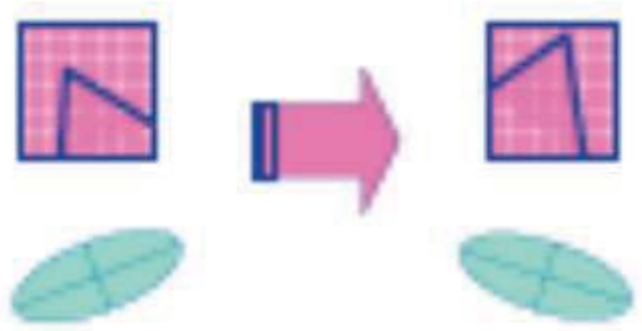
3. If both eigenvalues are high, so the local auto-correlation function is sharply peaked, then shifts in any direction will result in a significant increase; this indicates a corner.

Computer eigen values are difficult. A quick measure on corner strength is:

$$\det(\mathbf{C}) - \alpha \, \text{trace}^{2}(\mathbf{C})$$

Harris Corner: Rotational Invariant

- Rotation invariance

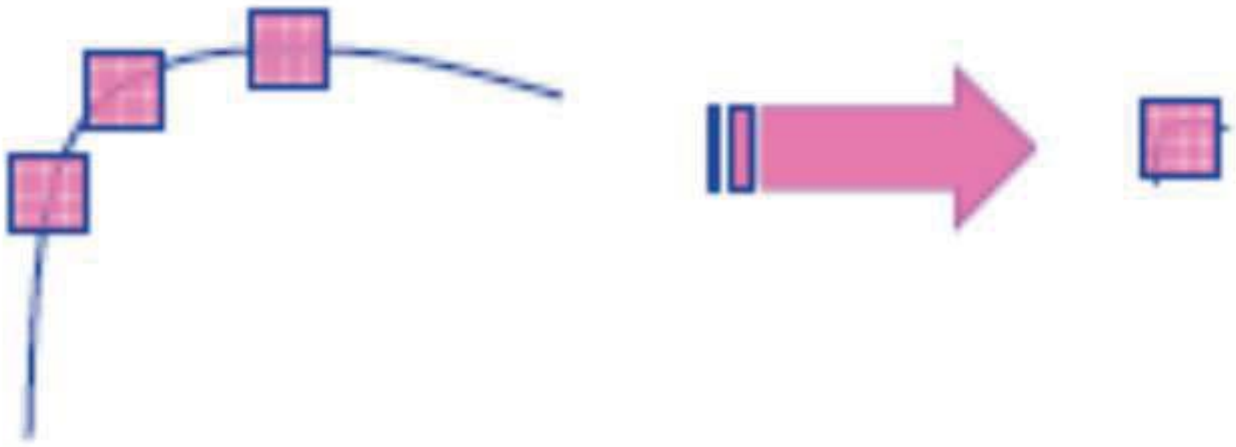


Ellipse rotates but its shape (i.e., eigenvalues) remains the same

Corner response R is invariant to image rotation

Harris Corner: Not Scale Invariant

- But not invariant to *image scale*

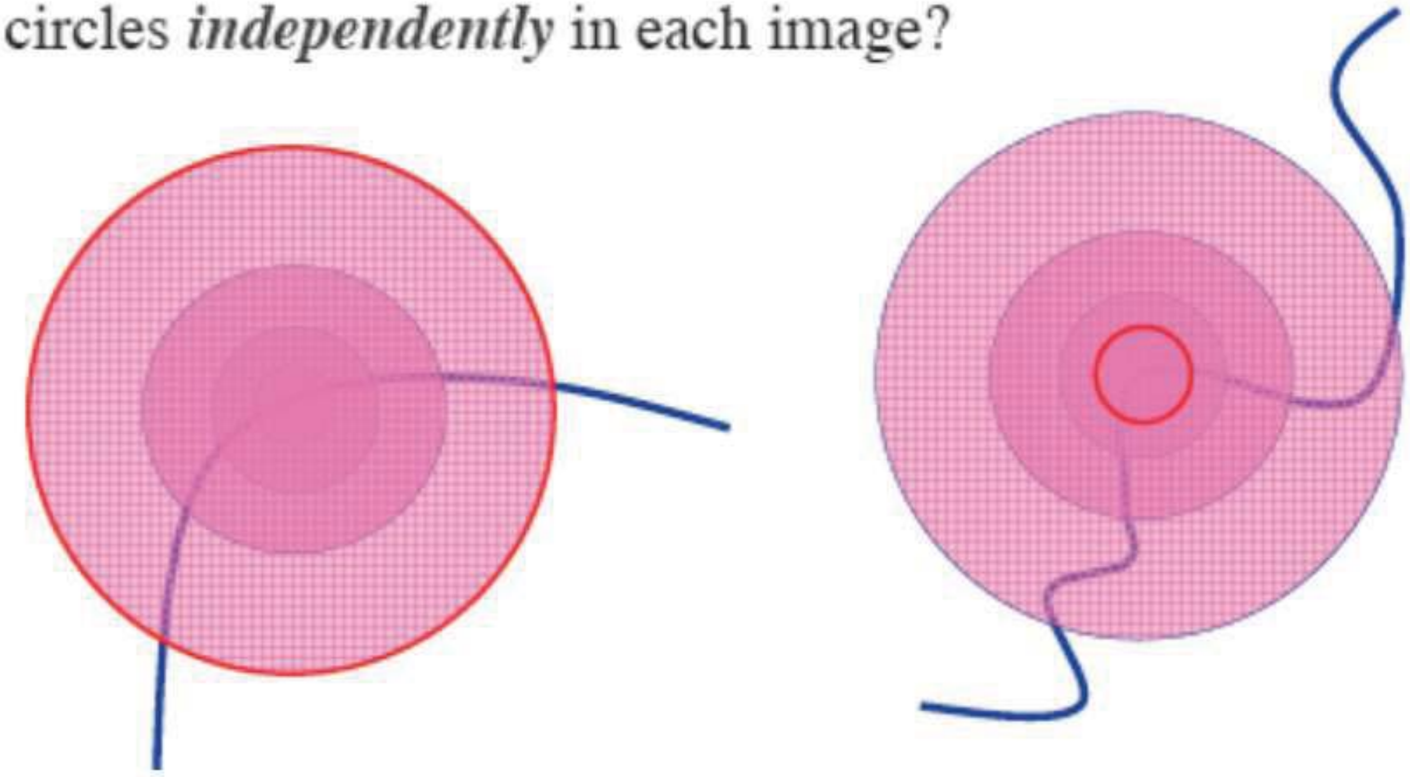


Fine scale: All points will be classified as edges

Coarse scale: Corner

Harris Corner: The Problem

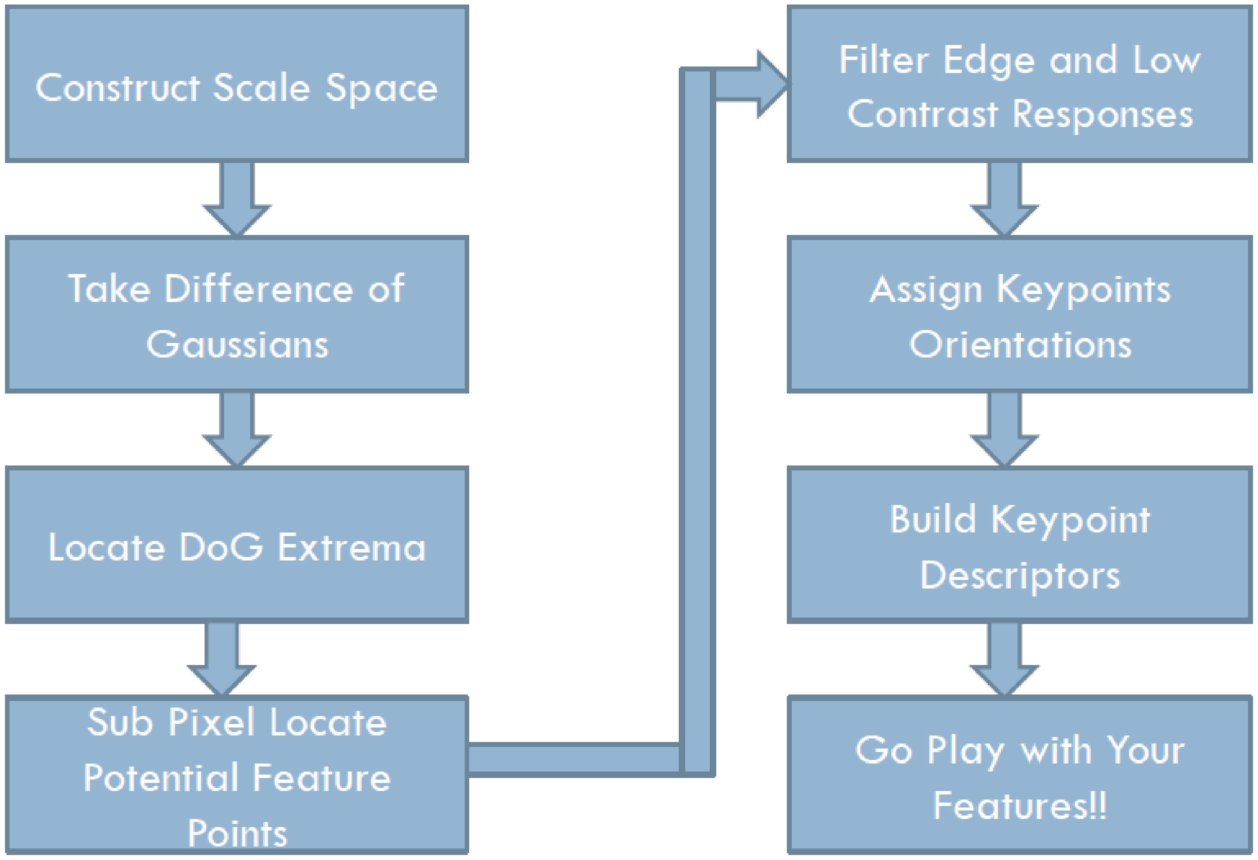
- The problem: How do we choose corresponding circles *independently* in each image?



SIFT

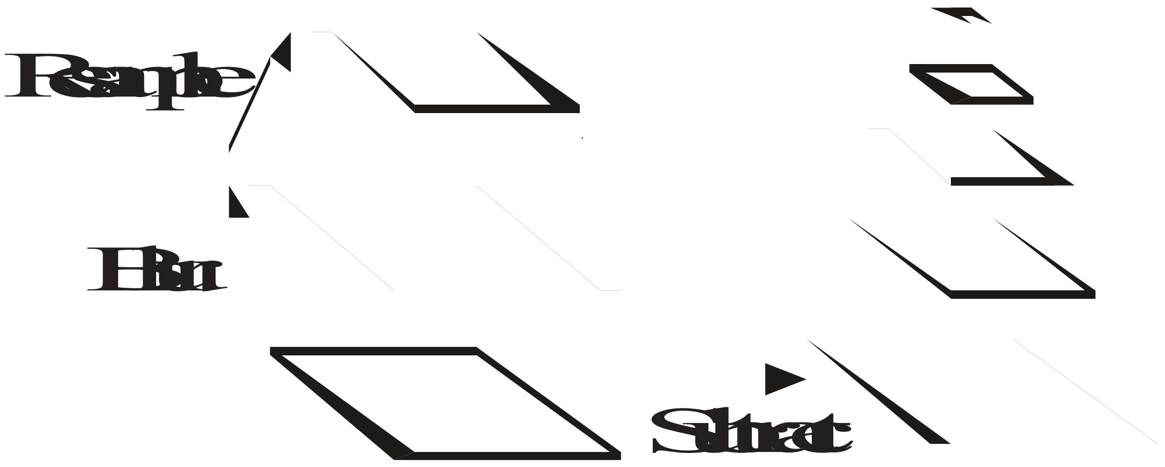
- **SIFT resolves the problem of Harris corner with two major improvements.**
- **First, a scale space representation is built to facilitate the search for features (corners) at a distinctive scale**
- **Second, difference of Gaussian (DoG) was used to replace the computation of *C(x,y)***
  - It is computationally efficient for building scale space representations
  - It approximates Laplacian of Gaussian (LoG) and its extremes also provide stable anchor points (such as Harris corners) for feature extraction.
- **This results from the heat equation** $\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G.$ **We will come back to this point later on.**

Overview of SIFT


Construct Scale Space
Take Difference of Gaussians
Locate DoG Extrema
Sub Pixel Locate Potential Feature Points
Filter Edge and Low Contrast Responses
Assign Keypoints Orientations
Build Keypoint Descriptors
Go Play with Your Features!!

Construct Scale Space

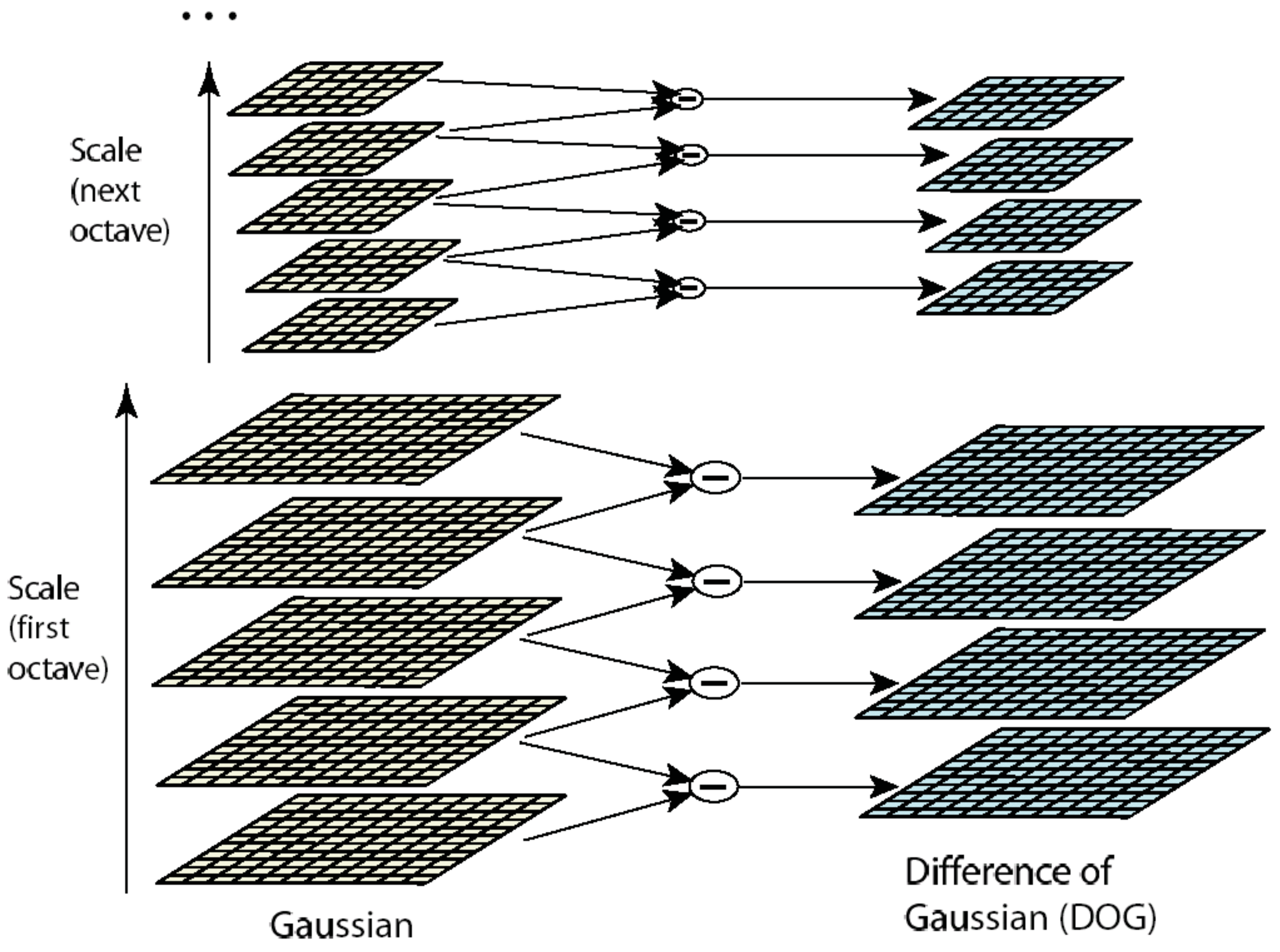
- All scales must be examined to identify scale-invariant features. An efficient way is to construct the Difference of Gaussian (DoG) pyramid:



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

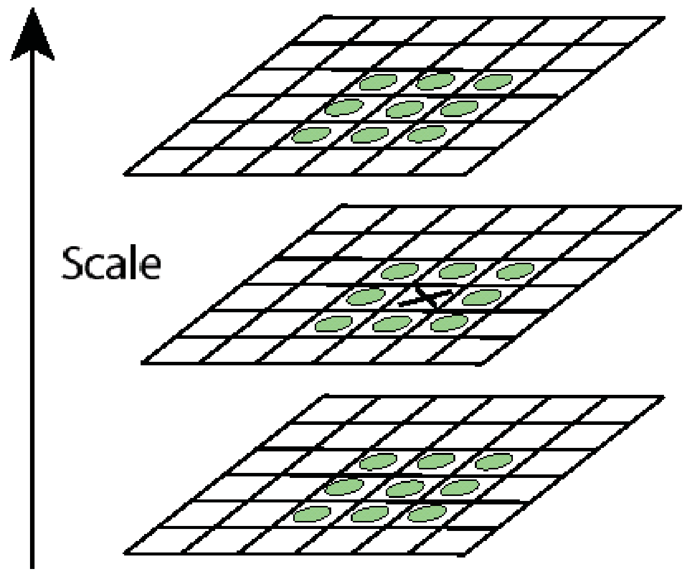
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Take Difference of Gaussian



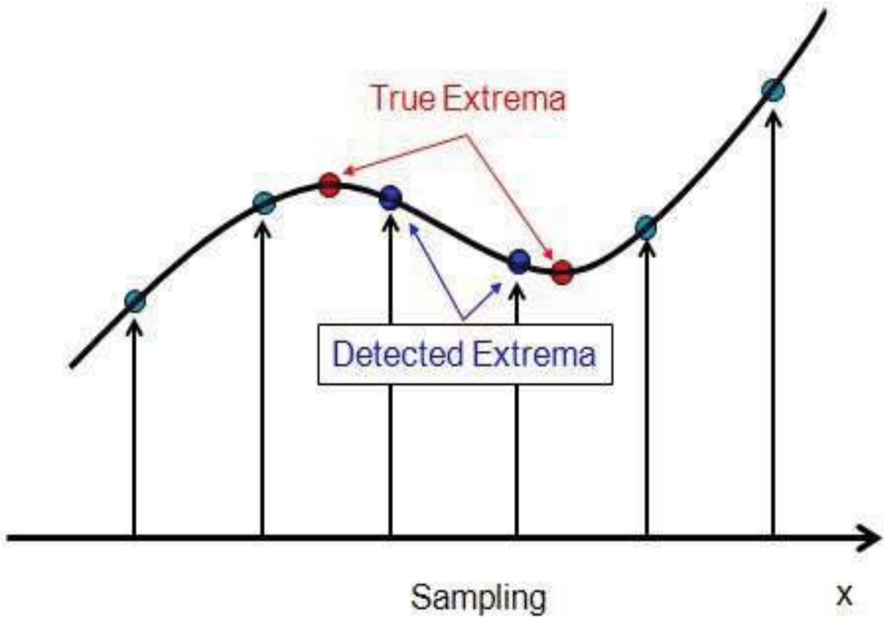
Locate DoG Extrema

- Detect maxima and minima of difference-of-Gaussian in scale space
- These extrema are potential candidates for features and referred to as key-points or interest-points.



Sub-Pixel Localization

- The problem



- The solution

-Take Taylor series expansion

$$D(\vec{x}) = D + \frac{\partial D^T}{\partial \vec{x}} \vec{x} + \frac{1}{2} \vec{x}^T \frac{\partial^2 D^T}{\partial \vec{x}^2} \vec{x}$$

$$\hat{x} = -\frac{\partial^2 D}{\partial \vec{x}^2}^{-1} \frac{\partial D}{\partial \vec{x}}$$

- Differentiate and set to 0

-to get location in terms of (x,y,σ)

Keypoint Filtering - Low Contrast

- Low Contrast Points Filter

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^{T}\hat{\mathbf{x}}.$$

$D(\hat{x})$ smaller than 0.03 will be discarded because it is likely to be generated by noise.

Eliminating Edge Responses

- **Edge also has high response along one direction; but is not a useful feature.**
- **Use eigen values of Hessian matrix to eliminate edges:**

$$\mathrm{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

  - Eigenvalues are proportional to principle curvatures
  - The ratio between trace and determinant could identify stable features

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta, Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

Orientation Assignment

- Use scale of point to choose correct image:

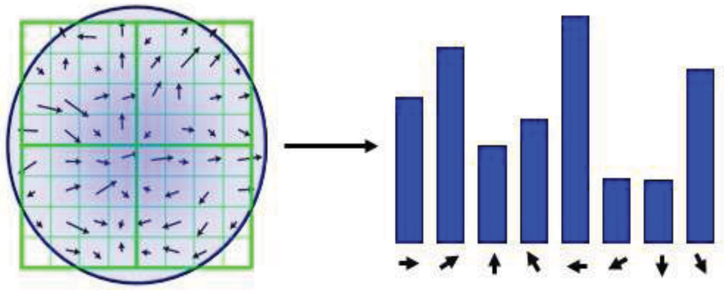
$$L(x,y) = G(x,y,\sigma) * I(x,y)$$

- Compute gradient magnitude and orientation using finite differences:

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

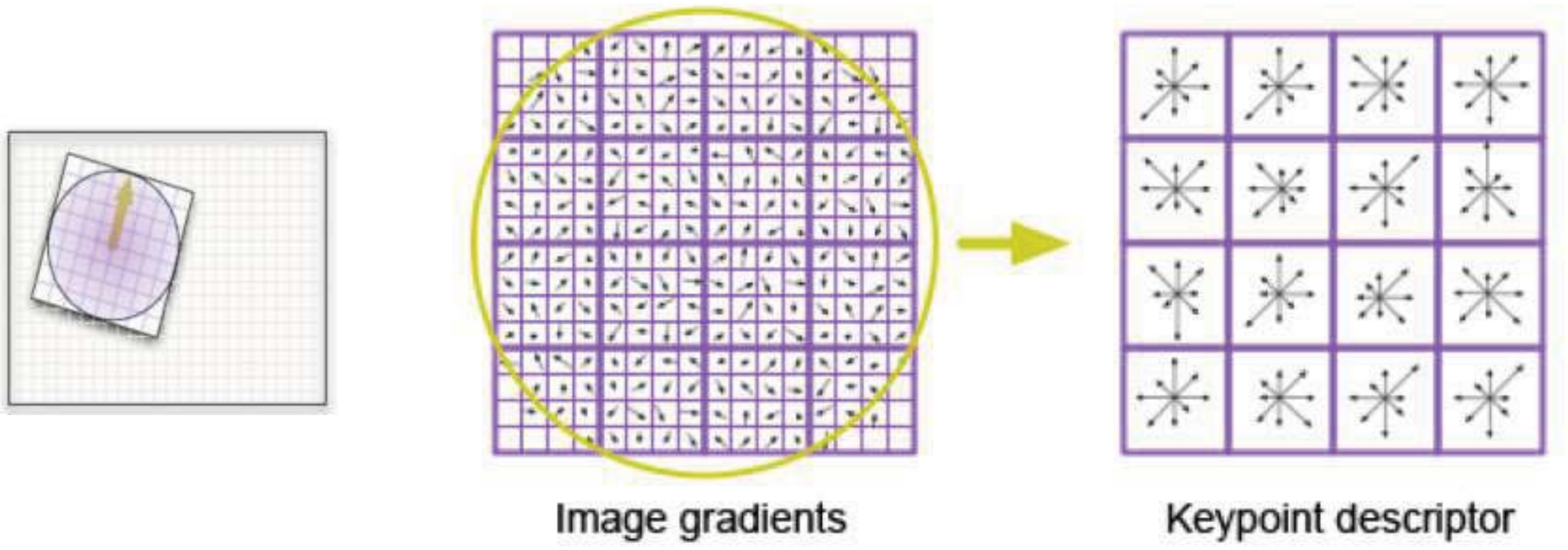
$$\theta(x,y) = \tan^{-1}\left(\frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-1,y))}\right)$$

- Use directional histogram



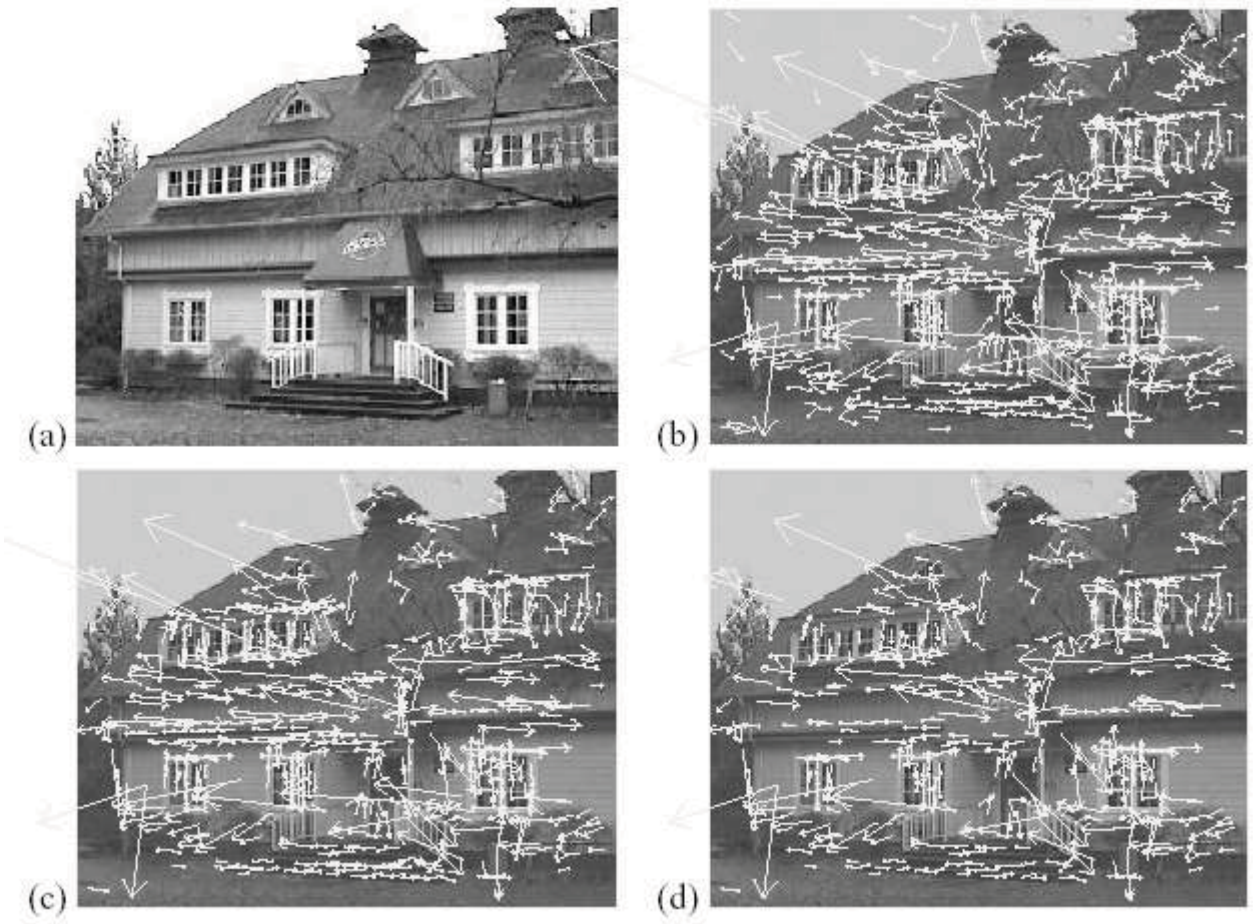
Compute Final SIFT Vector

- **Actual implementation uses 4x4 descriptors from 16x16 which leads to a 4x4x8=128 element vector**



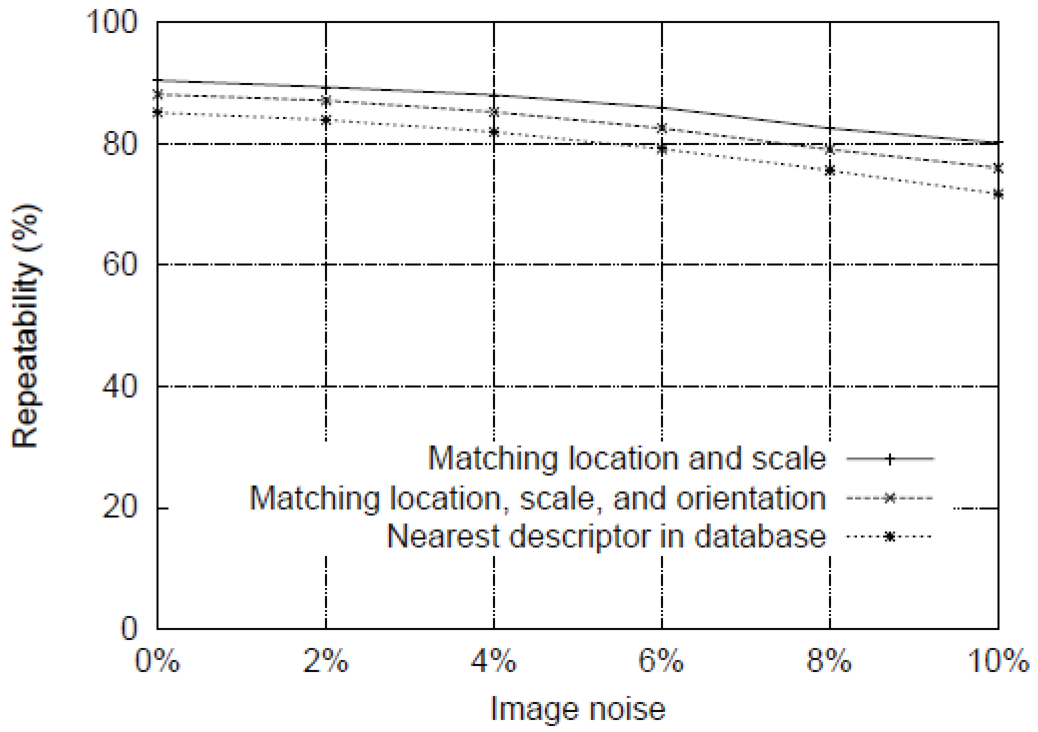
Example of Key-point Detection

Threshold on value at DOG peak and on ratio of principle curvatures (Harris approach)

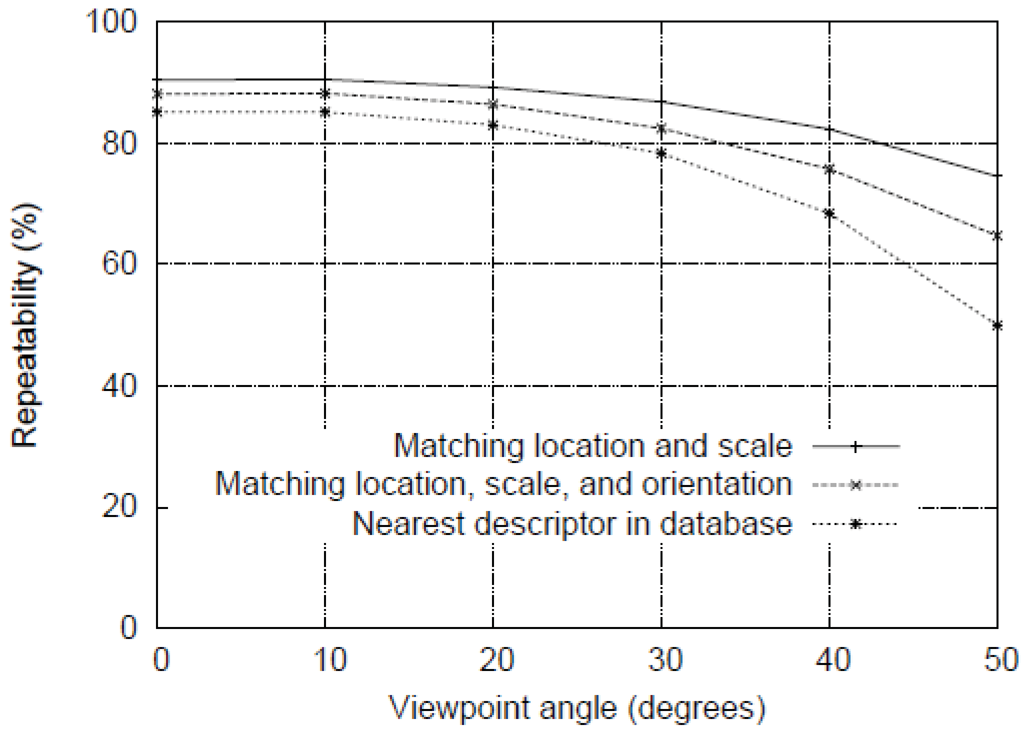


(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures

Repeatability against Noise



Repeatability against View Angles



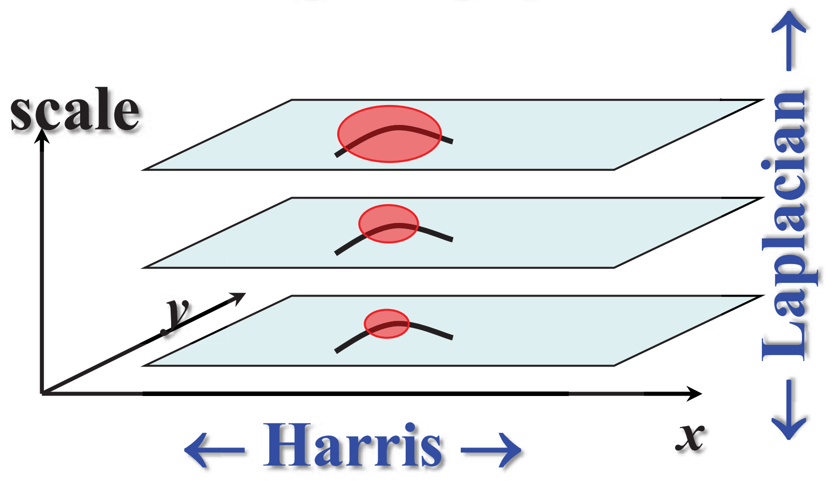
Variations of SIFT

- **SIFT uses DoG in both spatial and scale space operations. DoG is an approximation of LoG. Threshold and edge elimination are part of the computations.**
- **Alternatively, other derivative based functions or operators can also be used to compute scale space representation:**
  - SIFT $|I(\mathbf{x}) * G(s_{n-1}) - I(\mathbf{x}) * G(s_n)|$
  - Square gradient: $s^2(L_x^2(\mathbf{x}, s) + L_y^2(\mathbf{x}, s))$ $L(\mathbf{x}, s) = G(s) * I(\mathbf{x})$
  - Laplacian: $|s^2(L_{xx}(\mathbf{x}, s) + L_{yy}(\mathbf{x}, s))|$
  - Harris function: $\det(\mathbf{C}) - \alpha \operatorname{trace}^2(\mathbf{C})$

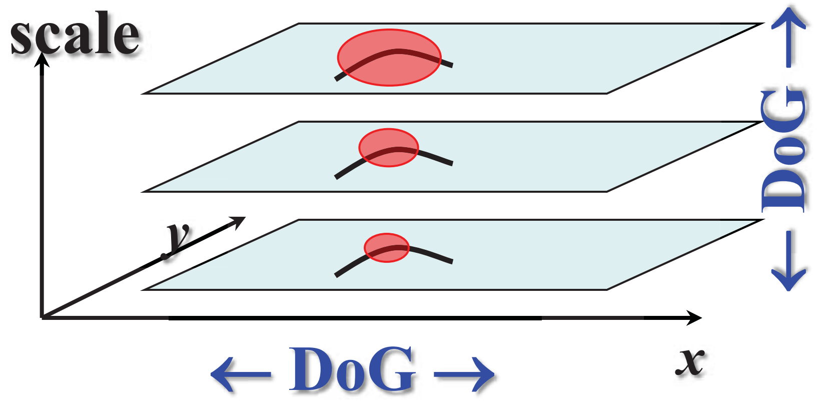
$$\text{with } \mathbf{C}(\mathbf{x}, s, \tilde{s}) = s^2 G(\mathbf{x}, \tilde{s}) \star \begin{bmatrix} L_x^2(\mathbf{x}, s) & L_x L_y(\mathbf{x}, s) \\ L_x L_y(\mathbf{x}, s) & L_y^2(\mathbf{x}, s) \end{bmatrix}$$

Two Alternatives on Finding Key-points

- **Harris-Laplacian**[1]
  Find local maximum of:
  - Laplacian in scale
  - Harris corner detector in space (image coordinates)

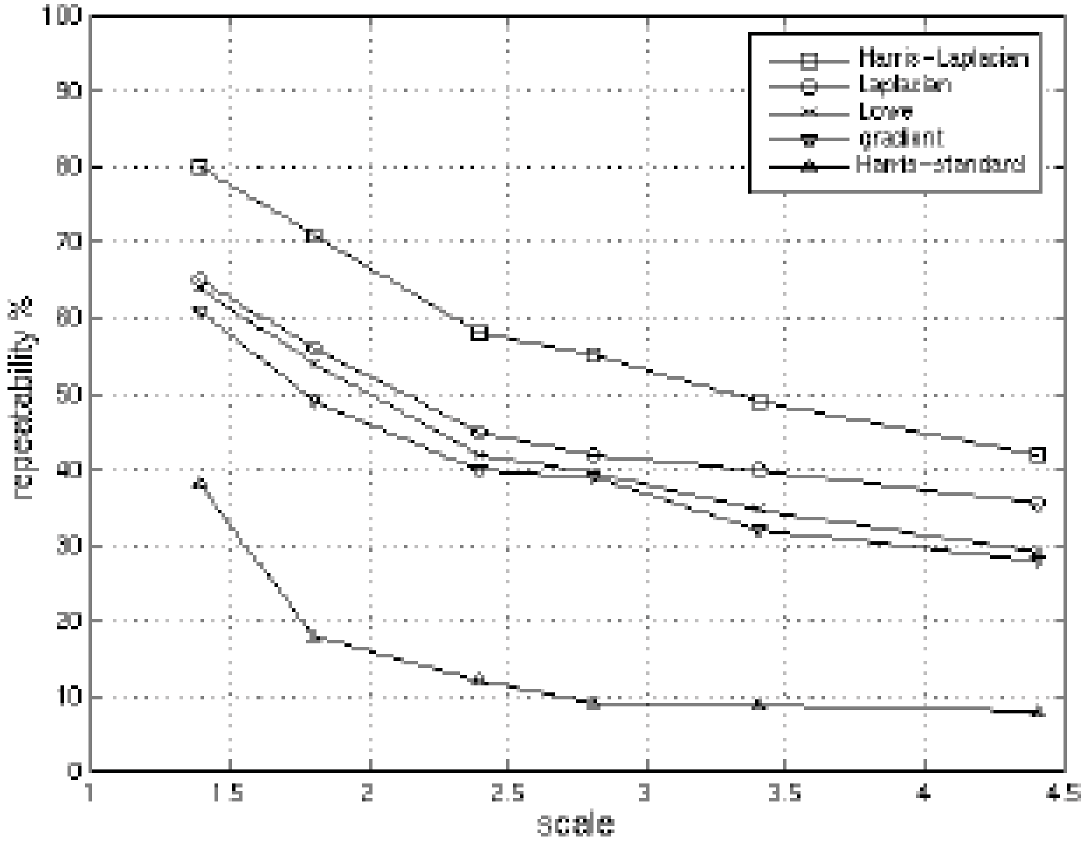


- **SIFT**[2]
  Find local maximum of:
  – Difference of Gaussians in space and scale



[1] K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001
[2] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 2004

Performance Comparisons



Some Computational Considerations

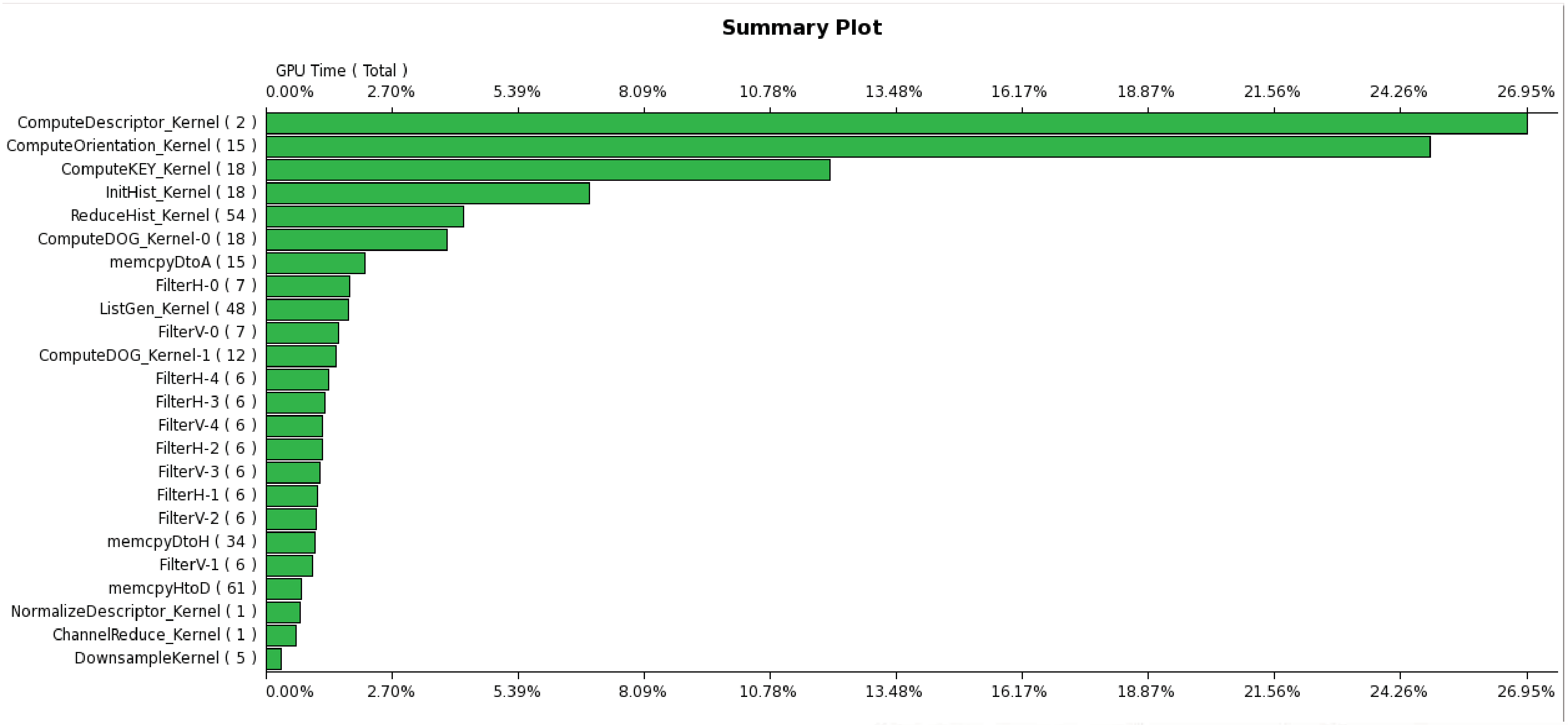
- Discrete implementation of SIFT provides opportunity for the development of fast algorithm.
- The Gaussian kernel for computing derivatives, for example, need to be quantized and windowed.
- A coarse quantization with square windowing can help to save MIPS by change convolution to additions.
- Speed Up Robust Features (SURF) made successful use of the property for the computation of Hessian Matrix

$$\mathrm{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

needed for the detection of extrema.

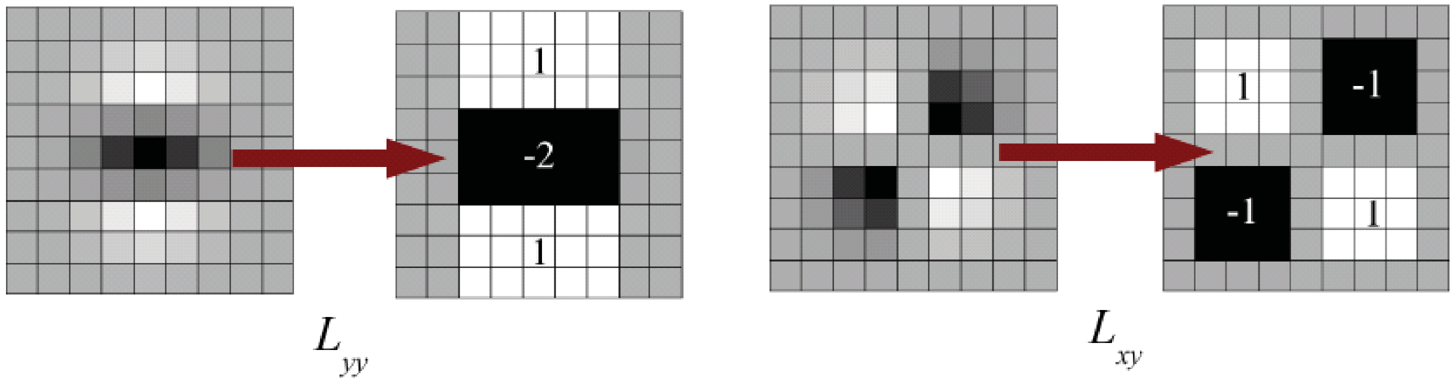
Computational Profile on SIFT

- 8800GTX(800*600)



Approximating the Gaussian Kernel

- Approximated second order derivatives with box filters for the computation of 2nd order derivatives (Fast-Hessian).



Comparisons of Different Algorithms

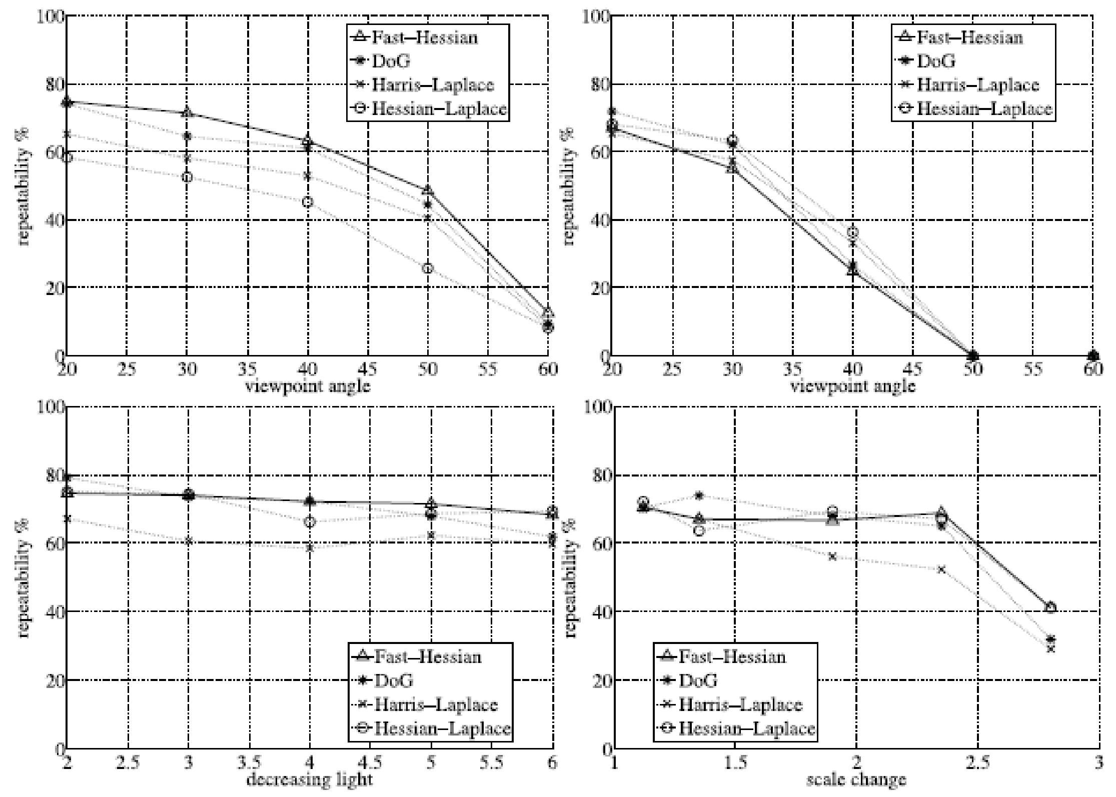


Fig. 6. Repeatability score for image sequences, from left to right and top to bottom, Wall and Graffiti (Viewpoint Change), Leuven (Lighting Change) and Boat (Zoom and Rotation).

Potential Improvements on SIFT

- The vector representation is a heuristic scheme and could be replaced by more rigorous vector-generation method. PCA, for example, could be used to compute the canonical orientation.
- The algorithm is build on intensity image only. Color image introduces further math sophistications.
- SIFT is affine invariant (covariant); it can tolerate only small, insignificant perspective variations.
- SIFT makes use the solutions of linear equation of heat diffusion. An evolution of non-linear diffusion could potentially minimizes the need to search in scale space.

Summary

- We did a tutorial review on the development of SIFT in pattern recognition.
- The focus is on SIFT; but we also briefly reviewed several variations of SIFT.
- We also quickly covered one computation scheme - SURF.
- Potential work on improving SIFT is discussed.
- This tutorial mainly talks about feature extraction. Pattern matching using SIFT is not covered.