Detectors and Descriptors

P J Narayanan

CS5765. Computer Vision. Spring 2013

CVIT, IIIT, Hyderabad

Recognition Strategies

- Problem: Distinguish between day and night images. Or, distinguish between outdoor and indoor images
  - Solution: Use global features describing the image.

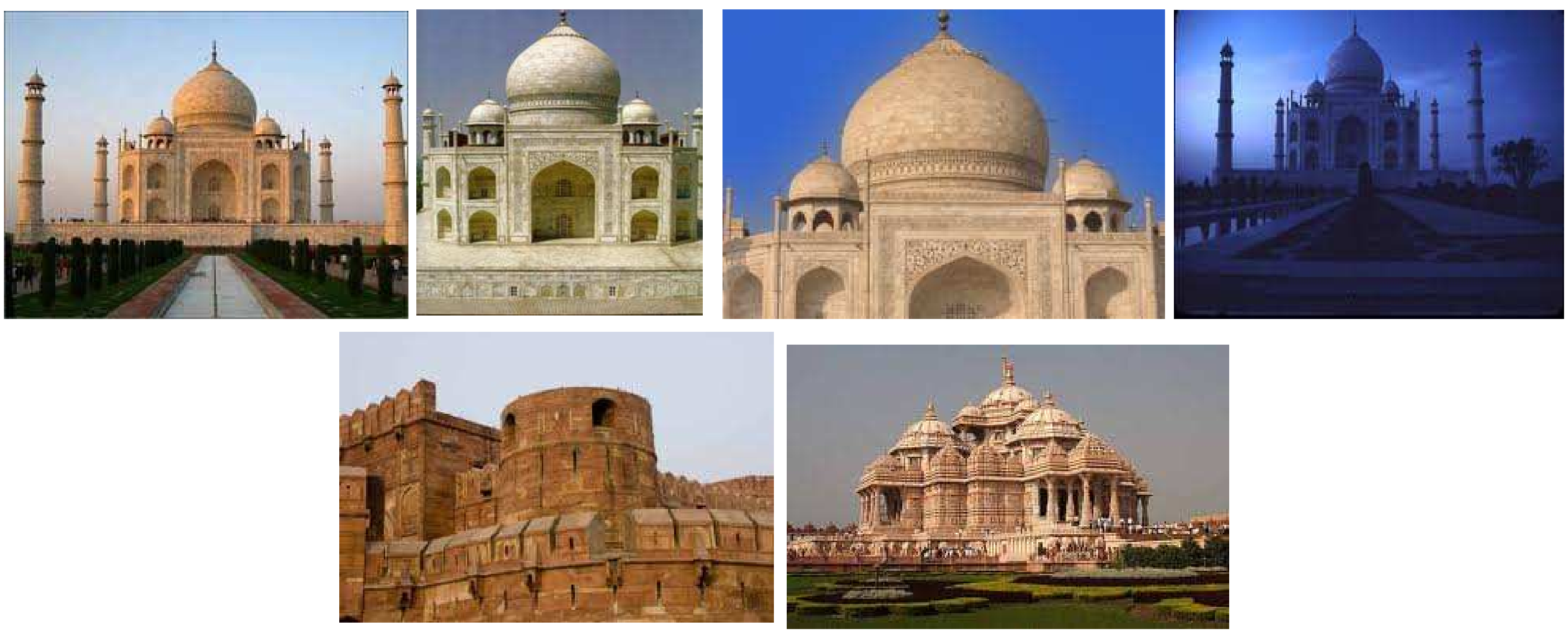
    Such as colour histogram, amount of edges, etc.
  - *Is this document about cricket?*
- Problem: Recognize special objects like faces, person.
  - Solution: Design specialized detectors
  - *Mark sentences about a wicket falling*
- We want to search **our keywords** on Google
- Can we look for *local* information (like words in documents) for search and/or recognition?

Flexible Recognition/Retrieval

- How do we do more flexible recognition?
- Applications: Several
  - Searching for similar images. Give an image as the query to a search engine!
  - Distinguish between scenarios. Say chair images from tables. Or between individuals, etc.
- Strategy: find **interest points** and **descriptors**
  - Find “interesting” and “unique” features in image
  - Use their combination for high level reasoning and recognition.
- Like words in a document, enabling key-word search

Interest Points: Requirements

- Repeatable across different transformations, noise, color change, etc.
- Informative or discriminative towards the task. Should capture the essential information relevant to the task
- Efficient in terms of computation time and space

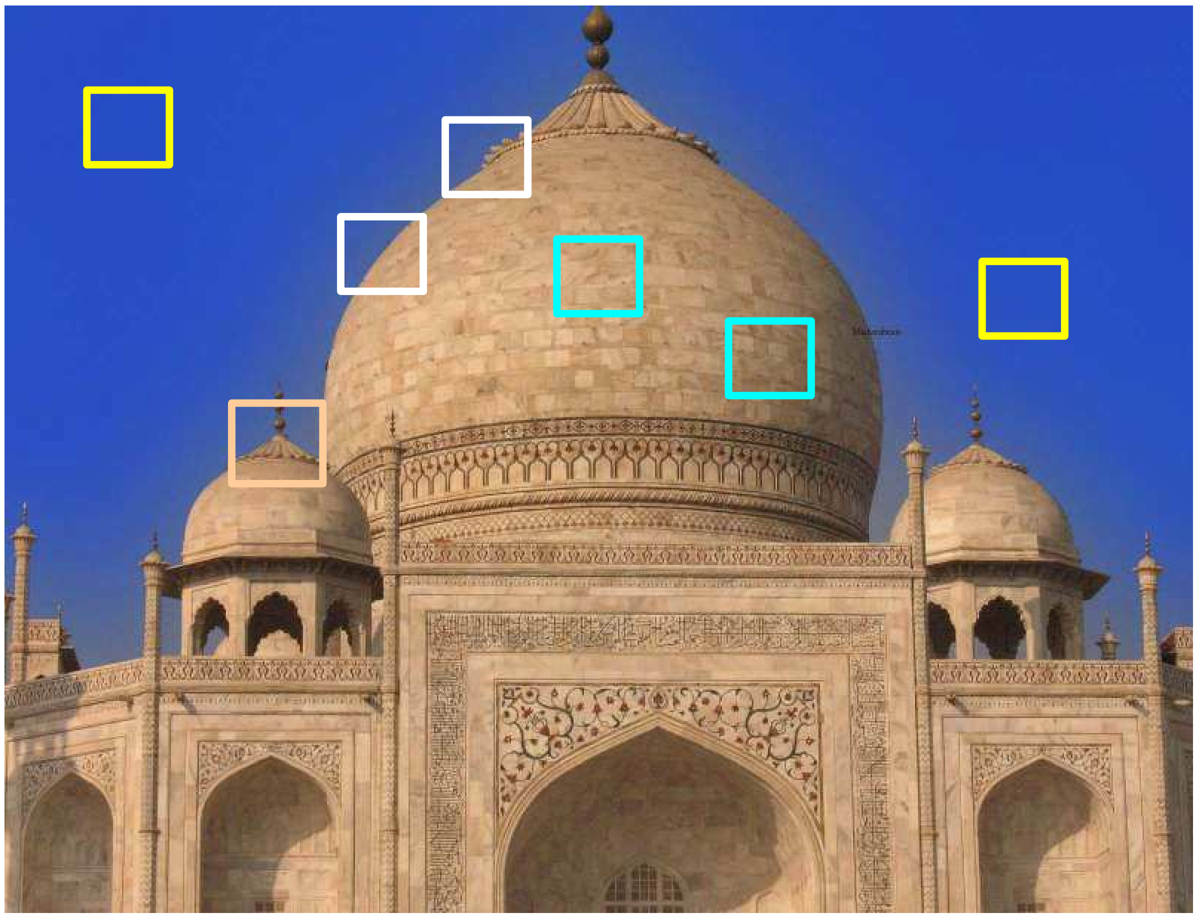


Interest Points

- What are the **important** elements in an image?
- Extract important elements and reason using them

Uniform Regions, Edges, and Corners

- Which points are **distinct** in the *local neighbourhood*?
- Which points are **distinct** in the *whole image*?



Detector & Descriptor

- Interest Point **Detector**: Where is the interest point?
  - Search the image and identify locations that qualify
  - What to look for? Big change in appearance that can be identified reliably across imaging scenarios
- **Descriptor**: What are the properties?
  - How do we describe it in a repeatable manner?
  - What properties to use? Color and appearance? Gradients or a measure of local changes? Texture pattern?

Corners

- Regions where movement in all direction cause serious change in appearance
- How does the appearance change with a small displacement $\vec{u} = [\Delta x \;\; \Delta y]^{\mathbf{T}}$ at (x, y)?
- $$D_{\vec{u}}(x,y) = \sum_{x,y} \sum_{\in W} w(x,y)\,[I(x+\Delta x, y+\Delta y) - I(x,y)]^2$$
  where $w(x,y)$ is a weighting funtion to give more importance to the centre. Gaussian in 2D works fine.
- Using Taylor expansion:
  $$I(x+\Delta x, y+\Delta y) = I(x,y) + I_x(x,y)\Delta x + I_y(x,y)\Delta y + \cdots$$

- Ignoring the quadratic terms and substituting in $D(\cdot)$

$$D_{\vec{u}}(x,y) = \sum_{x,y}\sum_{\in W} w(x,y)\,[I_x(x,y)\Delta x + I_y(x,y)\Delta y]^2$$

$$= \sum\sum_{W} w(x,y)\ \ [\Delta x \ \ \Delta y] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- $D_{\vec{u}}(x,y) = [\Delta x \ \ \Delta y]\ \mathbf{A}\ [\Delta x \ \ \Delta y]^{\mathbf{T}}$ where

$$\mathbf{A} = \sum\sum_{W} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$$

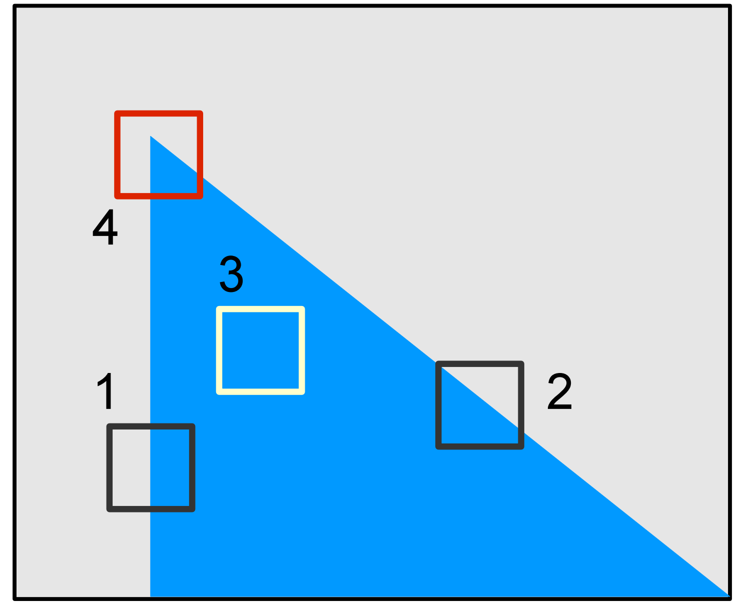
- $D(\cdot)$ should be high in all directions $\vec{u}$.
- Examine the eigenvalues of A. If both are high, $D(\cdot)$ will be high in all directions

Eigenanalysis

- $\mathbf{A} = [\mathbf{e_1} \;\; \mathbf{e_2}] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{e_1} \;\; \mathbf{e_2}]^{\mathbf{T}} = [\mathbf{e_1} \;\; \mathbf{e_2}] \mathbf{\Lambda} [\mathbf{e_1} \;\; \mathbf{e_2}]^{\mathbf{T}}$
- $\mathbf{x^T A x}$ is an ellipse in 2D.
- Eigenvectors define an orthonormal (rotation) matrix.
- $\mathbf{x^T A x} = ([\mathbf{e_1} \; \mathbf{e_2}]^{\mathbf{T}} \mathbf{x})^{\mathbf{T}} \mathbf{\Lambda} ([\mathbf{e_1} \;\; \mathbf{e_2}]^{\mathbf{T}} \mathbf{x}) = (\mathbf{Rx})^{\mathbf{T}} \mathbf{\Lambda} (\mathbf{Rx})$
- Eigenvectors define a rotation matrix. $\mathbf{Rx}$ orients the ellipse and the eigenvalues give the extents of its major and minor axes.

Harris Corner

- Get eigenvalues of the Harris matrix **A** at each location.
- Regions with large variation in all directions: High λ_1 and λ_2
- Regions with large variation in one direction: High λ_1 or λ_2
- Near constant regions: Low λ_1 and λ_2

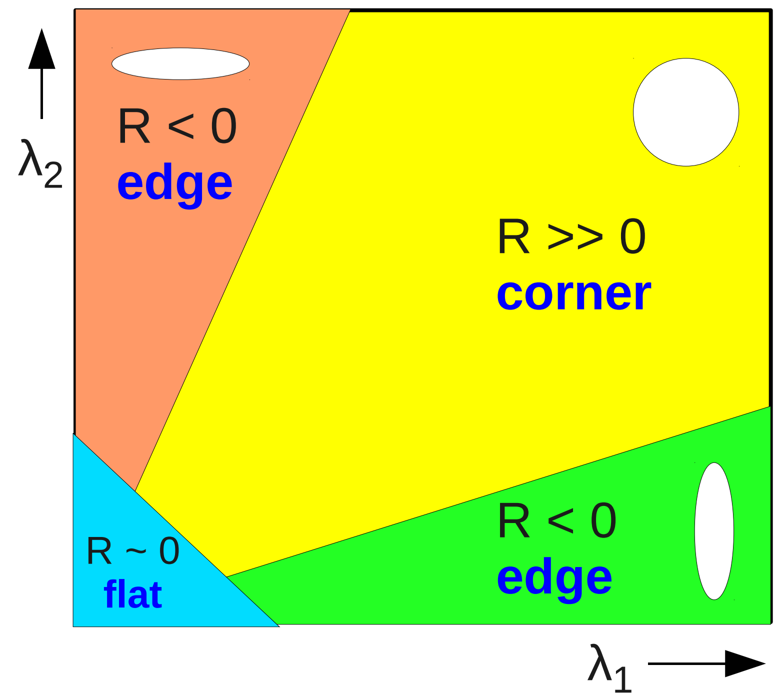


- We know that: $\text{trace}(A) = \lambda_1 + \lambda_2$ and $\det(A) = \lambda_1 \lambda_2$
- Define **cornerness** $R = \det(A) - k\ \text{trace}(A)^2$. Harris used k in the range $[0.04, 0.06]$
- $R < 0$ when only one eigenvalue is strong. $R >> 0$ for corneres, when both eigenvalues are strong

If $\lambda_1 = r\lambda_2$, $\frac{\det(A)}{\text{trace}(A)^2} = \frac{r\lambda_2^2}{(1+r)^2\lambda_2^2} = \frac{r}{(1+r)^2}$, a function of r only.

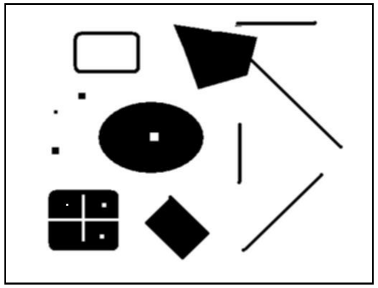
Corner Detection Algorithm

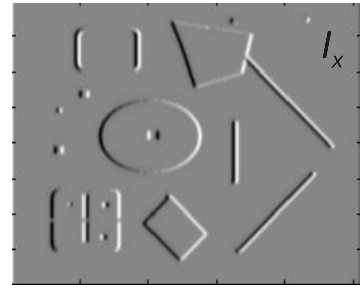
- Evaluate derivatives I_x, I_y, matrix $\mathbf{A}$ at all points based on a window and a weight function $w(x, y)$
- Evaluate R everywhere.
- Corner if $R > \tau$
- Find the maximum R in a 3×3 or 5×5 neighbourhood and declare it as a corner point.

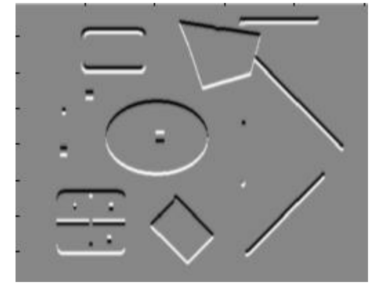


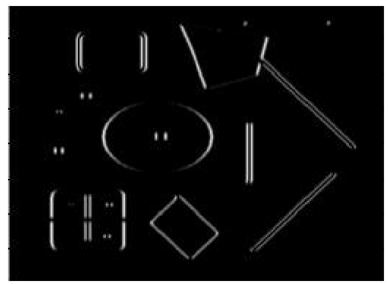
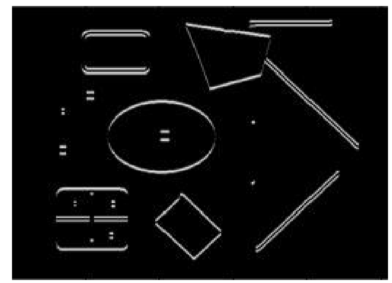
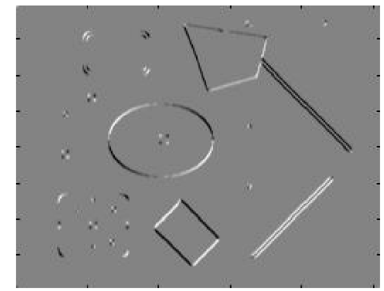
$$A(\sigma_I, \sigma_D) = G(0, \sigma_I) \star \begin{bmatrix} I_x^2(\sigma_D) & I_x(\sigma_D) I_y(\sigma_D) \\ I_x(\sigma_D) I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

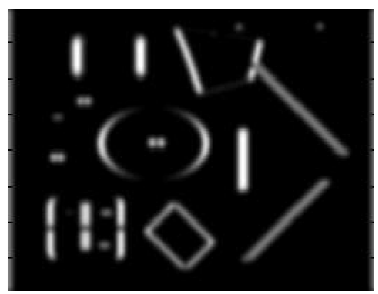
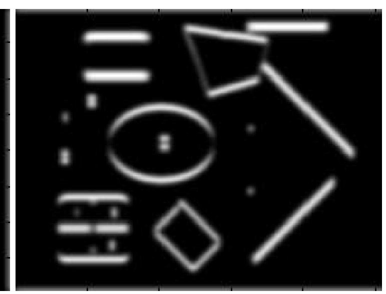
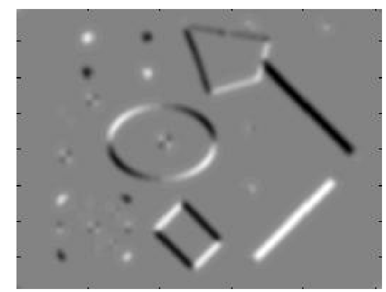
Different Stages

Image: 

Gradients: 



Squares/Products:   

Weighted Squares:   

Corners: 

Corners on an Image

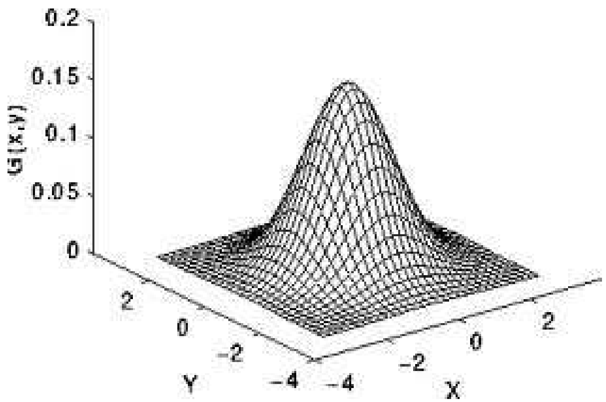
Other Corner Detectors

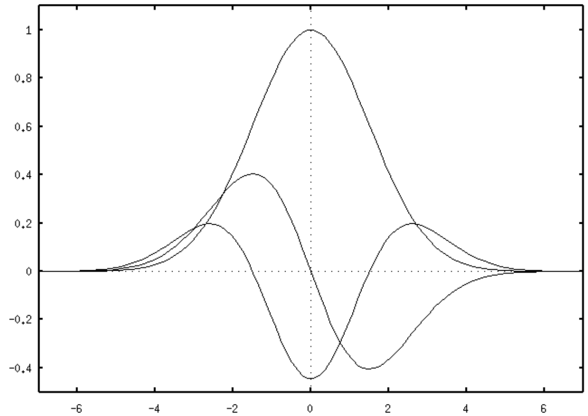
- Hessian
- Laplacian, Difference of Gaussian (DoG)
- Hessian-Affine, Harris-Affine
- Hessian-Laplace, Harris-Laplace
- Maximally Stable Extremal Regions (MSER)
- And others ...

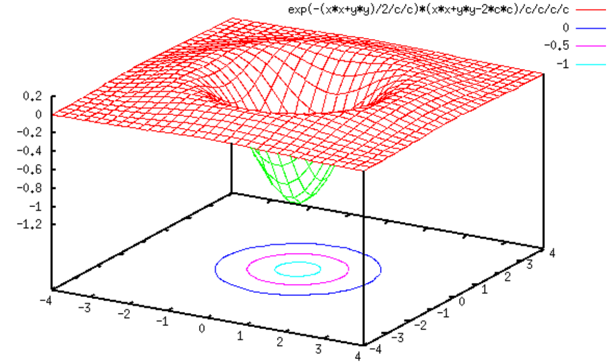
Corners at Different Scales

- Corners and surrounding regions are not always of same size/scale
- Need to look for corners in different scales
- Ideally, only one “natural” scale should succeed
- Laplacian of Gaussian (LoG): a blob detector
- Characteristic scale: Where response to LoG is maximum in scale space

LoG and Blob Detection





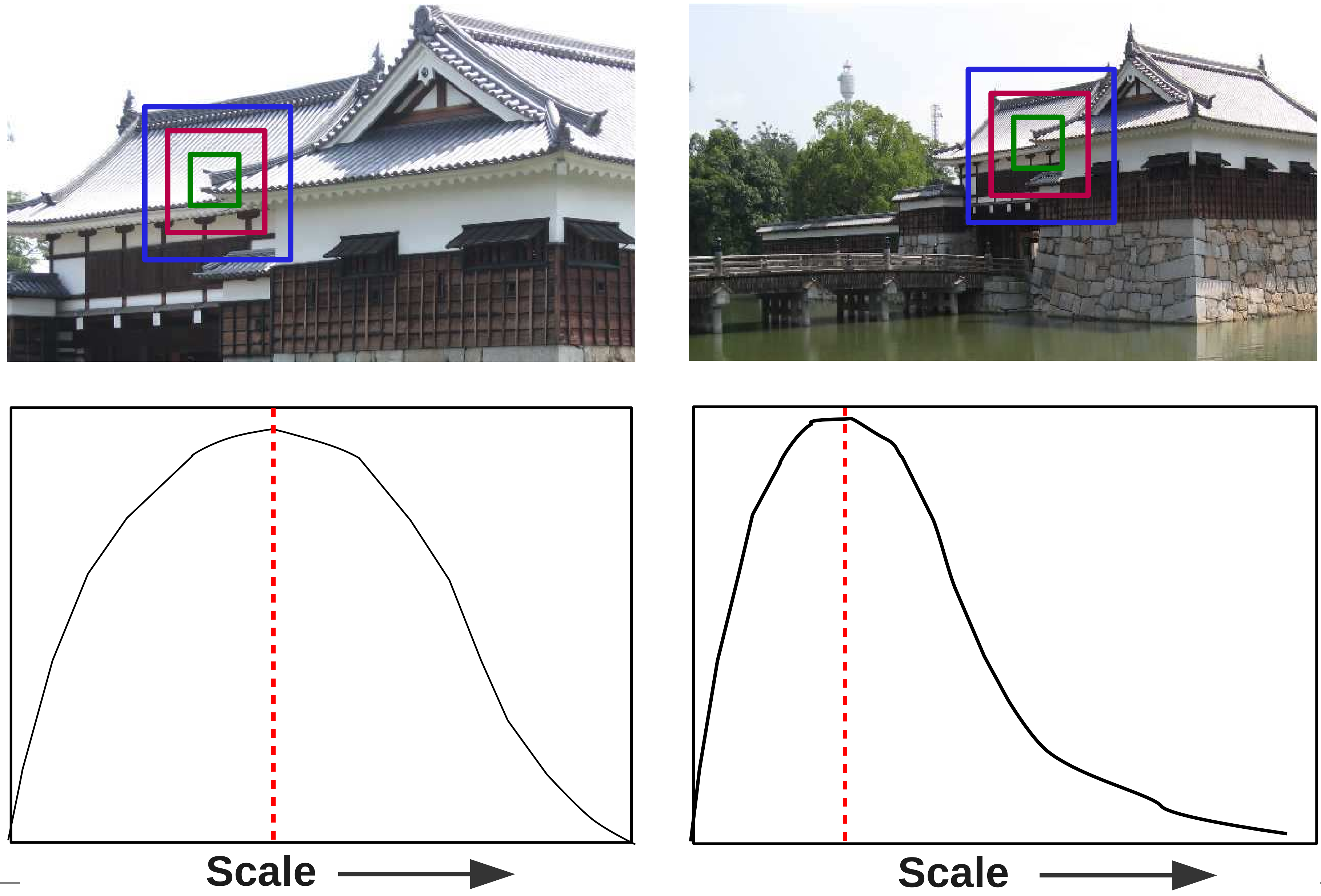


- Laplacian: $\nabla^2 f = f_{xx} + f_{yy}$
- Laplacian of Gaussian: $LoG(x, y, \sigma) = \nabla^2 G(x, y, \sigma)$

$$\nabla^2 G(x, y, \sigma) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2}$$

- LoG is a filter that can be applied everywhere
- LoG responds to dark/bright **blobs** matching its scale.
- Change scale by changing the σ of the Gaussian

LoG at Different Scales



Corners & Characteristic Scale

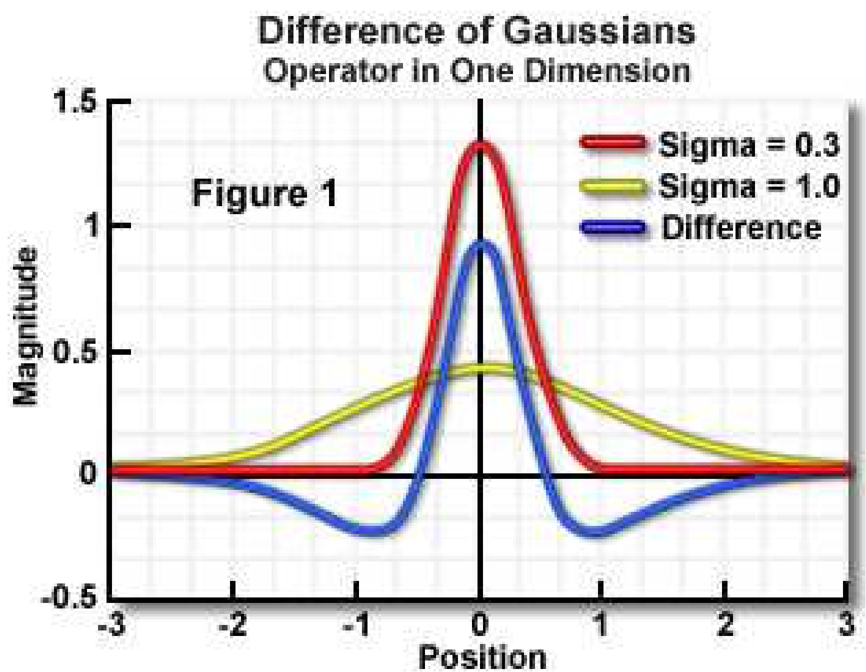
- Scale signature: Apply LoG at different scales at a point and plot the response against the scale.
- The response peaks at the **characteristic scale** of the feature
- The blob feature makes most sense at the characteristic scale.
- Corner detection (like Harris corners) should also recover the scale of the feature.
- Find the local maximum in an x-y-σ space.

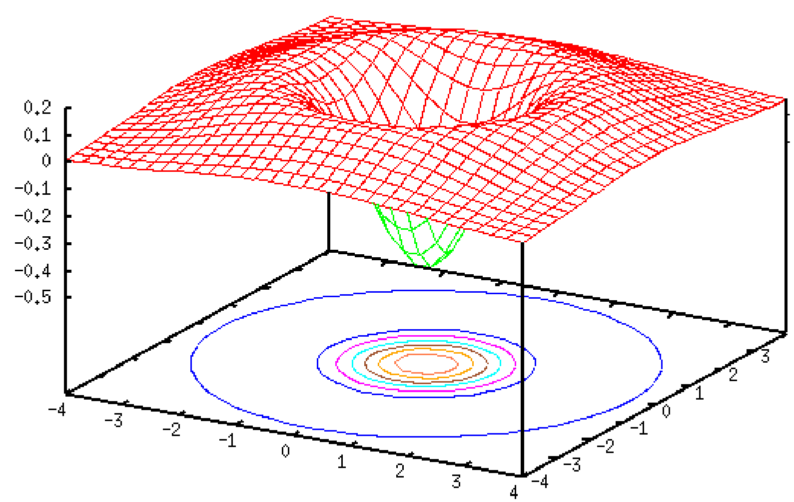
SIFT [Lowe 1999, 2001]

- **Scale Invariant Feature Transform**: An influential detector and descriptor by David Lowe.
- Detects the scale, orientation of the feature and gives a gradient-based descriptor for it.
- Translation, rotation, and scale invariant interest points.
- Useful for image matching, recognition, robotics, etc.
- Four steps:
  1. **Find the extrema in the scale space**
  2. **Detect the keypoint location**
  3. **Assign the dominant orientation**
  4. **Generate the gradient-based descriptor**

Difference of Gaussian (DoG)

- LoG gives the best results for the characteristic scale, but is expensive to compute.
- Lowe showed the *Difference of Gaussians (DoG)* does well. It approximates the scale-normalized LoG.
- Subtract Gaussians in different scales to get DoG

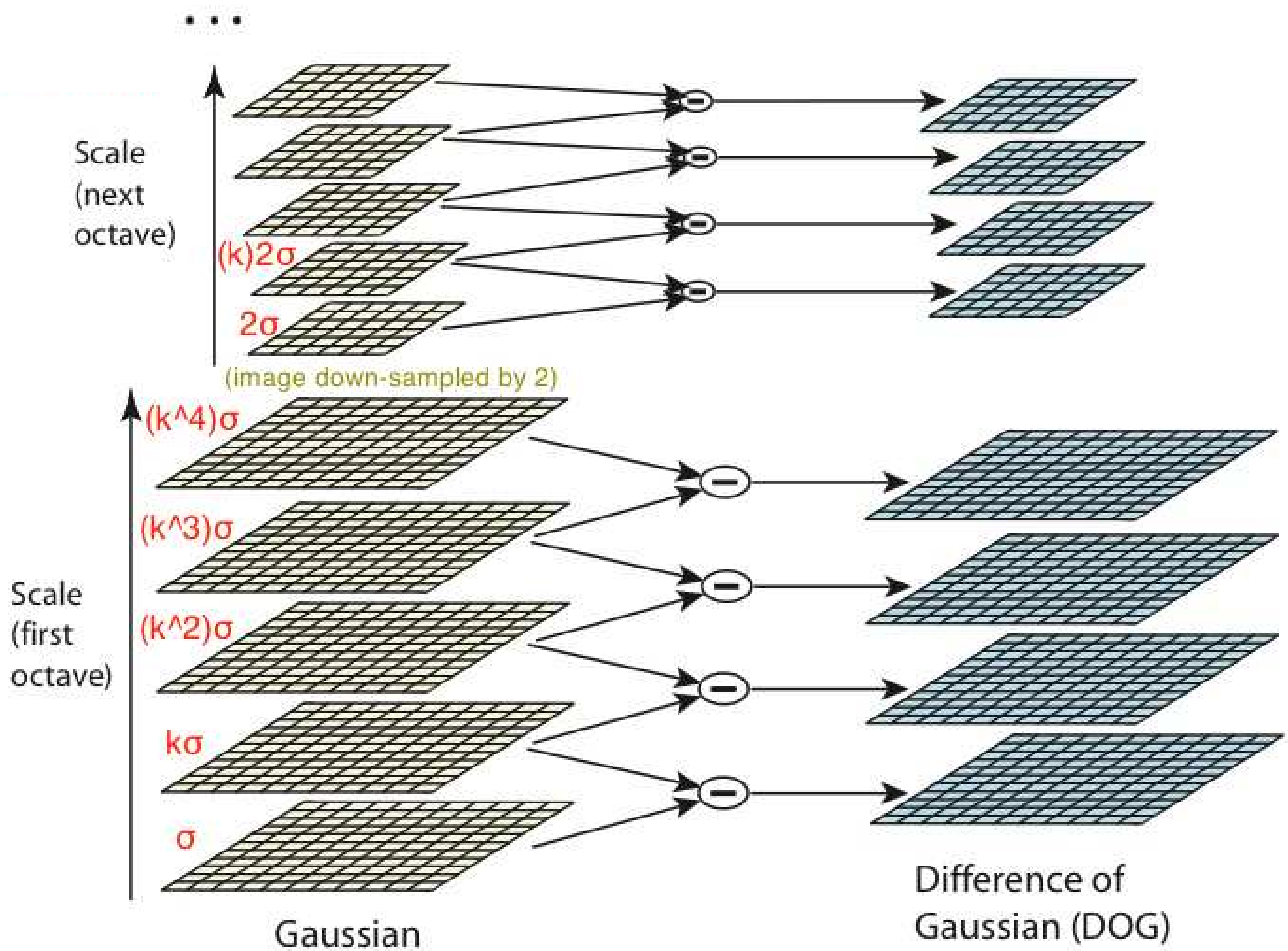




Gaussian and Laplacian Pyramid

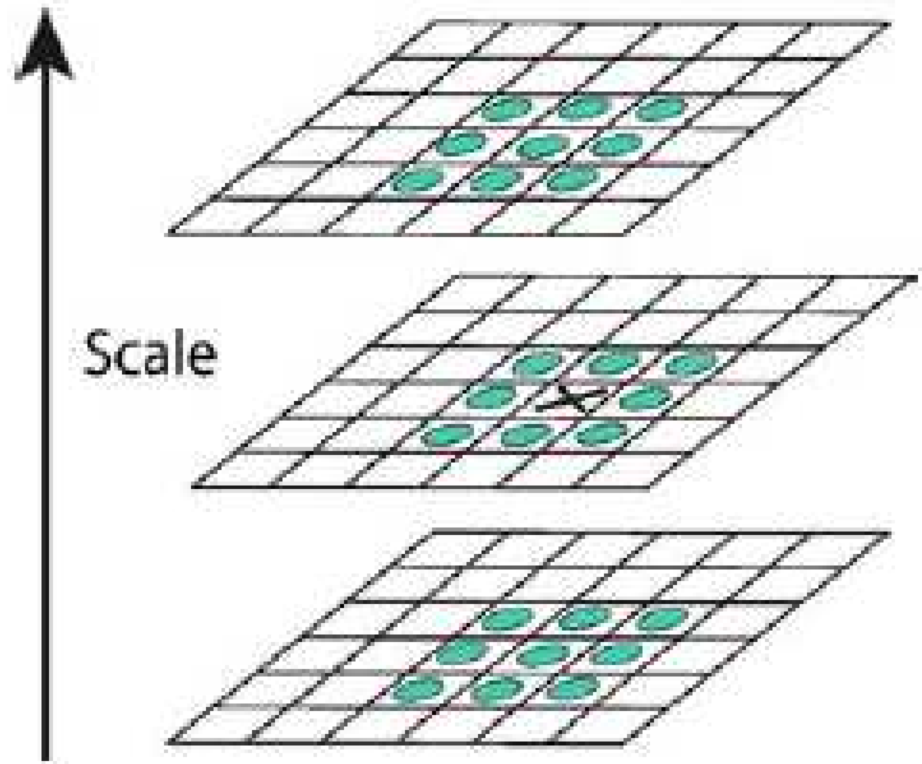
- DoG: Build a Gaussian Pyramid by convolving with Gaussians: $L(u,v) = G(x,y,k\sigma) \star I(u,v)$, for different values of k
- Subtract to get $D(u,v,k) = (G(x,y,k\sigma) - G(x,y,\sigma)) \star I(u,v)$
- Laplacian Pyramid: Lowest resolution Gaussian pyramid plus the differences at other resolutions.
- Typical value of $k = 1.2 - 1.3$. Make s levels in an octave where $s = \log_k 2$ or $k = 2^{1/s}$.
- For the next octave, downsample the image by a factor of 2 and repeat.
- David Lowe found 3 scales per octave ($k = 2^{\frac{1}{3}}$) and $\sigma = 1.6$ to work best based on experiments.

Scale Space for SIFT



Scale Space Extrema Detection

- Consider the 3D neighbourhood of each pixel with 8 pixels in the current scale and 9 each in scales above and below.
- Declare the pixel a candiate interest point if its resposne is larger or smaller than all neighbours.
- Remove unstable ones: Those of *low contrast* and those *near edges*



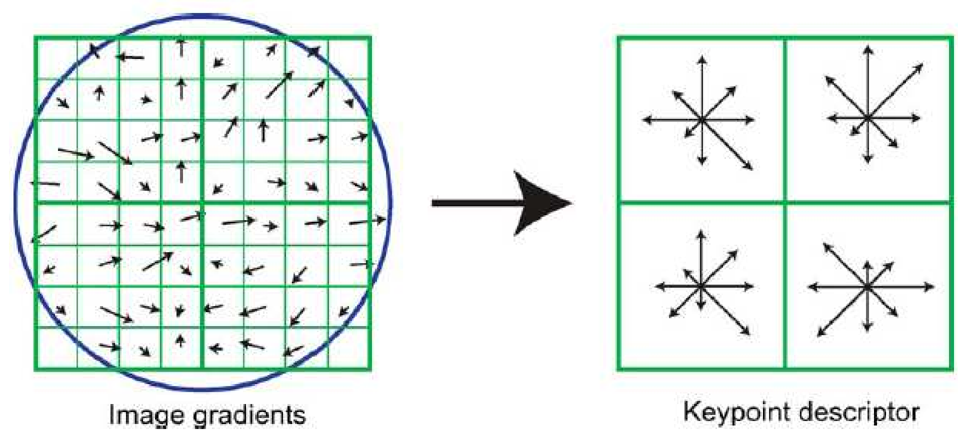
Orientation Assignment

- Given the selected (nearest) scale of each candidate key point, find the gradient magnitude and direction at the pixel in the Gaussian image of that scale.
- Form a 36-bin orientation histogram of pixels around the candidate point. Each pixel adds its Gaussian-weighted gradient magnitude to its orientation bin.
- Peak in the histogram represents the dominant orientation. This orientaiton is assigned to the keypoint.
- If another bin has a peak at 80% of the maximum, an additional keypoint with the new orientation is also generated.

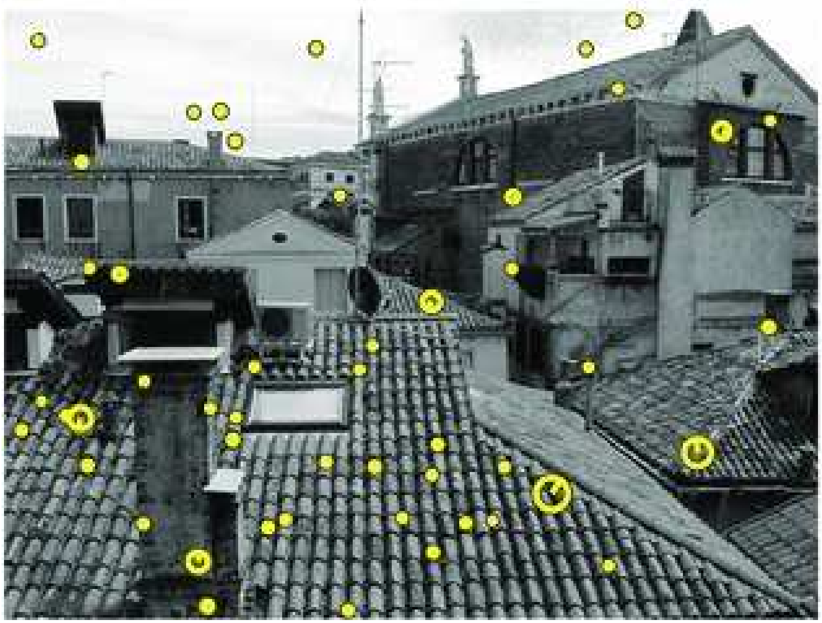
SIFT Descriptor

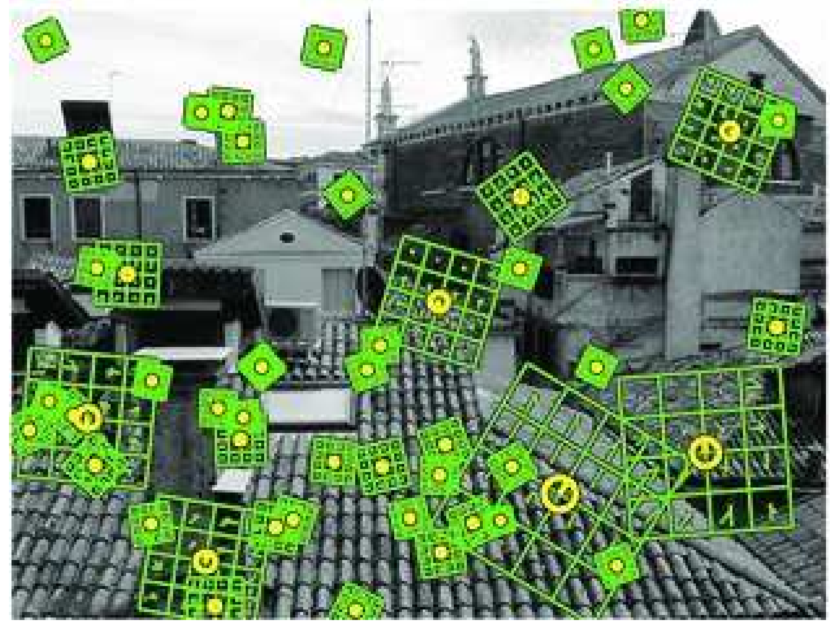
- Find gradient magnitude & direction at pixels around keypoint. Apply Gaussian weights to gradient magnitude.
- Rotate the directions with respect to the dominant direction to achieve rotation invarance.
- Find an orientation histogram of 8 bins and accumulate the weighted magnitudes in each.
- 16×16 regions, divided into 16 blocks of 4×4. Combine histograms into a vector. **Normalize**.

SIFT descriptor: A 128-dimensional ($4 \times 4 \times 8$) vector of gradient orientation histograms



An Example

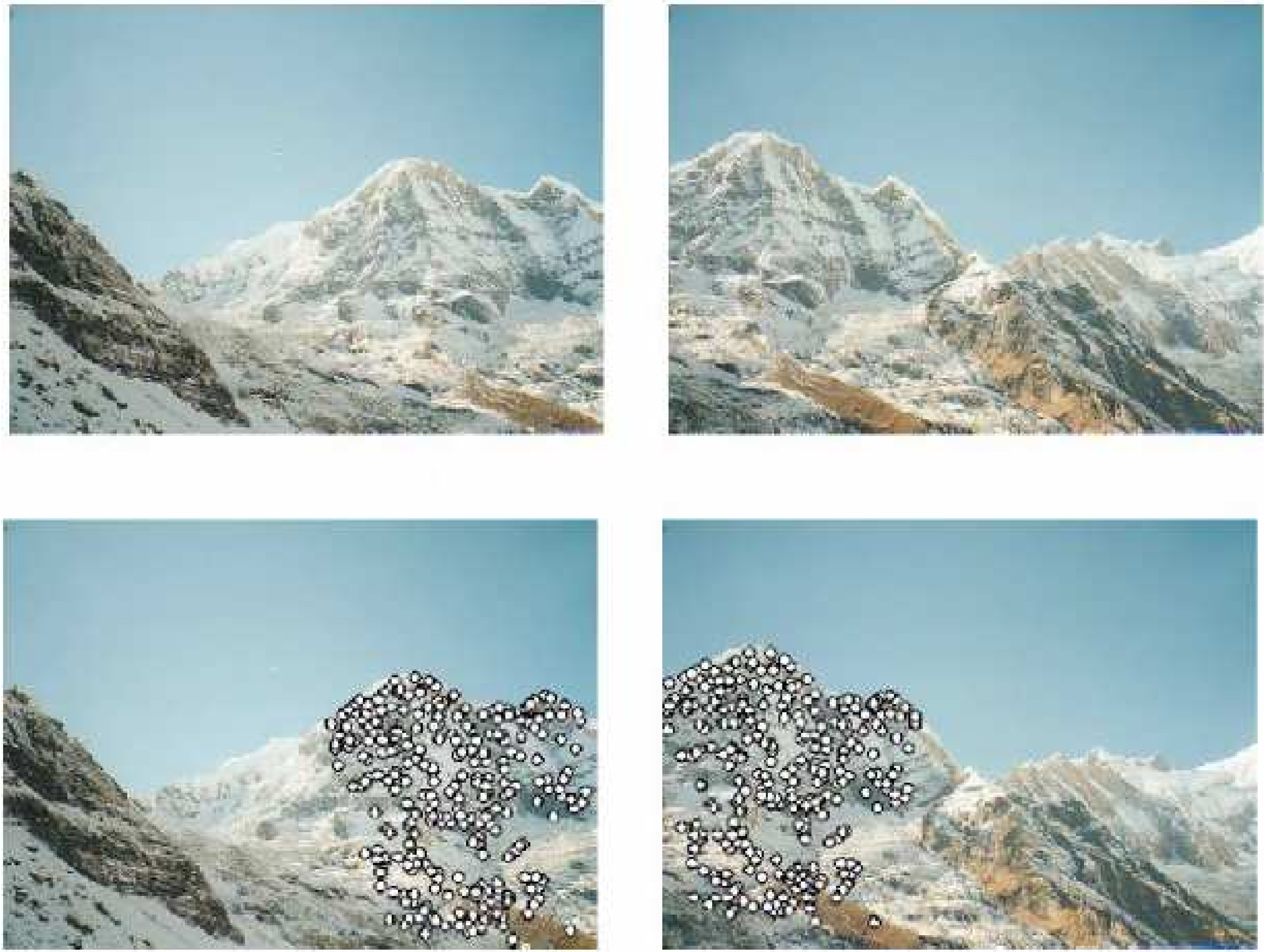




SIFT for Recognition

- Find SIFT keypoints and detectors in the image
- Can get sufficient matches for recognition even with a lot of occlusions.
- Use *ratio test* for matching a feature from one image to features in the other!

SIFT for Panorama



SIFT for Panorama

Matching: Find closest 2 neighbours in SIFT space.

If ratio of their distances is low (say < 0.7), correct match!!

RANSAC to compute homography

SIFT and Others

- SIFT is reasonably invariant to translation, rotation, scaling, intensity variations, and some affine transformations.
- Efficient implementations from David Lowe, VLFeat, etc.
- PCA-SIFT: Reduce dimensionality using PCA.
  It is not clear this benefits, however.
- SIFT is somewhat slow to compute, however.
- SIFT is also patented!!

Other Descriptors

- SURF: Speeded-Up Robust Features.
  - Fast computation and detection.
  - Uses integral images.
  - Patent free!!
- Shape Context: Describes shapes by counting points in bins in a log-polar space
- Geometric Blur: Some photometric information along with shape
- Binary features: Local Binary Pattern (LBP), BRISK, BRIEF, ...
- GIST: A global feature, good for scene classification
- And many others....

Local Binary Pattern (LBP)

10	7	9
12	12	10
14	**14**	**15**

0	0	0
1		0
1	1	1

5	10	5
3	7	12
9	5	11

0	1	0
0		1
1	0	1

- Compare the values of the middle pixel with its neighbours
- Encode greater as a “1” and lesser as a “0”
- An 8-bit binary number with 8 neighbours
- Can do at larger radius values also for different scales
- **The magnitude of the difference is no longer important. Only the sign is!**

Detector vs Descriptor

- David Lowe proposed SIFT Detector and Descriptor
- These are not necessarily related. Detector finds good points to focus on. Descriptor says what goes on around them
- Detecting a keypoint followed by describing it is the usual order
- However, of late, the descriptor is evaluated at regular intervatls in Computer Vision. Called **dense SIFT**
- Brings in sampling issues, etc., but is popular today

Thank You!

P J Narayanan

Thanks to the Computer Vision community worldwide
for many of the figures!!