CSCI435/CSCI935
Computer Vision: Algorithms & Systems

Keypoint Detection & Local Descriptors

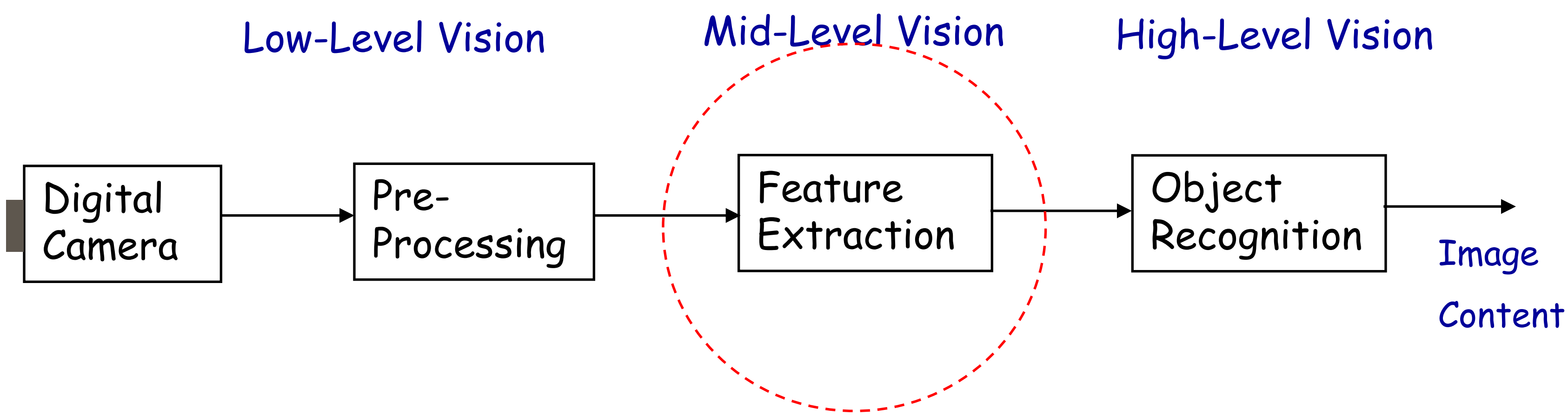
Lecturer: Prof Lei Wang
Room 3.219
Email: leiw@uow.edu.au
Web: https://scholars.uow.edu.au/lei-wang

Edge detection (review)

- Measurement
  - 1st and 2nd order derivation
- Detection
  - Convolution with edge operators (i.e. Sober's and Prewitt's)
  - Laplacian of a Gaussian (LoG) and Difference of Gaussians (DoG)
  - Edge linking/tracing and thinning
- Canny edge detection

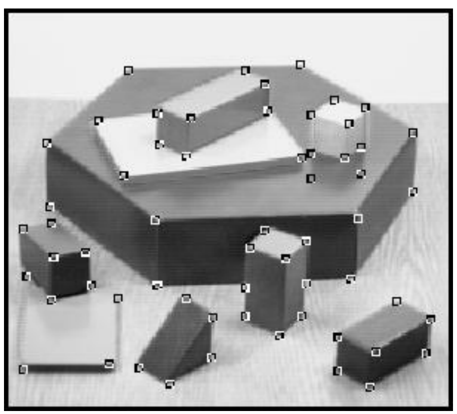
Machine Vision Concept (review)

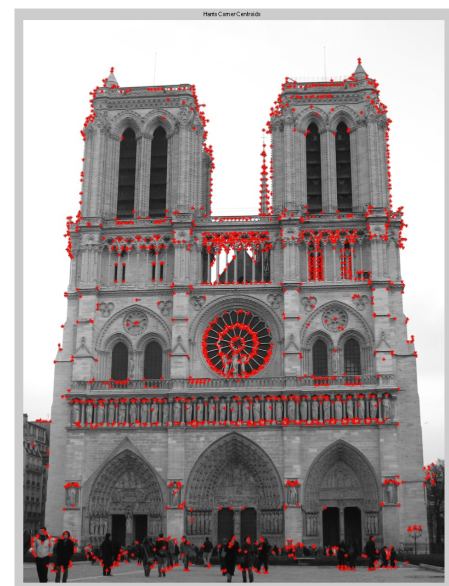
Machine Vision is a multistage process where each previous stage affects performance of all following stages



Intermediate-Level Vision

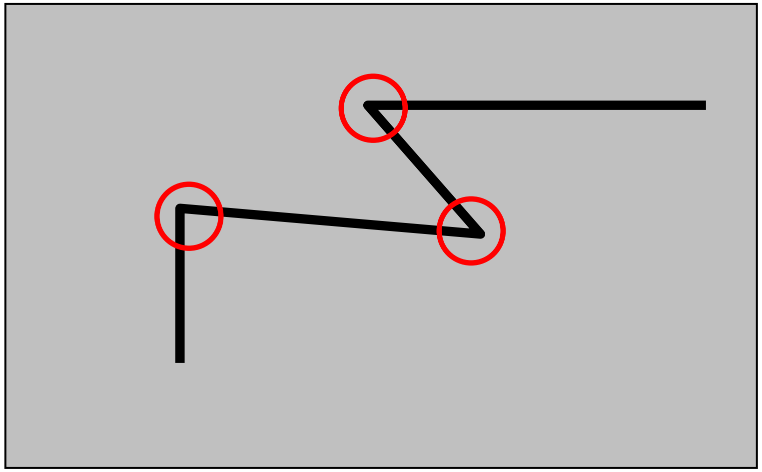
- In the past decades, it has been found that the corners and their spatial arrangement and local intensity/colour distribution around the corners carry much information about objects





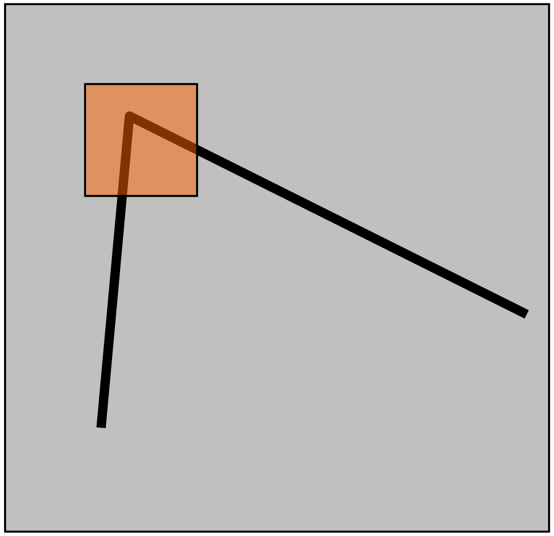
Harris Corner Detector

- An example

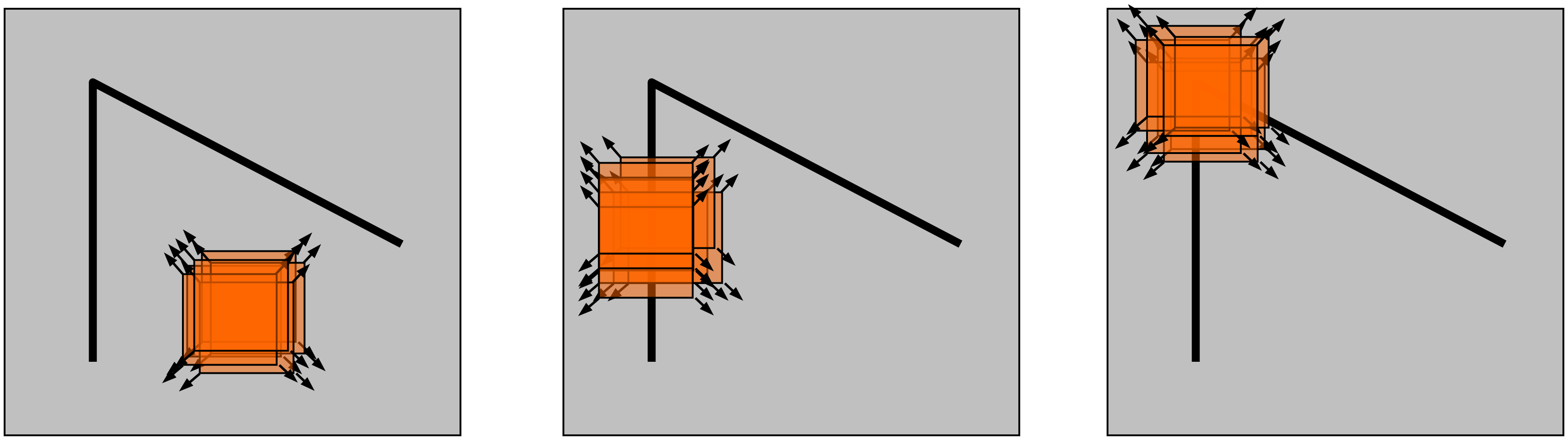


The Basic Idea

- We should easily localize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



Harris Detector: Basic Idea



"flat" region:
no change as shift window in all directions

"edge":
no change as shift window along the edge direction

"corner":
significant change as shift window in all directions

Harris Detector: Mathematics

Window-averaged change of intensity induced by shifting the image data by [*u,v*]:

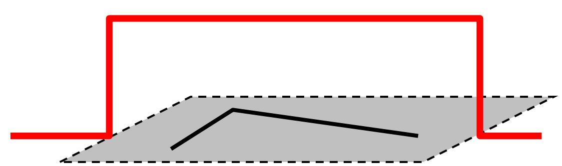
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

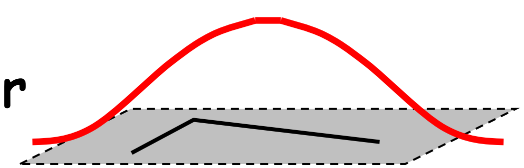
Intensity

Window function ***w(x,y)*** =



1 in window, 0 outside

or



Gaussian

Taylor series approx

$$E(u,v) \approx \sum_{x,y} w(x,y)[\underline{I(x,y) + uI_x + vI_y} - I(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= \sum_{x,y} w(x,y)\begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Harris Detector: Mathematics

Expanding I(x,y) in a Taylor series expansion, we have, for small shifts $[u,v]$, a *bilinear* approximation:

$$E(u,v) \cong [u,v] \ M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

M is also called "structure tensor"

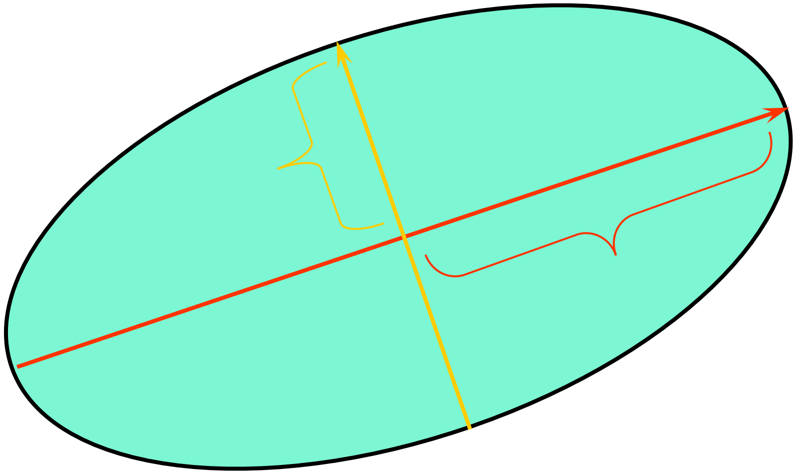
Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

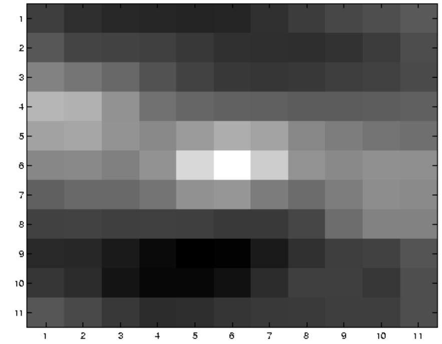
λ_1, λ_2 – eigenvalues of $\boldsymbol{M}$

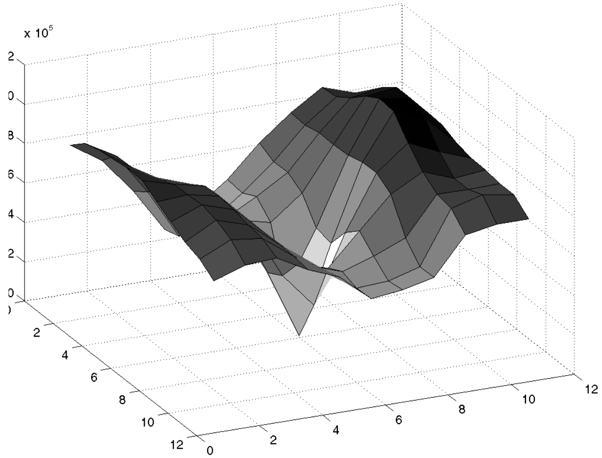
Ellipse $E(u,v)$ = const



Iso-intensity contour of $E(u,v)$

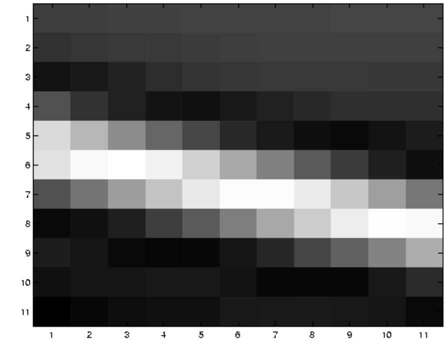
Selecting Good Features

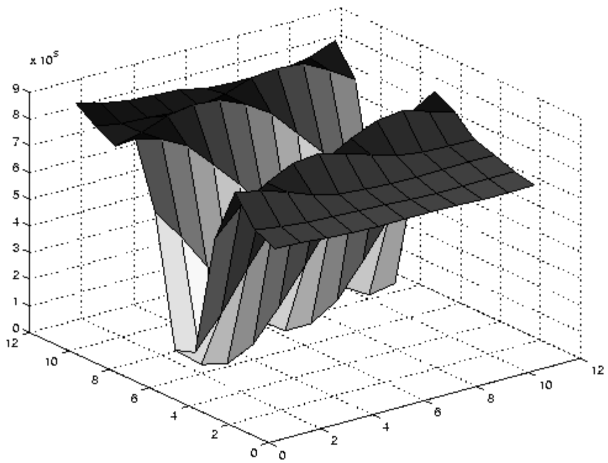




λ_1 and λ_2 are large

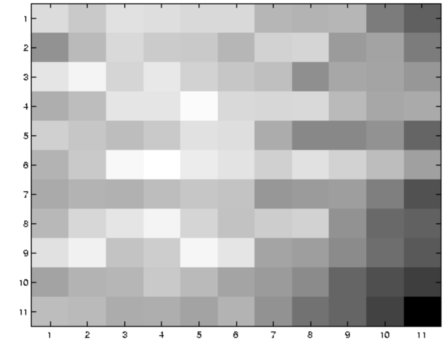
Selecting Good Features

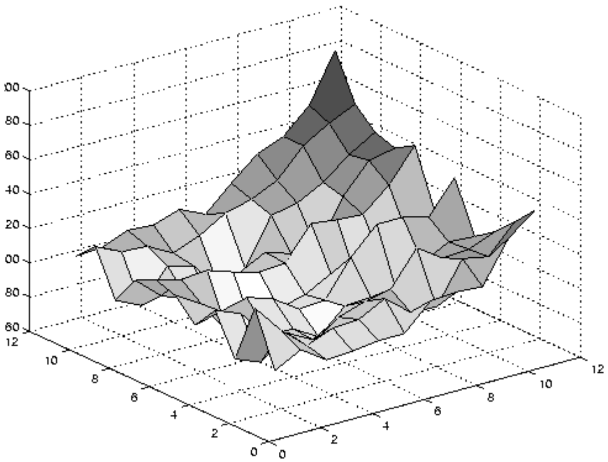




large λ_1, small λ_2

Selecting Good Features

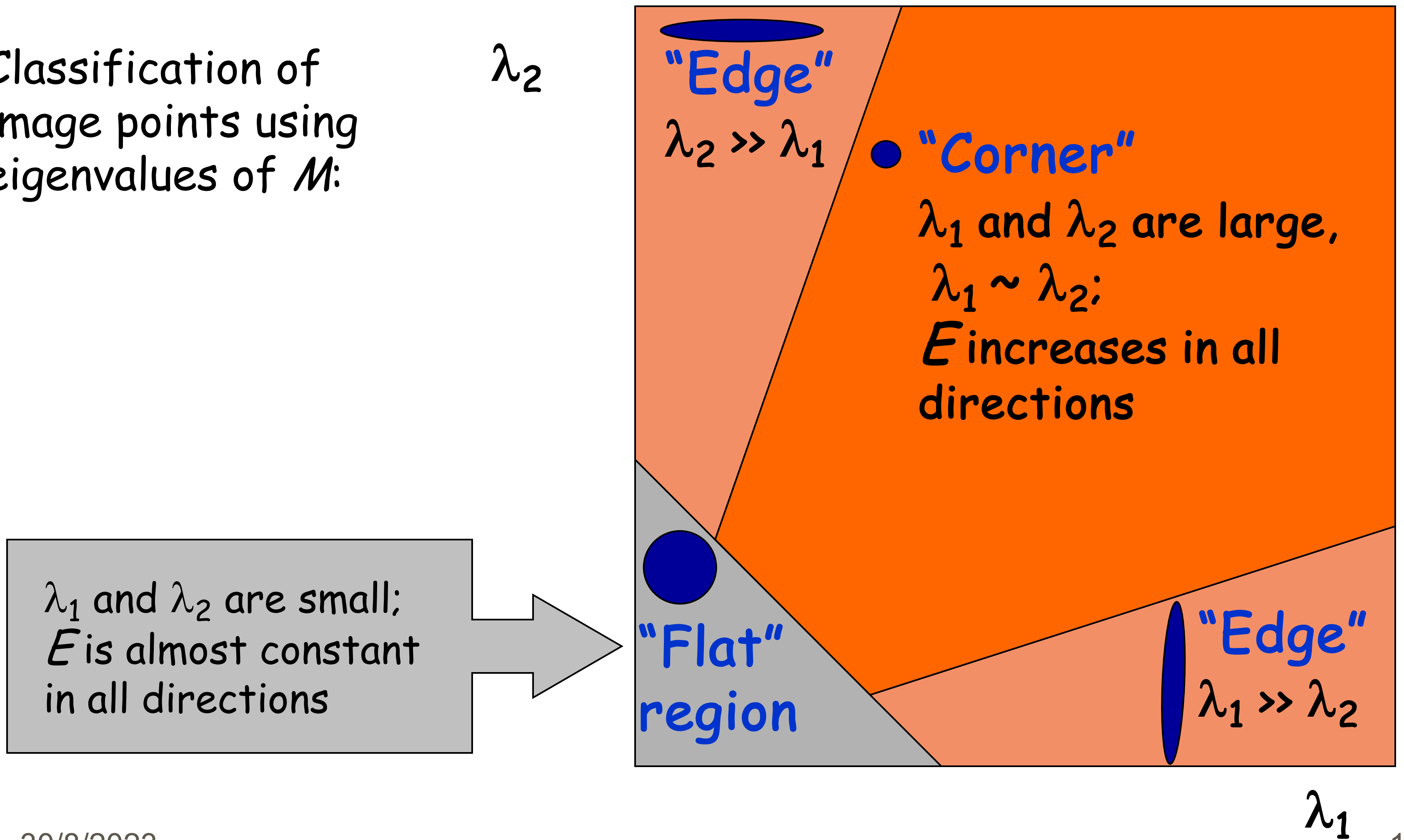




small λ_1, small λ_2

Harris Detector: Mathematics

Classification of image points using eigenvalues of M:



Harris Detector: Mathematics

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

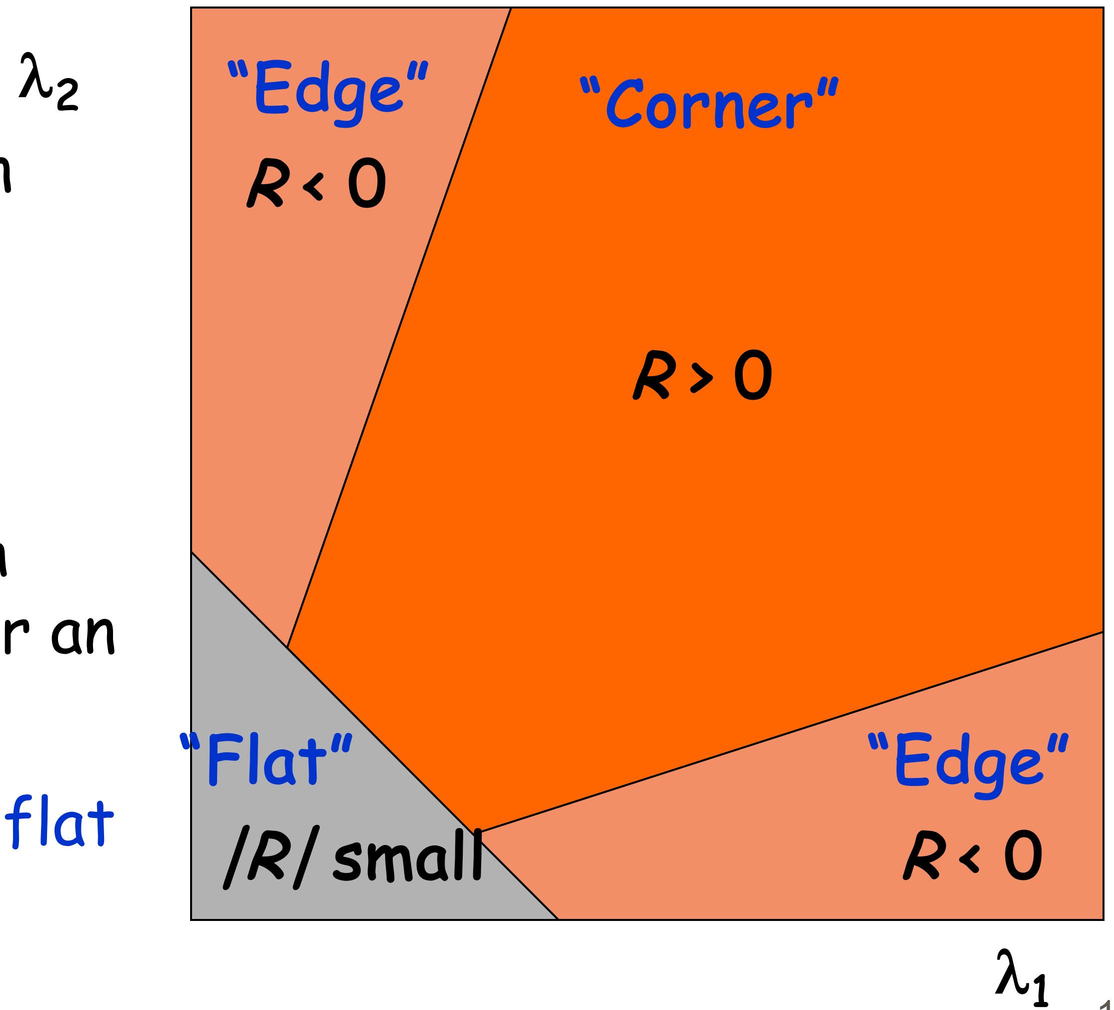
$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

This expression does not requires computing the eigenvalues.

(*k* – empirical constant, *k* = 0.04-0.06)

Harris Detector: Mathematics

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



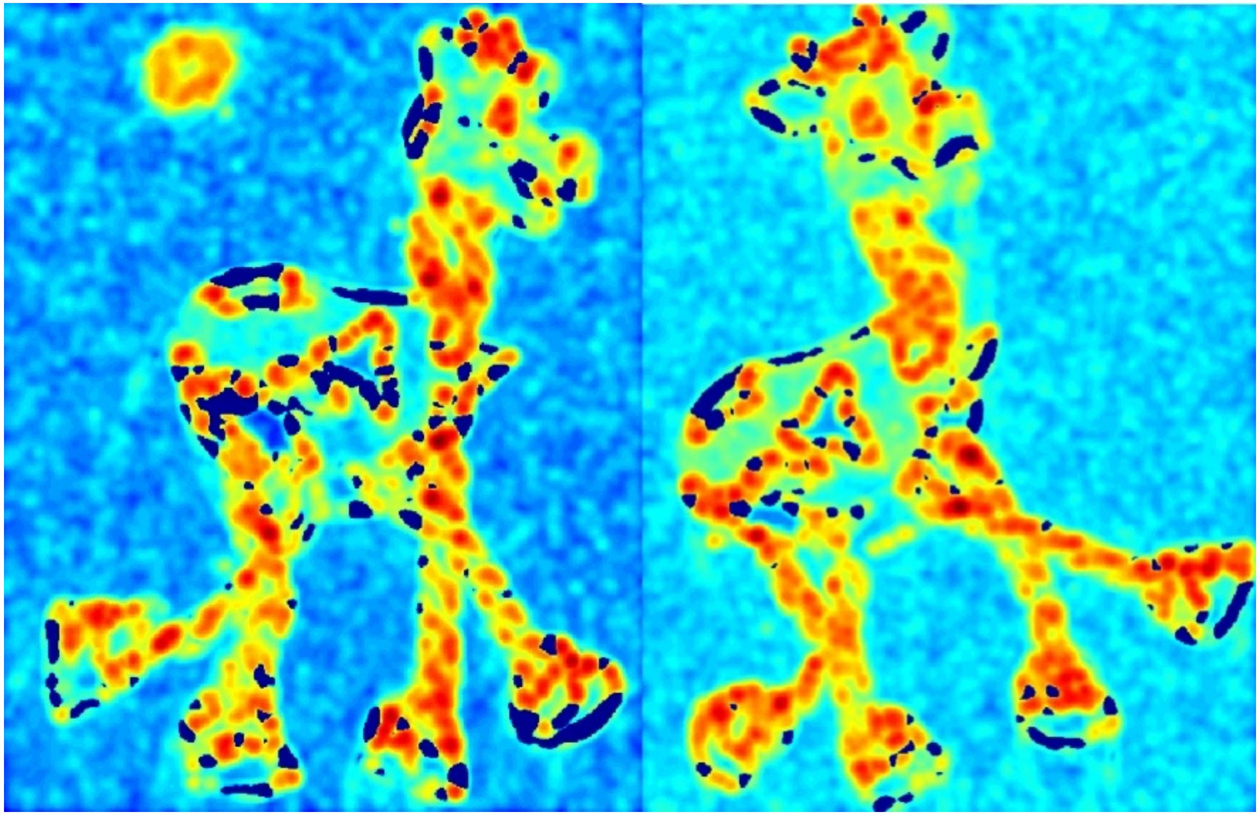
Harris Detector

- The Algorithm:
  - Find points with large corner response function R ($R >$ threshold)
  - Take the points of local maxima of R

Harris Detector: Workflow

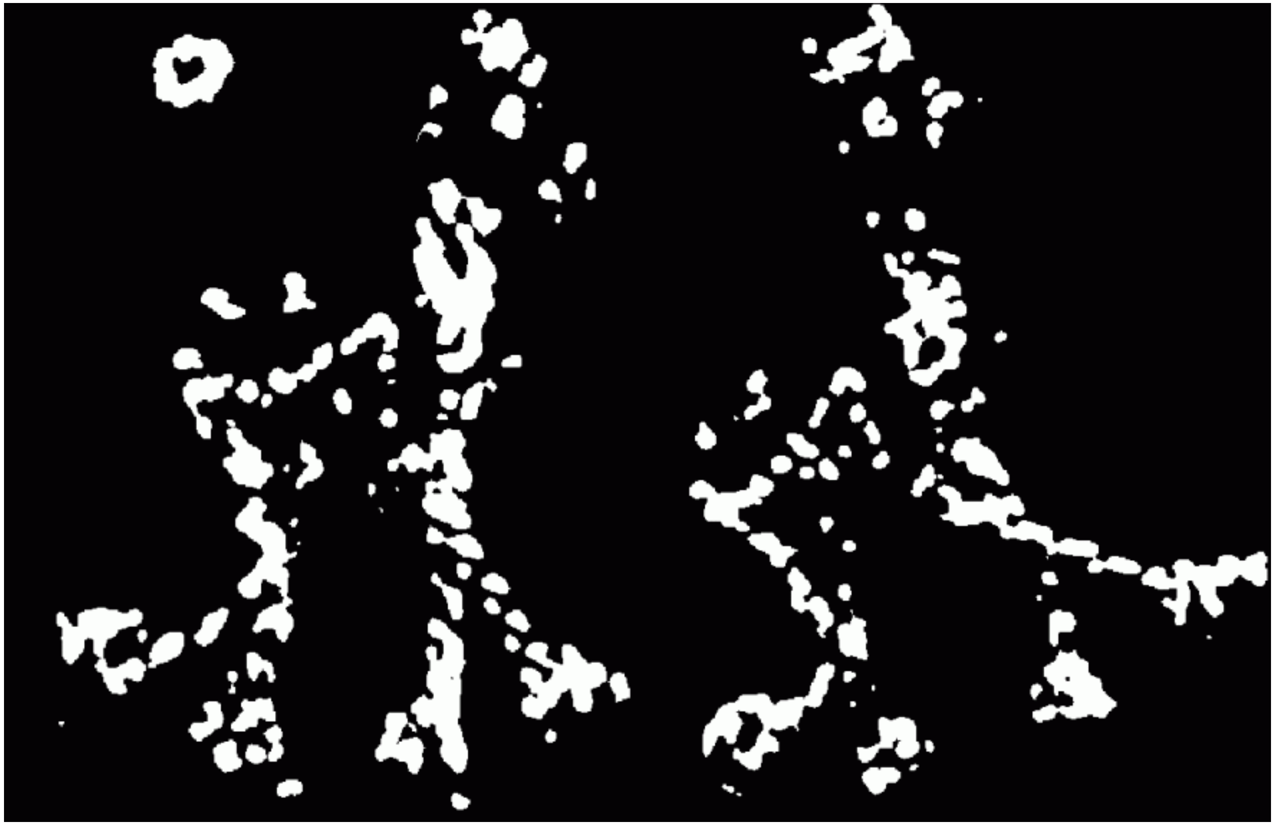
Harris Detector: Workflow

Compute corner response R



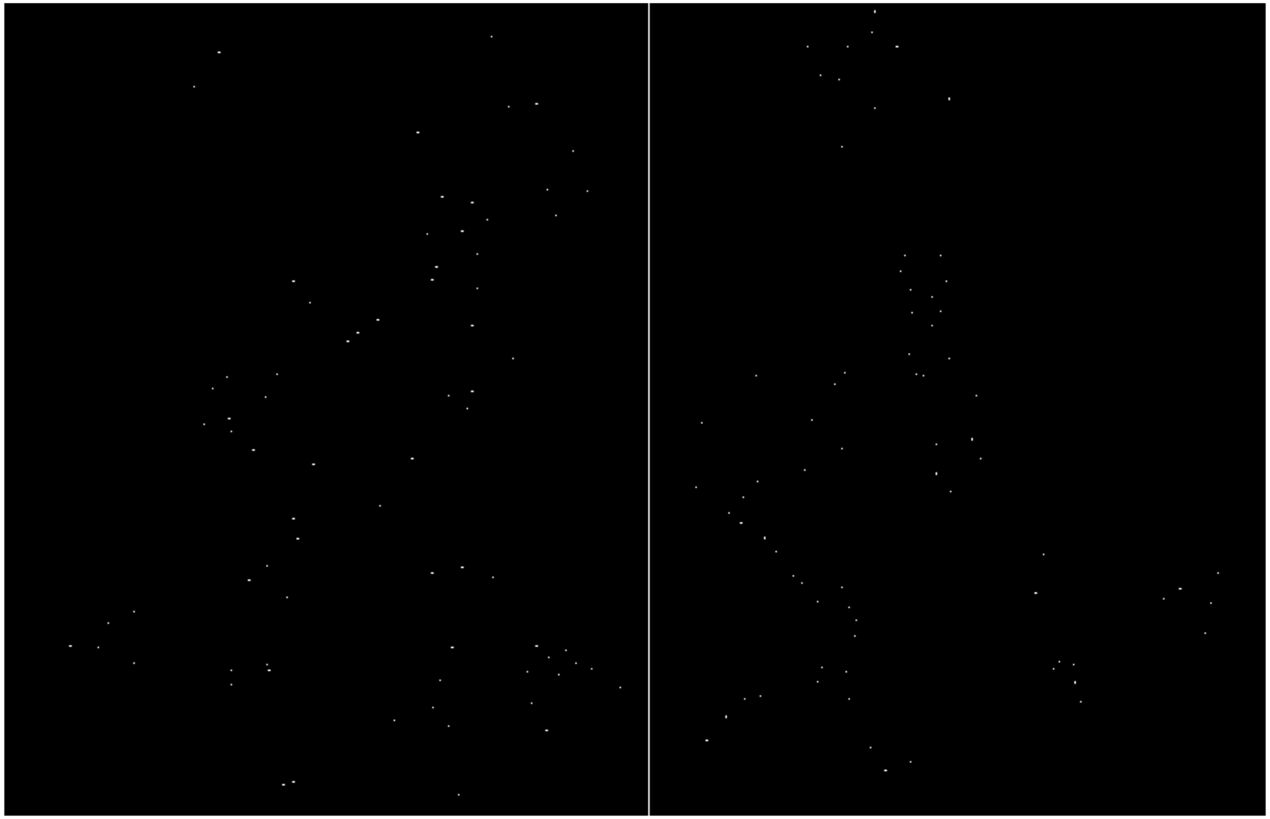
Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

Take only the points of local maxima of R



Harris Detector: Workflow

Harris Detector: Summary

- Average intensity change in direction $[u,v]$ can be expressed as a bilinear form:

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

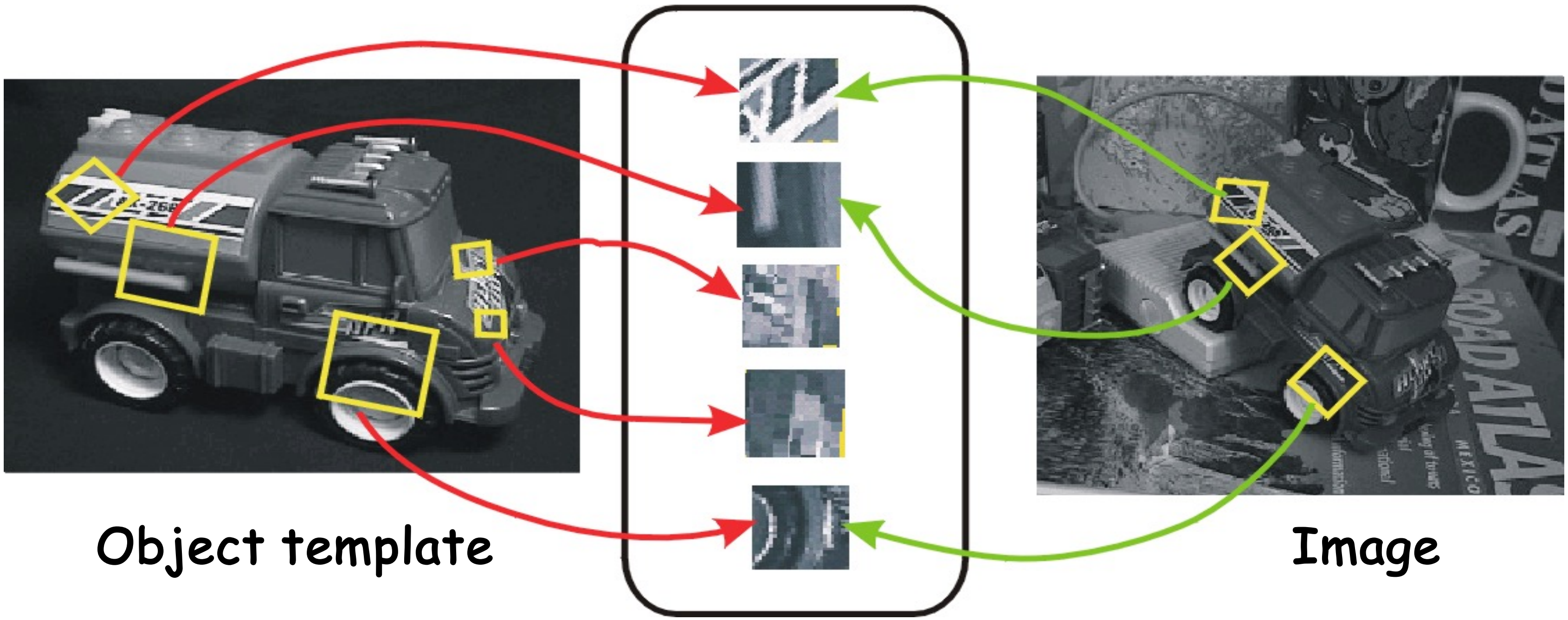
- Describe a point in terms of eigenvalues of M: *measure of corner response*

$$R = \lambda_1 \lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2$$

- A good (corner) point should have a *large intensity change* in *all directions*, i.e. R should be large positive

Use of Corners

- Image content is transformed into local feature extracted at the detected corners that are invariant to translation, rotation, scale, and illumination changes



Scale-Invariant Feature Transform (SIFT)

- SIFT in a brief is
  - Histogram of gradients @ Harris-corner-like keypoints
- Being invariant to
  - Scale
  - Rotation
  - Illumination changes
  - Small degree of viewpoints
  - Noise

SIFT Algorithm Overview

Filtered approach

- Scale-space extrema detection
  - Identify potential points: invariant to scale & orientation.
  - Difference-of-Gaussian function (DoG)
- Keypoint localization
  - Improve the estimate for location by fitting a quadratic
  - Extrema threshold for filtering out insignificant or edge points.
- Orientation Assignment
  - Orientation assigned to each keypoint and neighboring pixels based on local gradient.
- Keypoint Descriptor construction
  - Feature vector based on gradients of local neighborhood

Keypoint candidates: Scale Space

- We express the image at different scales by filtering it with a Gaussian kernel

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

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

Keypoint Candidates: why DoG's?

- Lindeberg(1994) and Mikolajczyk (2002) found that the maxima and minima of the scaled Laplacian $\sigma^2\nabla^2 G$ provides the most stable scale invariant features
- We can use the scaled images to approximate this:

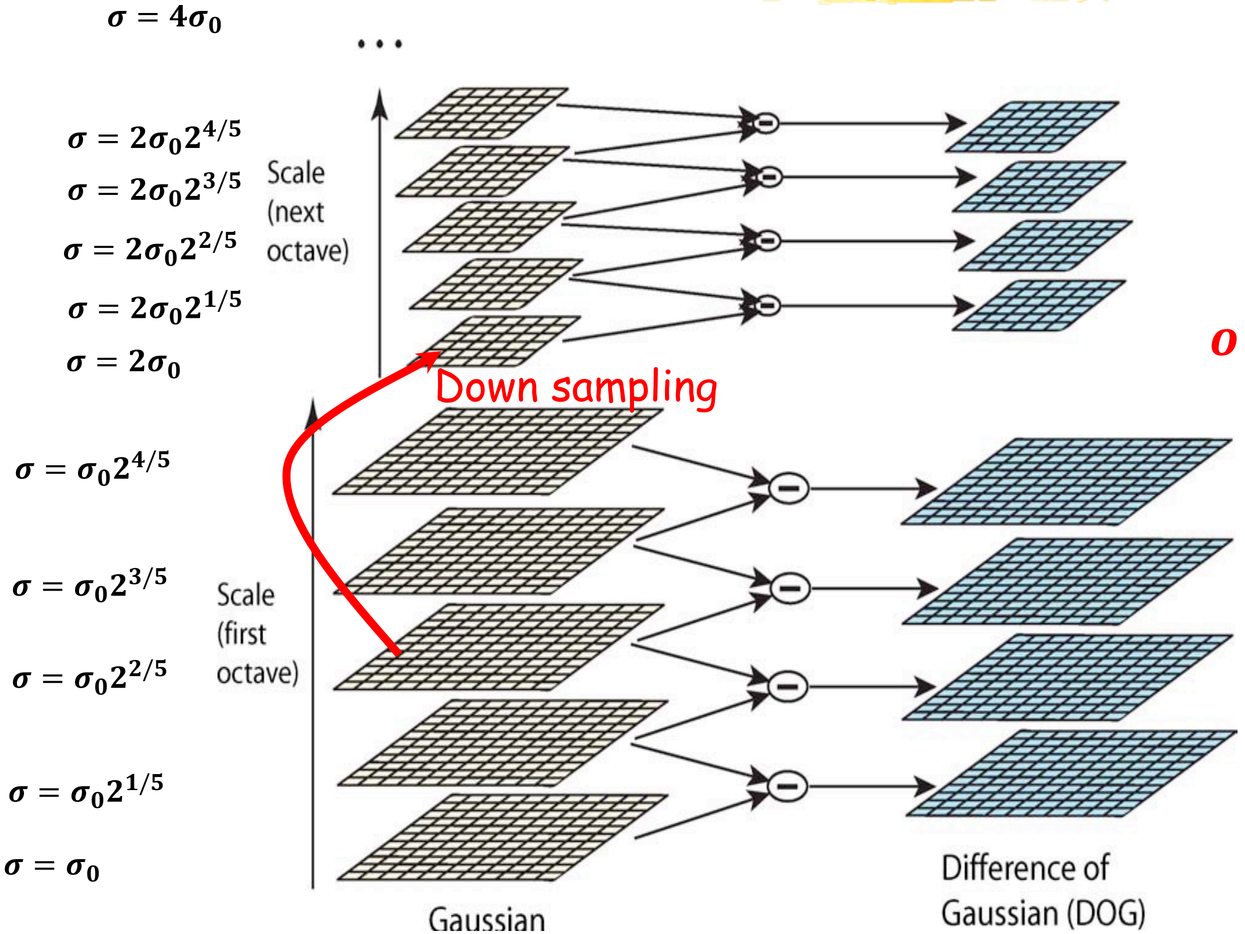
$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G.$$

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \qquad (1) \end{aligned}$$

- Efficient to compute
  - Smoothed images *L* needed later so *D* can be computed by simple image subtraction

Scale-Space Extrema Detection



$$\sigma(o, s) = \sigma_0 2^{o+s/S}$$

$$o \in o_{min} + [0, \cdots, O - 1]$$

$$o_{min} = 0 \text{ or } -1$$

$$s \in [0, \cdots, S - 1]$$

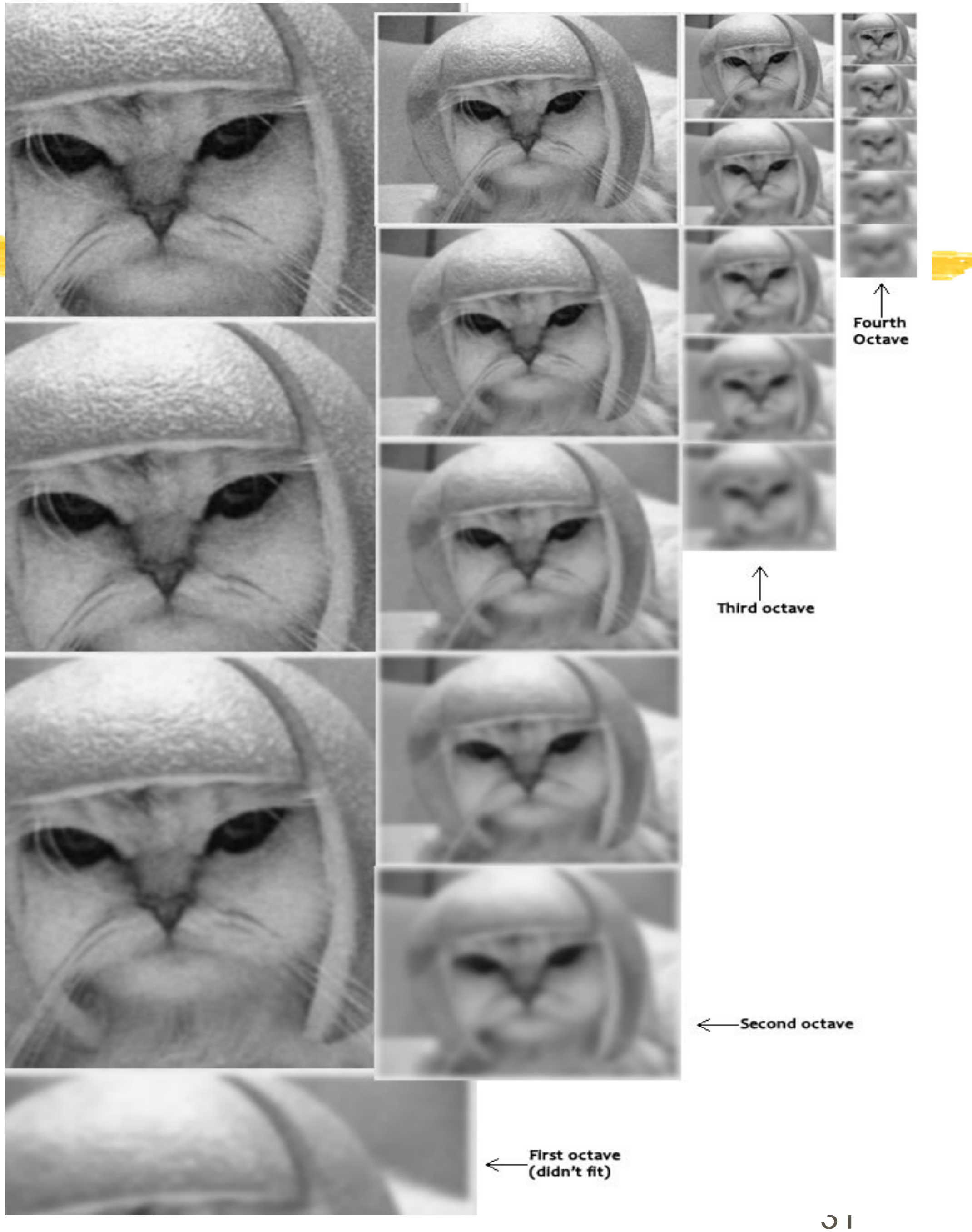
Typical Settings:

$\sigma_0 = 1.6$, $o_{min} = 0$

$O = 4$,$S = 5$

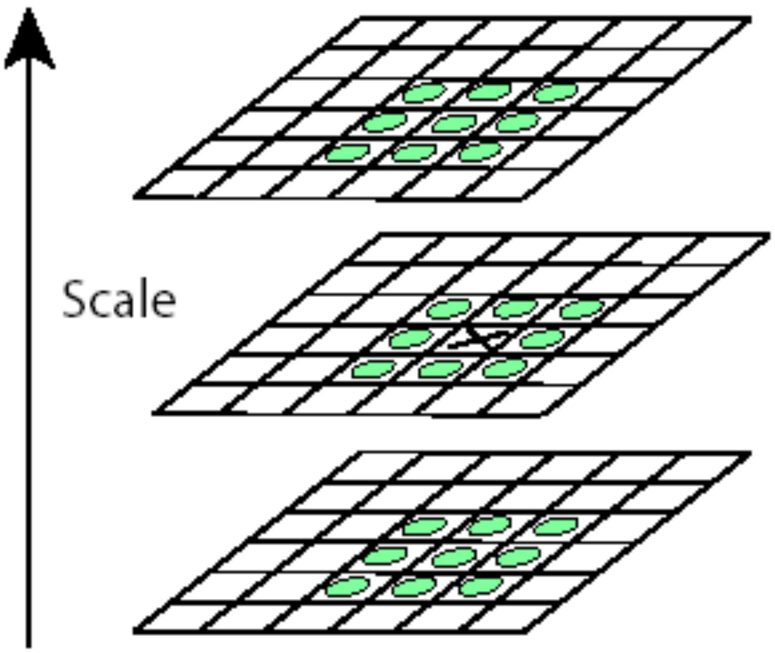
An example of scale-spaces in SIFT

- Images of the same size (vertical) form an octave. Above are four octaves. Each octave has 5 images. The individual images are formed because of the increasing "scale" (the amount of blur).



Scale-Space Extrema Detection…

- Maxima and minima in a 3*3*3 neighborhood are detected as the candidates of keypoints



DoG within an octave

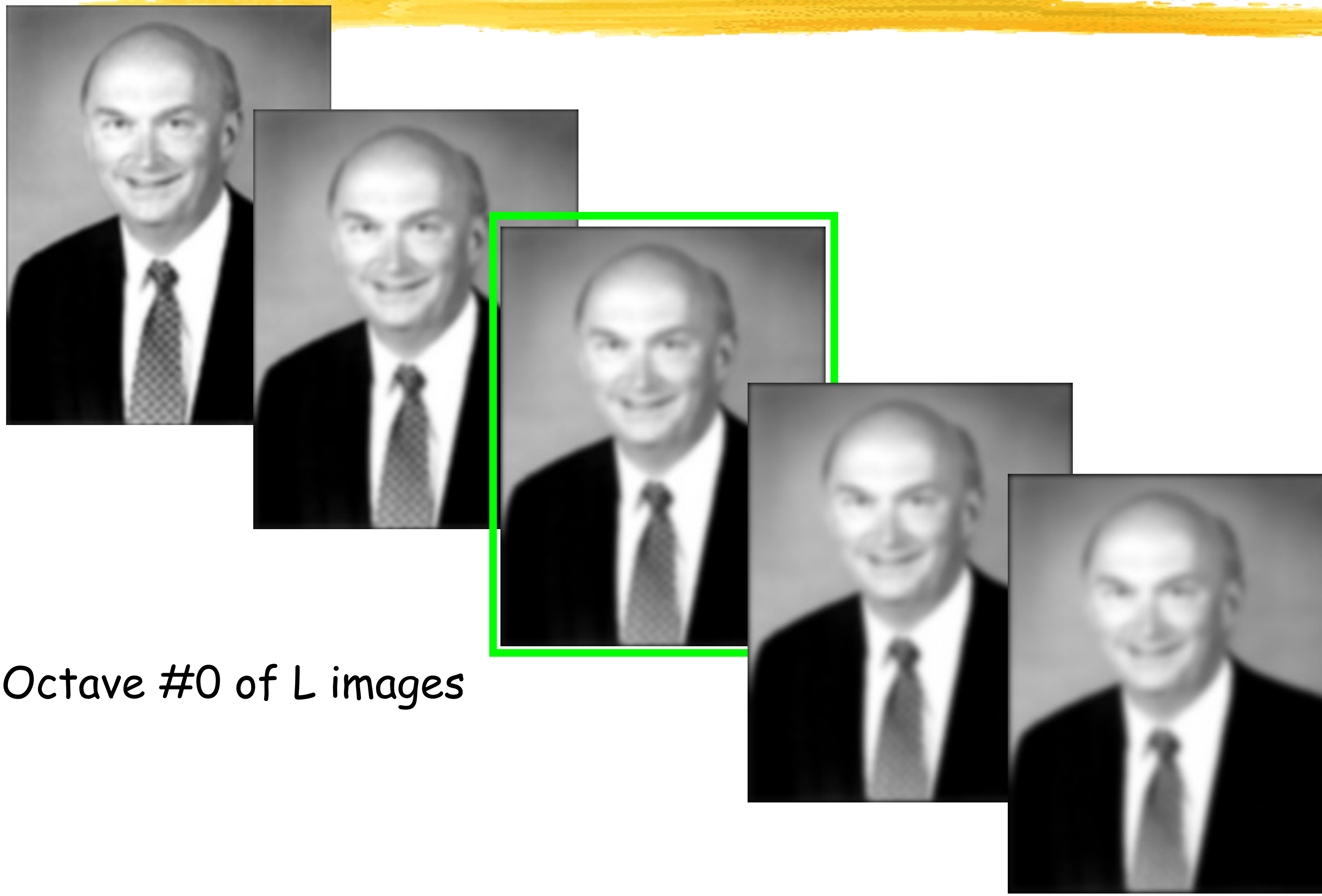
How Many Octaves ?

Original Image

Up-sampling

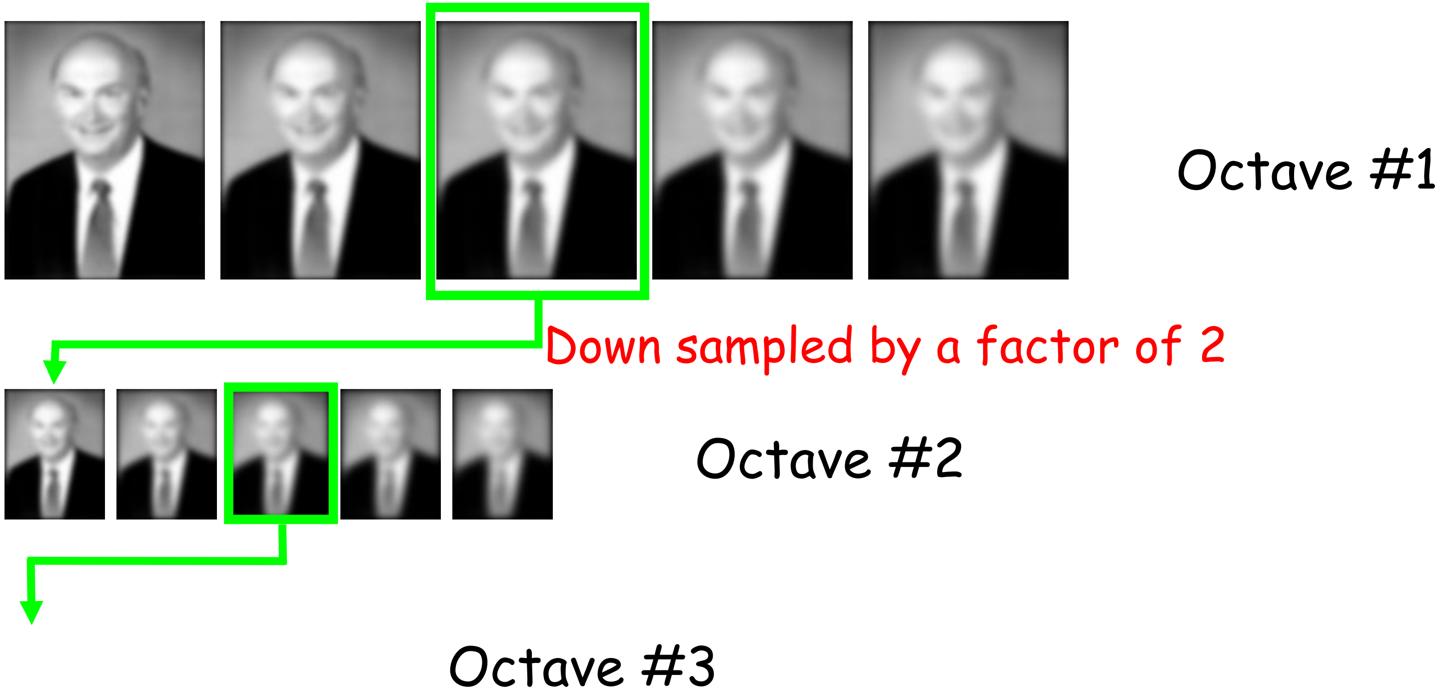
Starting Image

How Many Octaves ?...



Octave #0 of L images

How Many Octaves ?...



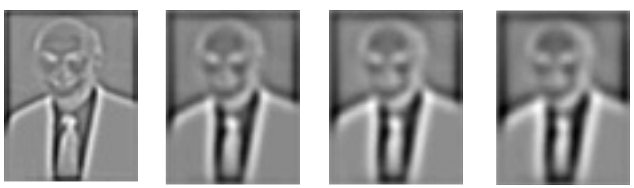
Until the image is too small

How Many Octaves ?...

Octave #0 DoGs

How Many Octaves ?...

DoGs of Octave #1

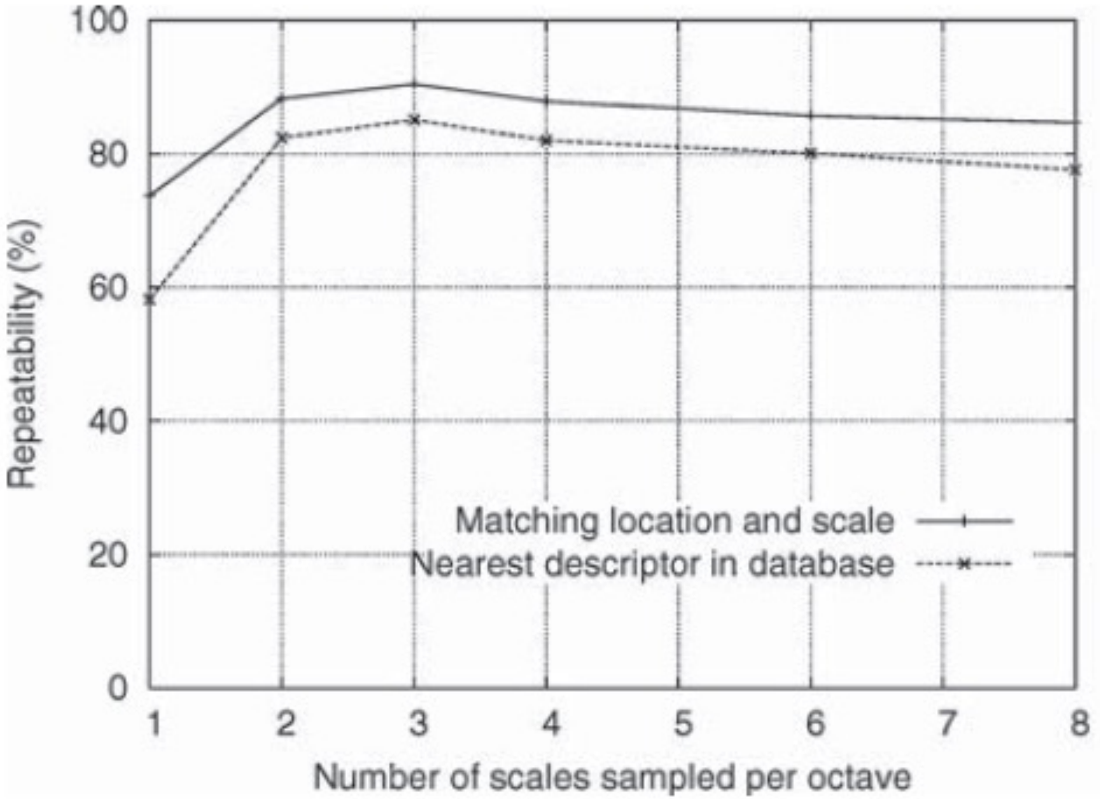


DoGs of Octave #2

DoGs of Octave #3

How Many Scales ? - Scale-space sampling

- How many fine scales in every octave?



Accurate Keypoint Localization

- From difference-of-Gaussian local extrema detection we obtain approximate locations for keypoints
- Originally these approximations were used directly
- For an improvement in matching and stability fitting to a 3D quadratic function is used

The Taylor Series Expansion

- Take Taylor Series Expansion of scale-space function D(x,y,σ)
  - Use up to quadratic terms

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

  - origin shifted to sample point
  - $x = (x, y, \sigma)^T$ offset from this sample point
  - to find location of extremun, take derivative and set to 0

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

Thresholding Keypoints

- The function value at the extrema is used to reject unstable extrema
  - Low contrast
  - Evaluate

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

  - Absolute value less than 0.03 at extrema location results in discarding of extrema (assuming image pixel values are in the range of [0,1])

Eliminating Edge Responses

- Difference-of-Gaussian function will be strong along edges
  - Some locations along edges are poorly determined and will become unstable when even small amounts of noise are added
  - These locations will have a large principal curvature across the edge but a small principal of curvature perpendicular to the edge
  - Therefore we need to compute the principal curvatures at the location and compare the two

Computing the Principal Curvatures

- Hessian matrix (The derivatives are estimated by taking differences of neighboring sample points)

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

- The eigenvalues of H are proportional to principal curvatures

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta \qquad \alpha = r\beta$$

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

- We are not concerned about actual values of eigenvalues, just the ratio of the two

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

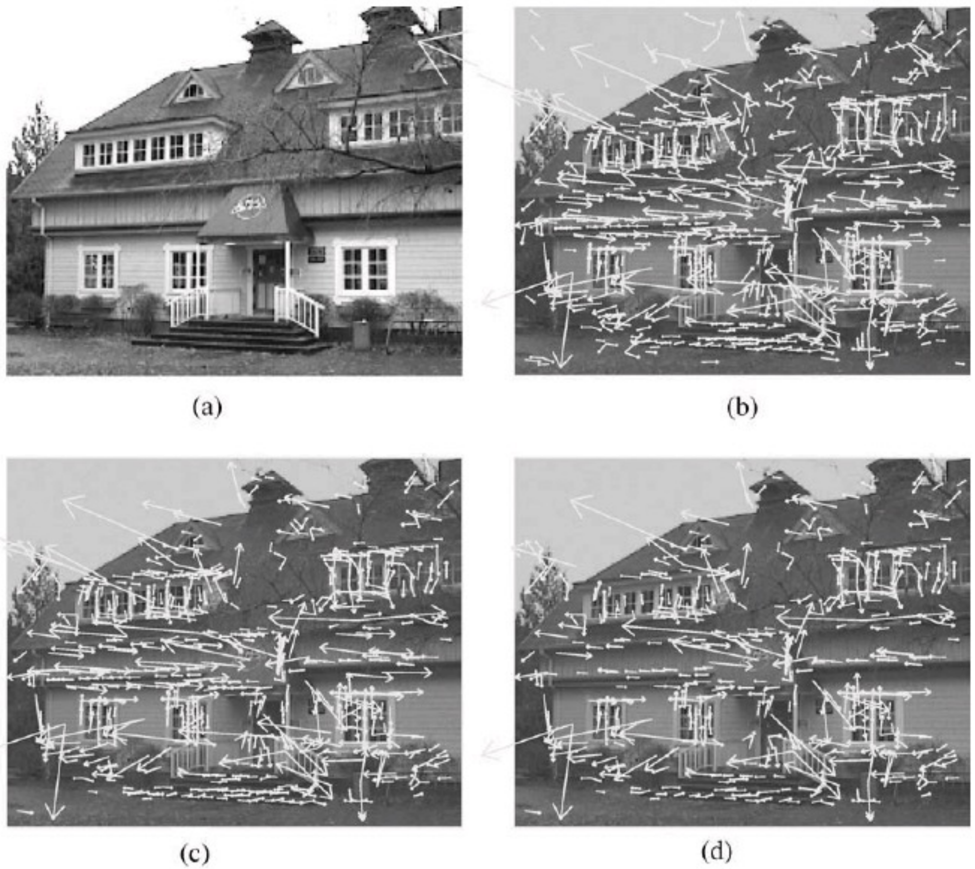
Eliminating Edge Responses…

- Threshold the ratio to remove the edge points

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r_0+1)^2}{r_0}$$

$$\boldsymbol{Typical\, r_0 = 10}$$

Stages of keypoint selection



Initial 832 keypoints

After applying a threshold on minimum contrast, 729 keypoints remain

The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures

Figure 5. This figure shows the stages of keypoint selection. (a) The 233 × 189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

Ref.[3]

Assigning an Orientation

- We finally have a keypoint and its scale that we are going to keep
- The next step is assigning an orientation for the keypoint
  - Used in making the matching technique invariant to rotation

Assigning an Orientation…

- Gaussian smoothed image, *L*, with closest scale is chosen (scale invariance)
- Points in region around the keypoint are selected and magnitude and orientations of the gradient are calculated

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

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

Keypoint Descriptor

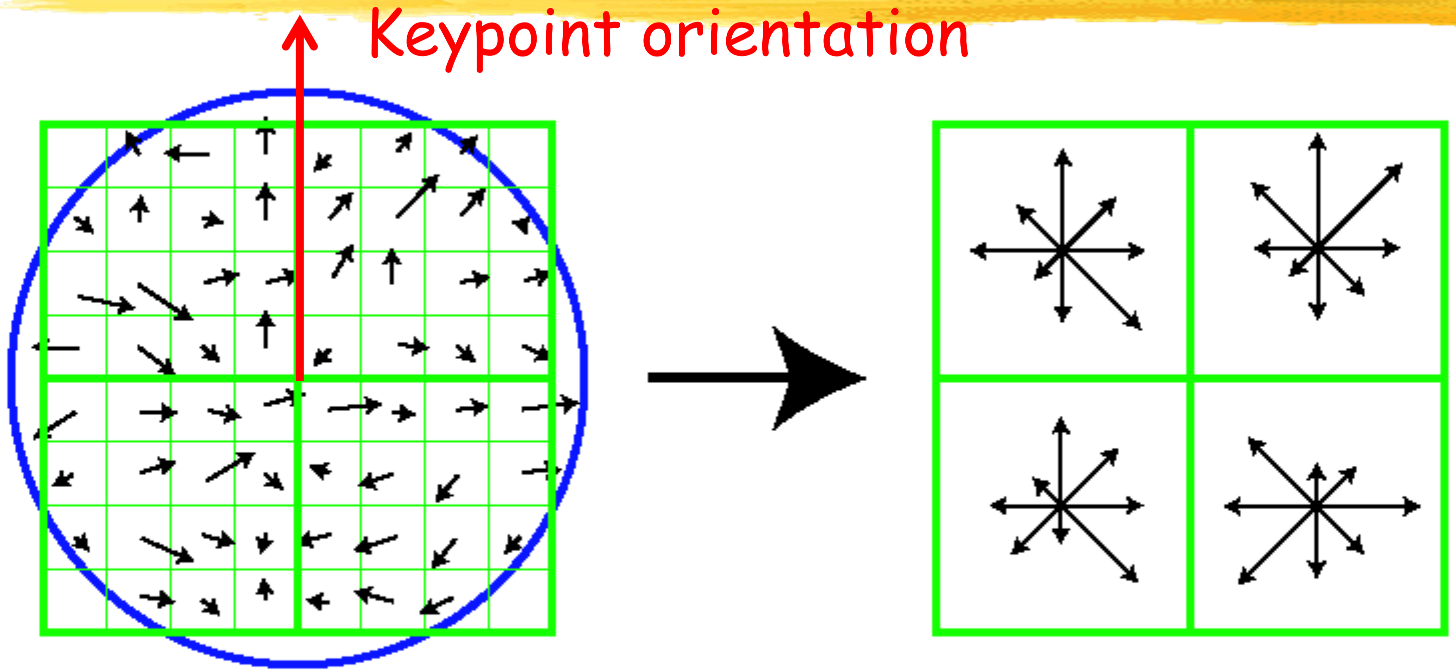


Image gradients

Keypoint descriptor

2x2 array (subregions) of orientation histograms, each has 8 orientation bins

Keypoint Descriptor…

- To make the descriptor robust to orientation,
  - the coordinates of the descriptor and gradient orientations are rotated relative to the key point orientation
- Each point in the subregions added to the histogram is weighted by
  - gradient magnitude
  - with σ of one half the width of the descriptor window
  - 1 – d , where d is the distance of a specific sample to the center of a bin

Keypoint Descriptor…

- Lowe found the best configuration is
  - 4x4 subregions with 8 bins, which results in 4x4x8=128 elements in the feature descriptor
- Vector normalization
  - Done at the end to ensure invariance to illumination change (affine)
  - Entire vector normalized to 1
  - To combat non-linear illumination (camera saturation) changes values in feature vector are thresholded to no larger than 0.2 and then the vector is re-normalized.

Summary on SIFT

- SIFT:
  - Found rough approximations for features by looking at the DoGs
  - Localized the keypoint more accurately
  - Removed poor keypoints
  - Determined the orientation of a keypoint
  - Calculated a 128 feature vector for each keypoint
  - (x, y, scale, orientation, 128 visual descriptor)

What to do with the features?

- SIFT:
  - Localized the keypoints
  - Removed poor keypoints
  - Determined the orientation of a keypoint
  - Calculated a 128D feature vector for each keypoint
  - (x, y, scale, orientation, 128D visual descriptor)

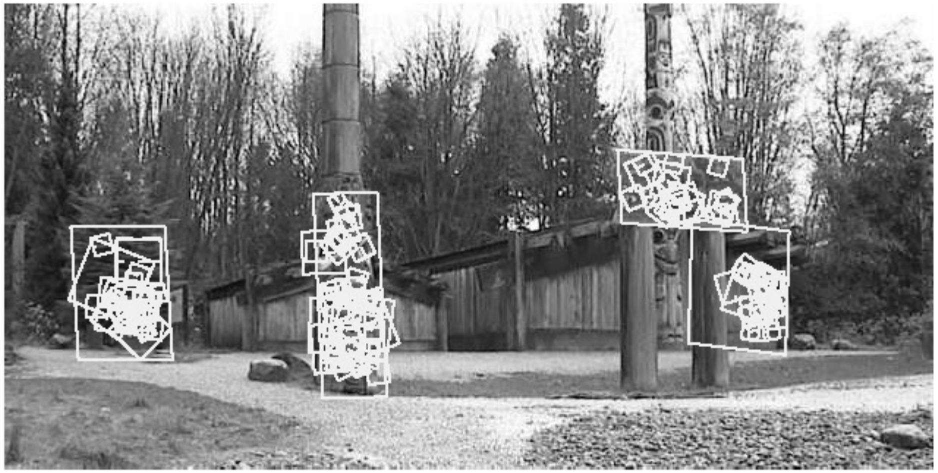
- What do we do now?

Object Detection

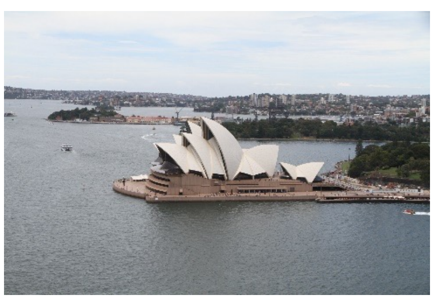
- Two images
  - One image is the training sample of the object we are looking for
  - The other image is the world picture that might contain instances of the training sample
  - Both images have features associated with them across different octaves
- Search and match features between the two
  - Many methods are available, See [Lowe2004] for a typical one

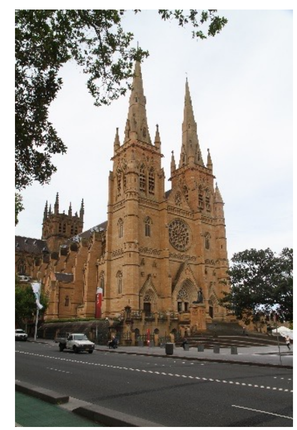
Examples

Examples…



Comparison of Images





Comparison of Images

- Given N images, $I_i, i \in [1, N]$, how similar is between I_i and I_j ?
- The similarity should be invariant to
  - Translation & rotation
  - Illumination changes
  - Scale
  - Some degree of change of viewpoints
  - cameras

Comparison of Images…

- Extract SIFT from each image, for I_i , there are n_i key points, its SIFT is

$$SIFT_i = \{f_{i1}, f_{i2}, \cdots, f_{in_i}\}$$

- In other words, each image is represented by a set of features, with each feature being 128 dimensions.
- The similarity or dissimilarity of two images can be measured by the distance between two SIFTs, i.e.

$$d_{ij} = dist(SIFT_i, SIFT_j)$$

  - *dist(.)* measures the distance between two sets, e.g. hausdorff distance

Hausdorff Distance

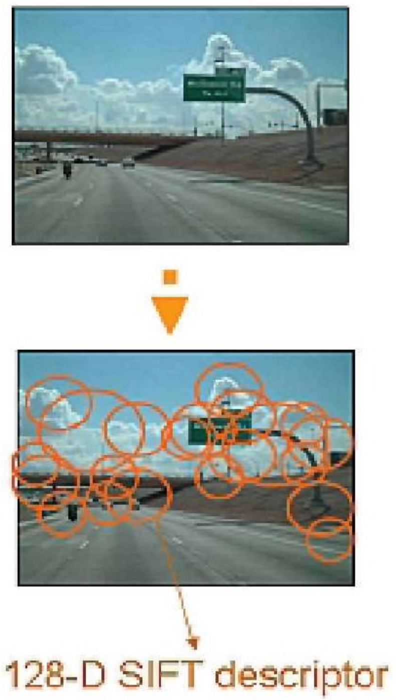
- $I_i = \{f_{i1}, f_{i2}, \cdots f_{in_i}\}$ and $I_j = \{f_{j1}, f_{j2}, \cdots, f_{jn_j}\}$
  - Hausdorff distance

$$H(I_i, I_j) = \max(h(I_i, I_j), h(I_j, I_i))$$

$h(I_i, I_j)$ ranks each interest point of I_i based on its nearest interest point of I_j and uses the most mismatched point

$$h(I_i, I_j) = \max_{f_{ik} \in I_i} \min_{f_{jm} \in I_j} \left\| f_{ik} - f_{jm} \right\|$$

Bag-of-Visual-Words (BoW) Model



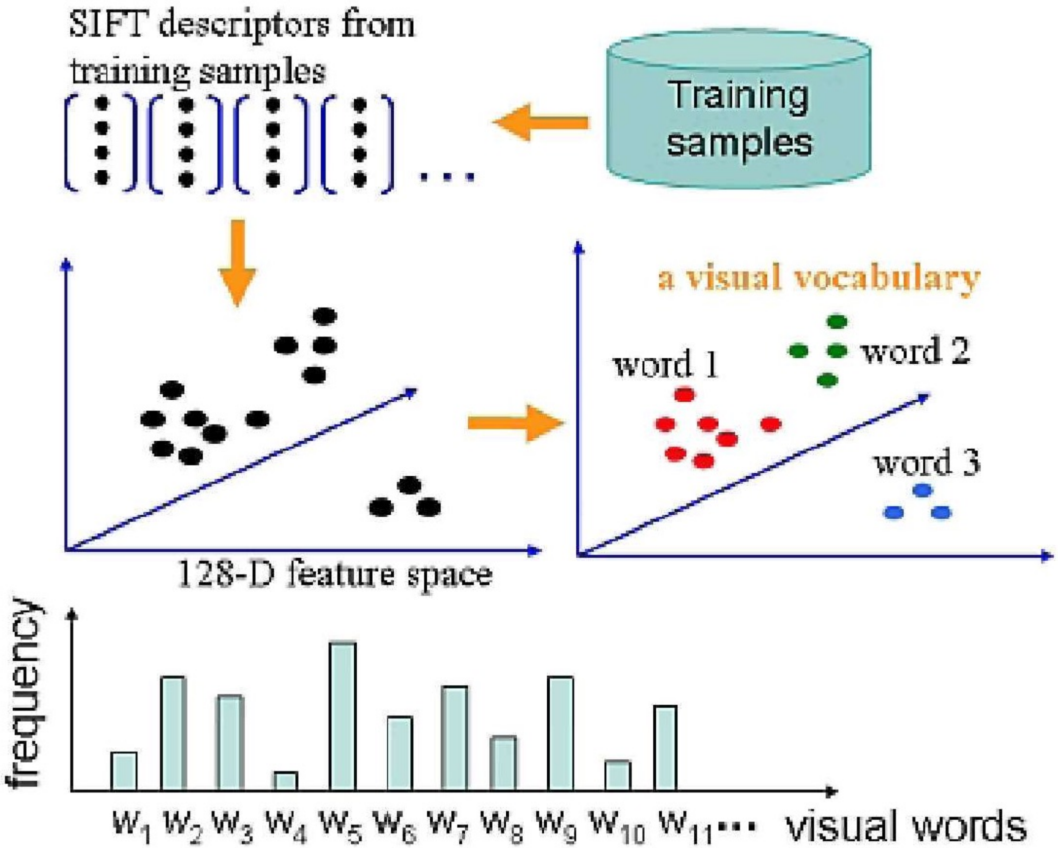


Image Comparison Using BoW

- Represent images using a Bag-of-features, such as SIFT
- Learning visual vocabulary
  - Cluster the features into K clusters, each cluster representing a visual word
  - K-means is often used
- Build a histogram of the visual words for each image
  - Normalize the histogram
- Compare the histograms of images

Note

- K-means
  - http://en.wikipedia.org/wiki/K-means_clustering

- Chi-squared (χ^2) distance.
  - Given two normalized histogram h_1 and h_2,

$$\chi^2(h_1, h_2) = \frac{1}{2}\sum_i \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)}$$

where $\sum_i h_1(i) = 1.0, \sum_i h_2(i) = 1.0$

Other Descriptors

- **HoG** - Histogram of Gradients (CVPR'05)
- **LBP** - Local binary Patterns (ICPR'00, PAMI'02)

- **SURF** – speeded up robust features (Bay et al. 2006)
- **BRIEF** - Binary Robust Independent Elementary Features (ECCV'10)
- **BRISK** - Binary Robust Invariant Scalable Keypoints (ICCV'11)
- **FREAK** - Fast Retina Keypoint (CVPR'12)

Suggested Readings

- E. R Davies, Computer Vision: Principles, Algorithms, Applications, Learning, Academic Press; 5th edition; 2017 - Chapter 6
- David Forsyth and Jean Ponce, *Computer Vision A Modern Approach*, 2012, Chapter 5
- C. Harris, M. Stephens. “A Combined Corner and Edge Detector”, Proceedings of the Fourth Alvey Vision Conference, pp. 147-151, 1998
- David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision 60(2), 91–110, 2004
- H. Bay, T. Tuytelaars, and L. V. Gool, SURF: Speeded Up Robust Features, ECCV 2006
- H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, SURF: Speeded Up Robust Features (SURF), Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008

OpenCV 4.6.0

- Module - features2d
  - `cv::cornerharris(…)`
- Module - non-free features2d
  - `cv:xfeatures2d::SiftFeatureDetector(…)`
  - `cv::xfeatures2d::SurfFeatureDetector(…)`
- Tutorials - Python
  - https://docs.opencv.org/4.6.0/db/d27/tutorial_py_table_of_contents_feature2d.html
  - Harris Corner Detection
  - Introduction to SIFT (Scale-Invariant Feature Transform)
  - FAST Algorithm for Corner Detection
  - Introduction to SURF (Speeded-Up Robust Features)

OpenCV 4.6.0

- Module – Machine Learning
  - cv::kmeans(….)
  - https://docs.opencv.org/4.6.0/d5/d38/group__core__cluster.html#ga9a34dc06c6ec9460e90860f15bcd2f88
- Example code on how to use
  https://docs.opencv.org/4.6.0/d9/dde/samples_2cpp_2kmeans_8cpp-example.html#a17