

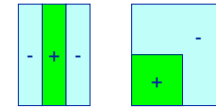
Feature Detection and Matching

- **Goal:** Develop matching procedures that can detect possibly partially-occluded objects or features specified as patterns of intensity values, and are invariant to position, orientation, scale, and intensity change
- Template matching
 - ◆ gray level correlation
 - ◆ edge correlation
- Hough Transform
- Chamfer Matching

1

Applications

- Feature detectors
 - ◆ Line detectors
 - ◆ Corner detectors
 - ◆ Spot detectors
- Known shapes
 - ◆ Character fonts
 - ◆ Faces
- Applications
 - ◆ Image alignment, e.g., Stereo
 - ◆ 3D scene reconstruction
 - ◆ Motion tracking
 - ◆ Object recognition
 - ◆ Image indexing and content-based retrieval



2

Example: Build a Panorama



M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003

3

How do we build panorama?

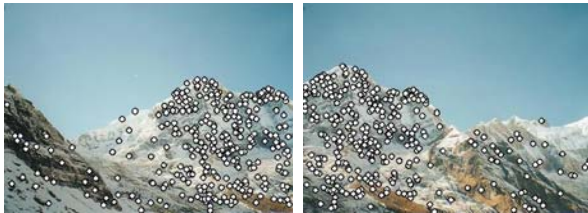
- We need to match (align) images



4

Matching with Features

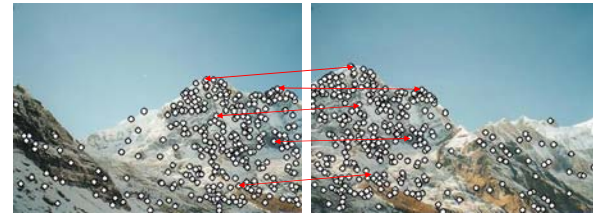
- Detect feature points in both images



5

Matching with Features

- Detect feature points in both images
- Find corresponding pairs



6

Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images



7

Matching with Features

- Problem 1:
 - ◆ Detect the *same* point *independently* in both images



no chance to match!

We need a repeatable detector

8

Matching with Features

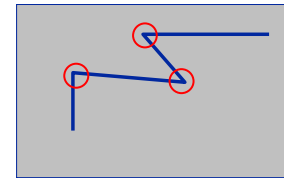
- Problem 2:
 - ◆ For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

9

Harris Corner Detector

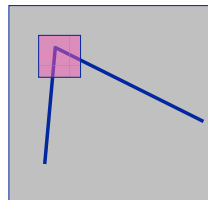


C. Harris, M. Stephens, "A Combined Corner and Edge Detector," 1988

10

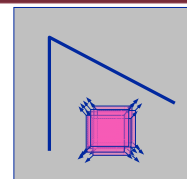
The Basic Idea

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

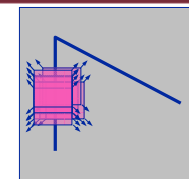


11

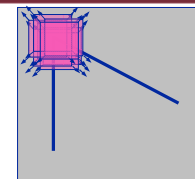
Harris Detector: Basic Idea



“flat” region:
no change in
all directions



“edge”:
no change along
the edge direction



“corner”:
significant change
in *all* directions

12

Harris Detector: Mathematics

Change of intensity for the shift $[u, v]$:

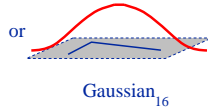
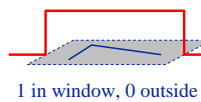
$$E(u, v) = \sum_{x, y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

Window
function

Shifted
intensity

Intensity

Window function $w(x, y) =$



Harris Detector: Mathematics

Expanding $E(u, v)$ in a 2nd order Taylor series, 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:

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

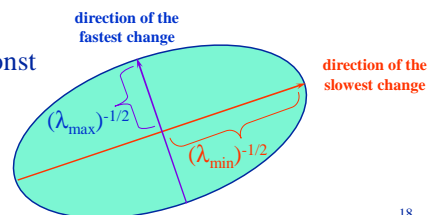
17

Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

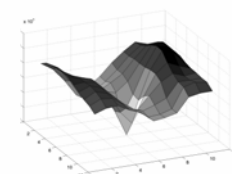
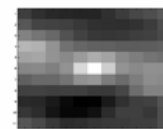
$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix} \quad \lambda_1, \lambda_2 - \text{eigenvalues of } M$$

Ellipse $E(u, v) = \text{const}$



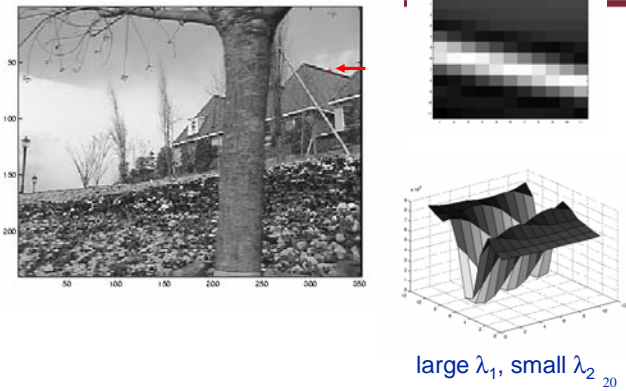
18

Selecting Good Features

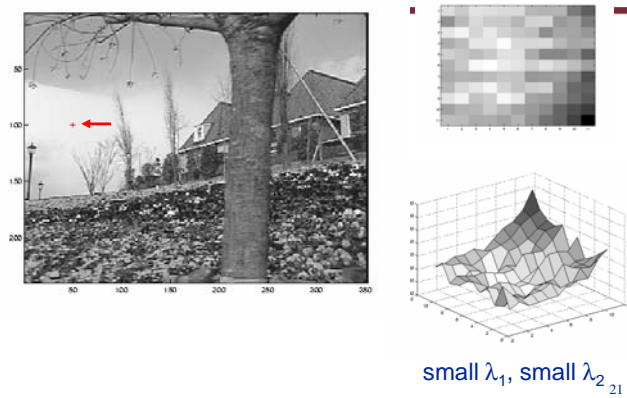


λ_1 and λ_2 are large₁₉

Selecting Good Features



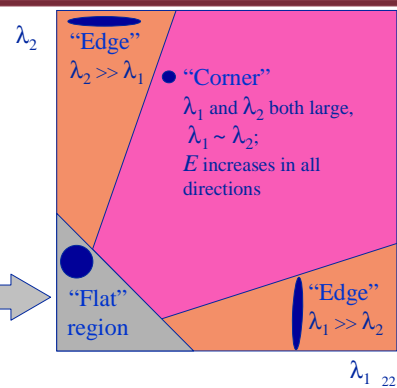
Selecting Good Features



Harris Detector: Mathematics

Classification of image points using eigenvalues of M :

λ_1 and λ_2 are small;
 E is almost constant
in all directions



Harris Detector: Mathematics

Measure of corner response:

$$R = \det M - k (\text{trace } M)^2$$

$$\det M = \lambda_1 \lambda_2$$

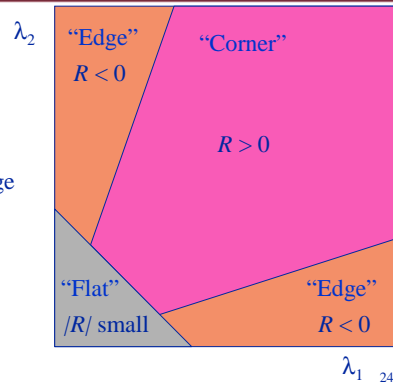
$$\text{trace } M = \lambda_1 + \lambda_2$$

k is an empirically-determined constant; e.g., $k = 0.05$

23

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



λ_1 24

Harris Detector

- The Algorithm:
 - ◆ Find points with large corner response function R ($R > \text{threshold}$)
 - ◆ Take the points of local maxima of R (for localization)

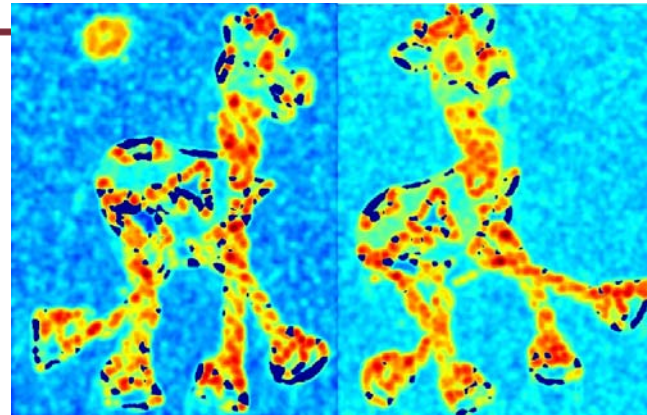
25

Harris Detector: Example



Harris Detector: Example

Compute corner response R



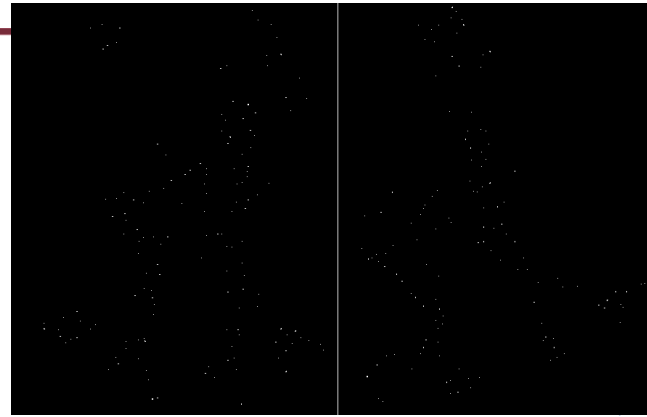
Harris Detector: Example

Find points with large corner response: $R > \text{threshold}$



Harris Detector: Example

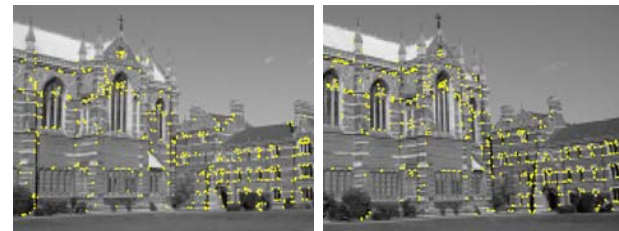
Take only the points of local maxima of R



Harris Detector: Example



Harris Detector: Example



Interest points extracted with Harris (~ 500 points)

31

Harris Detector: Summary

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

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

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

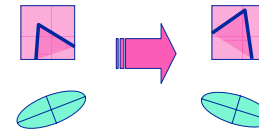
$$R = \lambda_1 \lambda_2 - k (\lambda_1 + \lambda_2)^2$$

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

32

Harris Detector: Some Properties

- Rotation invariance

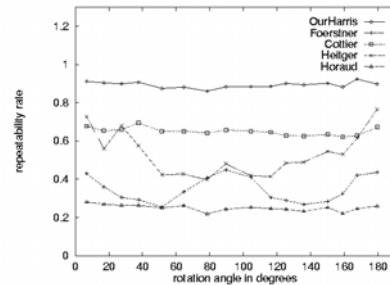
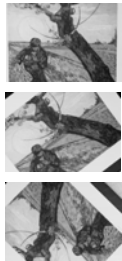


Ellipse rotates but its shape (i.e., eigenvalues) remains the same

Corner response R is invariant to image rotation

33

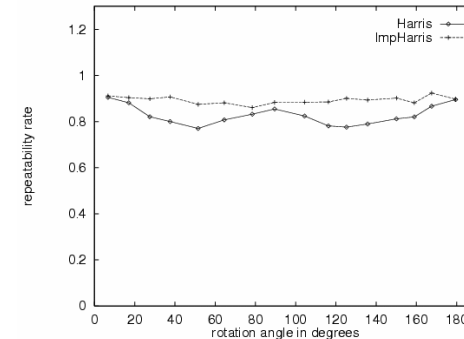
Harris Detector Properties: Rotation Invariance



[Comparing and Evaluating Interest Points, Schmid, Mohr & Bauckhage, ICCV 98]

34

Harris Detector Properties: Rotation Invariance

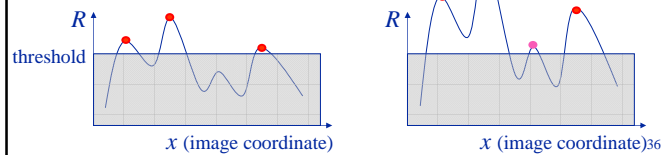


C.Schmid et.al. "Evaluation of Interest Point Detectors". IJCV 2000

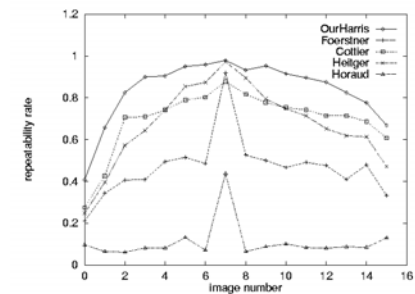
35

Harris Detector Properties: Intensity Changes

- Partial invariance to *affine intensity change*
 - ✓ Only derivatives are used \Rightarrow invariance to intensity shift $I \rightarrow I + b$
 - ✓ Intensity scale: $I \rightarrow a I$



Harris Detector Properties: Perspective Changes

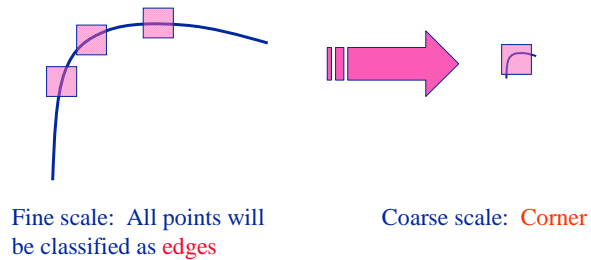


[Comparing and Evaluating Interest Points, Schmid, Mohr & Bauckhage, ICCV 98]

37

Harris Detector Properties: Scale Changes

- But **not** invariant to *image scale*



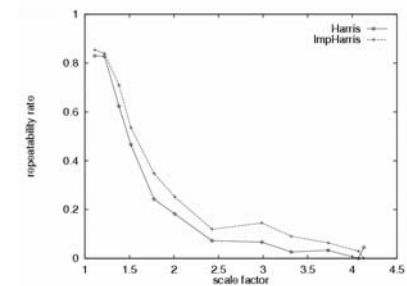
38

Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability rate:

$$\frac{\# \text{ correspondences}}{\# \text{ possible correspondences}}$$



C. Schmid et al., "Evaluation of Interest Point Detectors," IJCV 2000

39

Tomasi and Kanade's Corner Detector

- Idea: Intensity surface has 2 directions with significant intensity discontinuities
- Image gradient $[I_x, I_y]^T$ gives information about direction and magnitude of one direction, but not two
- Compute 2 x 2 matrix

$$M = \begin{bmatrix} \sum_Q I_x^2 & \sum_Q I_x I_y \\ \sum_Q I_x I_y & \sum_Q I_y^2 \end{bmatrix}$$

where Q is a $2n+1 \times 2n+1$ neighborhood of a given point p

40

Corner Detection (cont.)

- ◆ Diagonalize M converting it to the form

$$M = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- ◆ Eigenvalues λ_1 and λ_2 , $\lambda_1 \geq \lambda_2$, give measure of the edge strength (i.e., magnitude) of the two strongest, perpendicular edge directions (specified by the eigenvectors of M)
- ◆ If $\lambda_1 \approx \lambda_2 \approx 0$, then p 's neighborhood is approximately constant intensity
- ◆ If $\lambda_1 > 0$ and $\lambda_2 \approx 0$, then single step edge in neighborhood of p
- ◆ If $\lambda_2 > \text{threshold}$ and no other point within p 's neighborhood has greater value of λ_2 , then mark p as a corner point

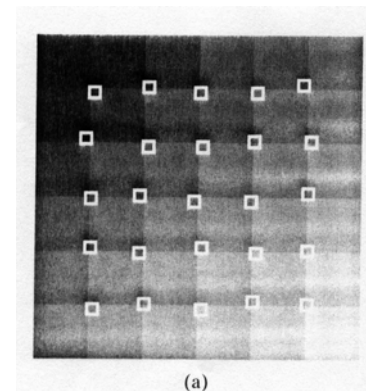
41

Tomasi and Kanade Corner Algorithm

- Compute the image gradient over entire image
- For each image point p :
 - ◆ form the matrix M over $(2N+1) \times (2N+1)$ neighborhood Q of p
 - ◆ compute the smallest eigenvalue of M
 - ◆ if eigenvalue is above some threshold, save the coordinates of p in a list L
- Sort L in decreasing order of eigenvalues
- Scanning the sorted list top to bottom: For each current point, p , delete all other points on the list that belong to the neighborhood of p

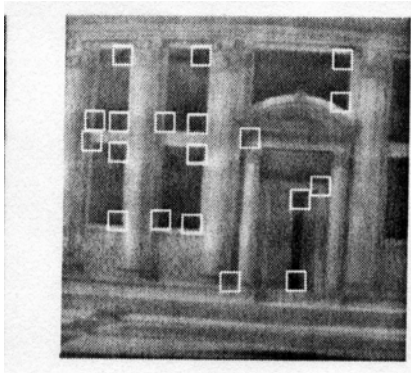
42

Results



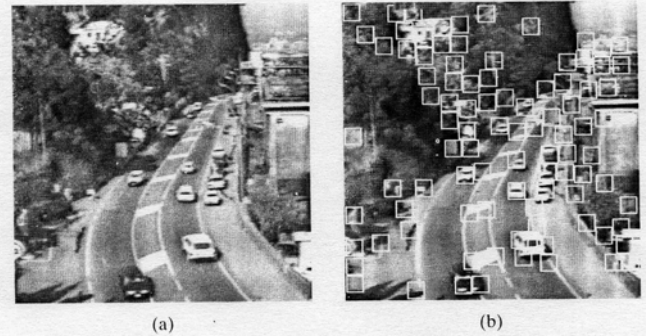
43

Results



44

Results



(a)

(b)

Moravec's Interest Operator

- Compute four directional variances in horizontal, vertical, diagonal and anti-diagonal directions for each 4 x 4 window
- If the minimum of four directional variances is a local maximum in a 12 x 12 overlapping neighborhood, then that window (point) is "interesting"

46

$$V_h = \sum_{j=0}^3 \sum_{i=0}^2 (P(x+i, y+j) - P(x+i+1, y+j))^2$$

$$V_v = \sum_{j=0}^2 \sum_{i=0}^3 (P(x+i, y+j) - P(x+i, y+j+1))^2$$

$$V_d = \sum_{j=0}^2 \sum_{i=0}^2 (P(x+i, y+j) - P(x+i+1, y+j+1))^2$$

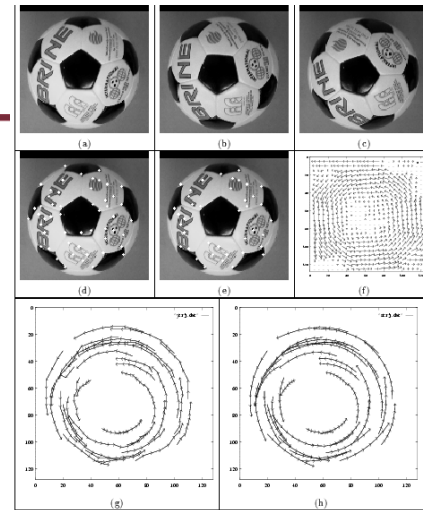
$$V_a = \sum_{j=0}^2 \sum_{i=1}^3 (P(x+i, y+j) - P(x+i-1, y+j+1))^2$$

47

$$V(x, y) = \min(V_h(x, y), V_v(x, y), V_d(x, y), V_a(x, y))$$

$$I(x, y) = \begin{cases} 1, & \text{if } V(x, y) \text{ is a local max} \\ 0, & \text{otherwise} \end{cases}$$

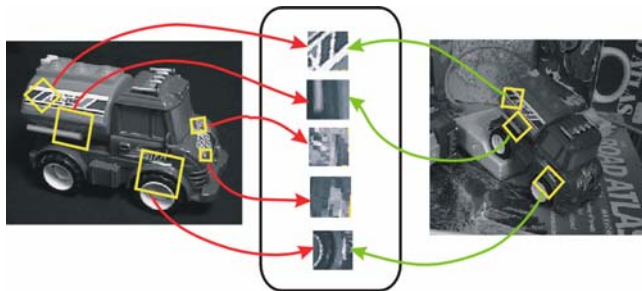
48



49

Invariant Local Features




- **Goal:** Detect *the same* interest points regardless of *image changes* due to translation, rotation, scale, etc.



50

Models of Image Change

• Geometry

- ◆ Rotation 
- ◆ Similarity (rotation + uniform scale) 
- ◆ Affine (scale dependent on direction) 
valid for: orthographic camera, locally planar object

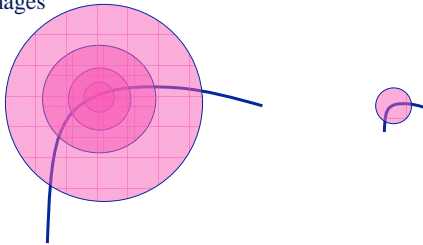
• Photometry

- ◆ Affine intensity change ($I \rightarrow aI + b$) 

51

Scale Invariant Detection

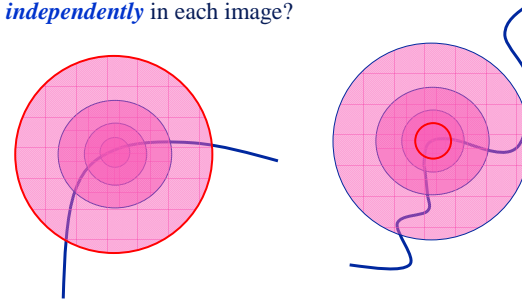
- Consider regions (e.g., circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



52

Scale Invariant Detection

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



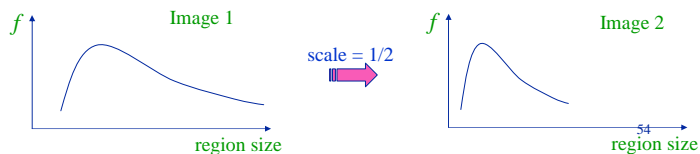
53

Scale Invariant Detection

- Solution:
 - Design a function on the region (circle) that is "scale invariant," i.e., the same for corresponding regions, even if they are at different scales

Example: Average intensity. For corresponding regions (even of different sizes) it will be the same

- For a point in one image, we can consider it as a function of region size (circle radius)



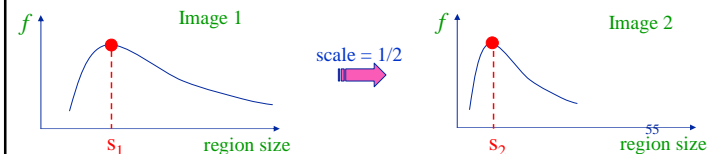
54

Scale Invariant Detection

- Common approach: Take a local maximum of this function

Observation: Region size, for which the maximum is achieved, should be *invariant* to image scale

Important: This scale invariant region size is found in each image *independently*!



55

Scale Invariant Detection

- A “good” function for scale detection has one stable sharp peak



- For many images: a good function would be a one which responds to contrast (sharp local intensity change)

56

Scale Invariance

Requires a method to repeatably select points in location and scale:

- The only reasonable scale-space kernel is a Gaussian (Koenderink, 1984; Lindeberg, 1994)
- An efficient choice is to detect peaks in the Laplacian (DoG) Pyramid (Burt & Adelson, 1983; Crowley & Parker, 1984 – but examining more scales)
- Difference-of-Gaussian with constant ratio of scales is a close approximation to Lindeberg’s scale-normalized Laplacian (can be shown from the heat diffusion equation)

57

Scale Invariant Detection

- Functions for determining scale

Kernels:

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$$

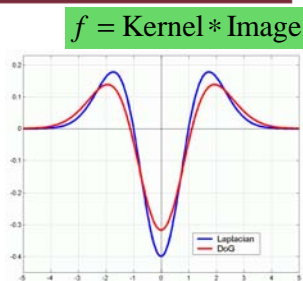
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

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

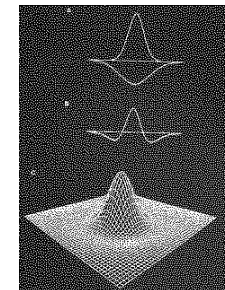


Note: both kernels are invariant to scale and rotation

58

Scale Invariant Detection

- Compare to human vision: eye’s response



59

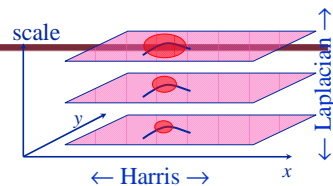
Shimon Ullman, Introduction to Computer and Human Vision Course, Fall 2003

Scale Invariant Detectors

- **Harris-Laplacian¹**

Find local maximum of:

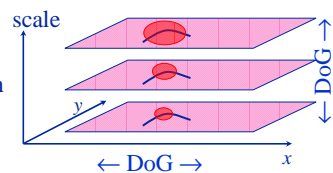
- ◆ Harris corner detector in space (image coordinates)
- ◆ Laplacian in scale



- **SIFT keypoints²**

Find local extrema of:

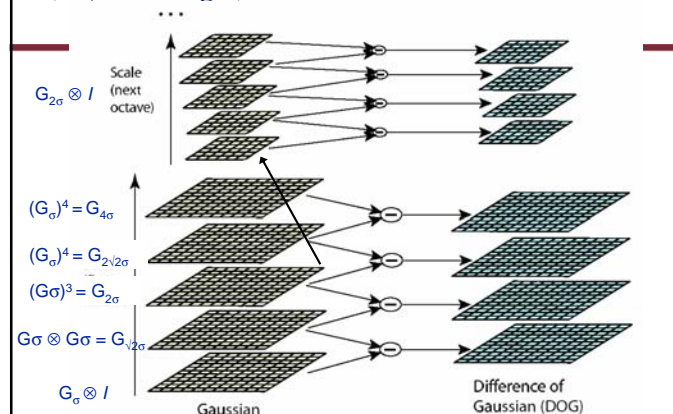
- Difference of Gaussians in space and scale



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 ⁶⁰

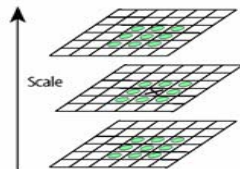
SIFT Operator: Scale Space Processed One Octave (i.e., Doubling σ) at a Time



61

SIFT: Key Point Localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Fit a quadratic to surrounding values for sub-pixel and sub-scale interpolation (Brown & Lowe, 2002)



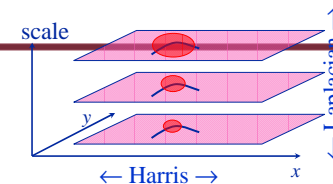
62

Scale Invariant Detectors

- **Harris-Laplacian¹**

Find local maximum of:

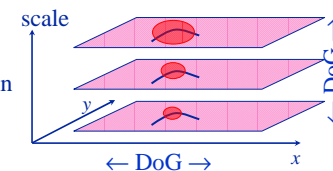
- ◆ Harris corner detector in space (image coordinates)
- ◆ Laplacian in scale



- **SIFT (Lowe)²**

Find local maximum of:

- Difference of Gaussians in space and scale

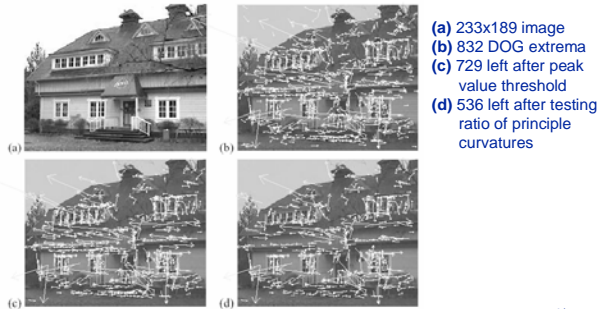


¹ K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points," ICCV 2001

² D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints," IJCV 2004 ⁶³

Example of SIFT Keypoint Detection

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



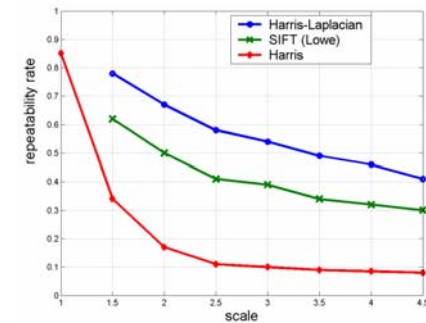
64

Scale Invariant Detectors

- Experimental evaluation of detectors w.r.t. scale change

Repeatability rate:

$$\frac{\# \text{ correspondences}}{\# \text{ possible correspondences}}$$



K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points," ICCV 2001

Scale Invariant Detection: Summary

- Given:** Two images of the same scene with a large *scale difference* between them
- Goal:** Find *the same* interest points *independently* in each image
- Solution:** Search for *maxima* of suitable functions in *scale* and in *space* (over the image)

Methods:

- Harris-Laplacian [Mikolajczyk, Schmid]: Maximize Laplacian over scale, Harris' measure of corner response over the image
- SIFT [Lowe]: Maximize Difference-of-Gaussians over scale and space

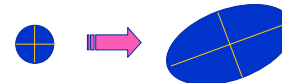
67

Affine Invariant Detection

- Previously we considered:
Similarity transform (rotation + uniform scale)



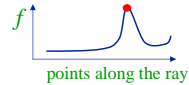
- Now we go on to:
Affine transform (rotation + non-uniform scale)



68

Affine Invariant Detection

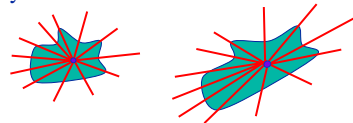
- Take a local intensity extremum as initial point
- Go along every ray starting from this point and stop when extremum of function f is reached



$$f(t) = \frac{|I(t) - I_0|}{\frac{1}{t} \int_0^t |I(t) - I_0| dt}$$

- We will obtain approximately corresponding regions

Remark: we search for scale in every direction



T.Tuytelaars, L.V.Gool. "Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions," BMVC 2000

69

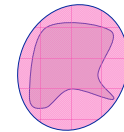
Affine Invariant Detection

- The regions found may not exactly correspond, so we approximate them with ellipses
- Geometric Moments:

$$m_{pq} = \int_{\Omega} x^p y^q f(x, y) dx dy$$

Fact: moments m_{pq} uniquely determine the function f

Taking f to be the characteristic function of a region (1 inside, 0 outside), moments of orders up to 2 allow to approximate the region by an ellipse



This ellipse will have the same moments of orders up to 2 as the original region

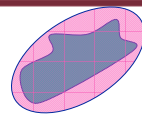
70

Affine Invariant Detection

- Covariance matrix of region points defines an ellipse:



$$q = Ap$$



$$p^T \Sigma_1^{-1} p = 1$$

$$q^T \Sigma_2^{-1} q = 1$$

$$\Sigma_1 = \langle pp^T \rangle_{\text{region 1}}$$

$$\Sigma_2 = \langle qq^T \rangle_{\text{region 2}}$$

($p = [x, y]^T$ is relative to the center of mass)

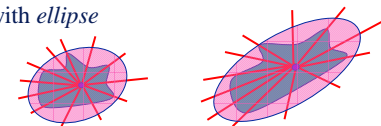
$$\Sigma_2 = A \Sigma_1 A^T$$

Ellipses, computed for corresponding regions, also correspond!

71

Affine Invariant Detection

- Algorithm Summary (detection of affine invariant regions):
 - ◆ Start from a *local intensity extremum* point
 - ◆ Go in *every direction* until the point of extremum of some function f
 - ◆ Curve connecting the points is the region boundary
 - ◆ Compute *geometric moments* of orders up to 2 for this region
 - ◆ Replace the region with *ellipse*



T.Tuytelaars, L.V.Gool. "Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions," BMVC 2000

72

Affine Invariant Detection

- Maximally Stable Extremal Regions
 - Threshold image intensities: $I > I_0$
 - Extract *connected components* ("Extremal Regions")
 - Find a threshold when an extremal region is "Maximally Stable," i.e., a *local minimum* of the relative growth of its square
 - Approximate region with an *ellipse*

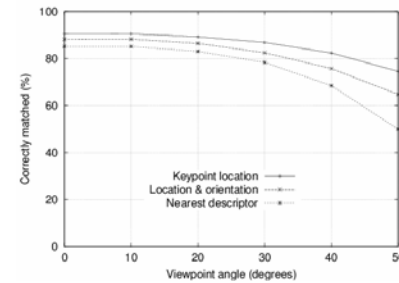


J.Matas et al. "Distinguished Regions for Wide-baseline Stereo," 2001

73

Feature Stability to Affine Change

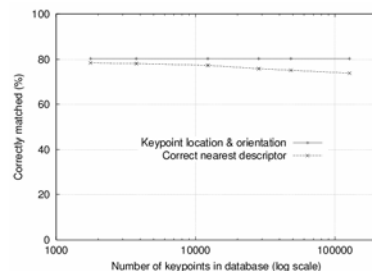
- Match features after random change in image scale and orientation, with 2% image noise, and affine distortion
- Find nearest neighbor in database of 30,000 features



74

Distinctiveness of Features

- Vary size of database of features, with 30 degree affine change, 2% image noise
- Measure % correct for single nearest-neighbor match



75

Affine Invariant Detection : Summary

- Under affine transformation, we do not know in advance shapes of the corresponding regions
- Ellipse given by geometric *covariance matrix* of a region robustly approximates this region
- For corresponding regions ellipses also correspond

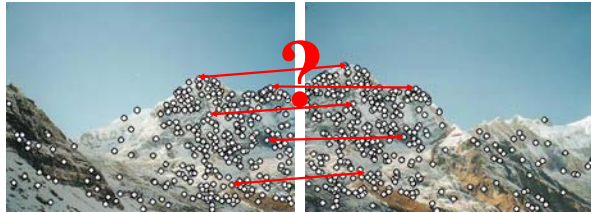
Methods:

- Search for extremum along rays [Tuytelaars, Van Gool]
- Maximally Stable Extremal Regions [Matas et al.]

76

Feature Point Descriptors

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



Point descriptor should be:

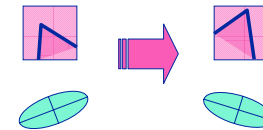
1. Invariant
2. Distinctive

77

Descriptors Invariant to Rotation

- Harris corner response measure:
depends only on the eigenvalues of the matrix M

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



C.Harris, M.Stephens. "A Combined Corner and Edge Detector". 1988

78

Descriptors Invariant to Rotation

- Find local orientation

Dominant direction of gradient



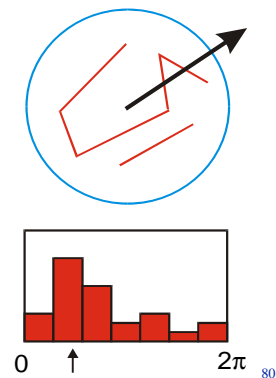
- Compute description relative to this orientation

¹ K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001

² D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". Accepted to IJCV 2004

SIFT: Select Canonical Orientation

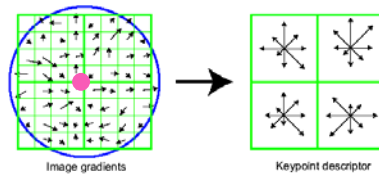
- Create histogram of local gradient directions computed at selected scale of Gaussian pyramid in neighborhood of a keypoint
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)



SIFT Keypoint Feature Representation

Descriptor overview:

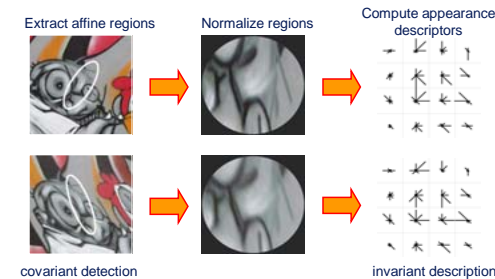
- ◆ Compute gradient orientation histograms on 4×4 neighborhoods, relative to the keypoint orientation using thresholded image gradients from Gaussian pyramid level at keypoint's scale
- ◆ Quantize orientations to 8 values
- ◆ 2×2 array of histograms
- ◆ SIFT feature vector of length $4 \times 4 \times 8 = 128$ values for each keypoint
- ◆ Norm



D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints," IJCV 2004

81

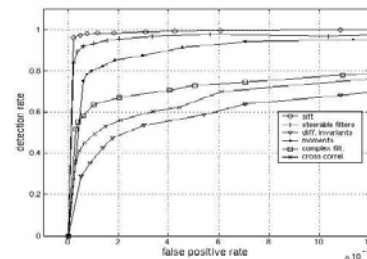
Describing Local Appearance



SIFT – Scale Invariant Feature Transform¹

- ◆ Empirically found² to show very good performance, invariant to *image rotation, scale, intensity change*, and to moderate *affine* transformations

Scale = 2.5
Rotation = 45°

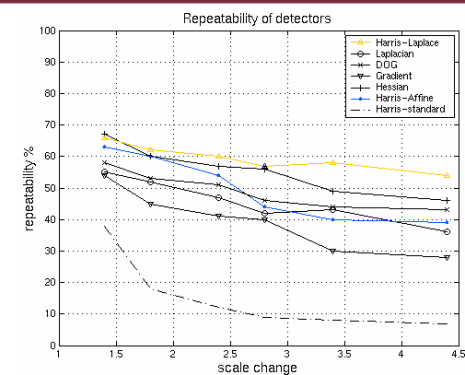


¹D.Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," IJCV 2004

84

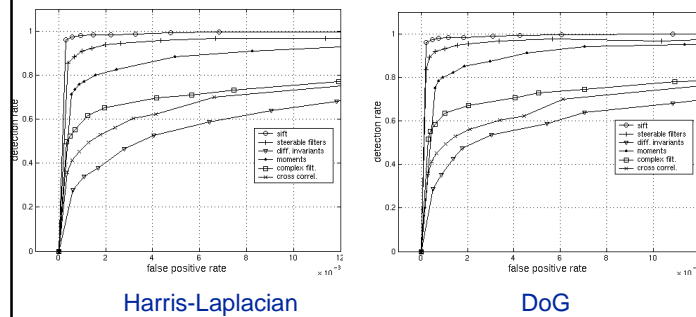
²K.Mikolajczyk, C.Schmid, "A Performance Evaluation of Local Descriptors," CVPR 2003

Evaluation of scale invariant detectors repeatability – scale changes



85

Invariance to Scale Change (factor 2.5)



86

Quantitative Evaluation of Descriptors

- Evaluation of different local features
 - ◆ SIFT, steerable filters, differential invariants, moment invariants, cross-correlation
- Measure : distinctiveness
 - ◆ receiver operating characteristics of detection rate with respect to false positives
 - ◆ detection rate = correct matches / possible matches
 - ◆ false positives = false matches / (database points * query points)

[A performance evaluation of local descriptors, Mikolajczyk & Schmid, CVPR'03]

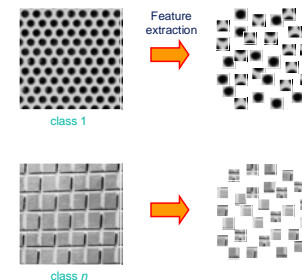
87

Feature Detection and Description Summary

- Stable (repeatable) feature points can be detected regardless of image changes
 - ◆ **Scale**: search for correct scale as *maximum* of appropriate function
 - ◆ **Affine**: approximate regions with *ellipses* (this operation is affine invariant)
- Invariant and distinctive descriptors can be computed
 - ◆ Invariant *moments*
 - ◆ *Normalizing* with respect to scale and affine transformation

88

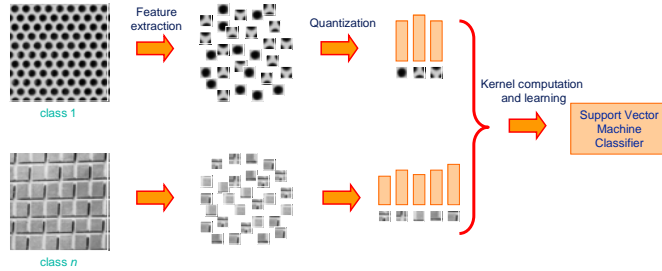
Local models for texture recognition



Lazebnik, Schmid & Ponce, CVPR 2003 and PAMI 2005

89

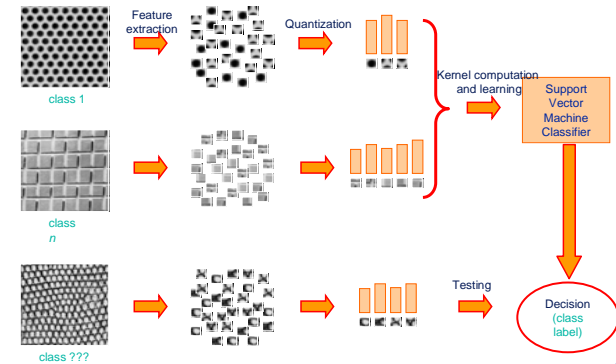
Local models for texture recognition



Lazebnik, Schmid & Ponce, CVPR 2003 and PAMI 2005

90

Local models for texture recognition



Lazebnik, Schmid & Ponce, CVPR 2003 and PAMI 2005

91

Image Correlation

- **Given:**
 - ◆ $n \times n$ image, M , of an object of interest, called a **template**
 - ◆ $n \times n$ image, N , that possibly contains that object (usually a window of a larger image)
- **Goal:** Develop functions that compare images M and N and measure their **similarity**
 - ◆ Sum-of-Squared-Difference (SSD):

$$SSD(k, l) = \sum_{i=1}^n \sum_{j=1}^n [M(i-k, j-l) - N(i, j)]^2$$

- ◆ (Normalized) Cross-Correlation (CC):

$$CC(k, l) = \frac{\sum_{i=1}^n \sum_{j=1}^n M(i-k, j-l) N(i, j)}{[\sum_{i=1}^n \sum_{j=1}^n M(i, j)^2 \sum_{i=1}^n \sum_{j=1}^n N(i, j)^2]^{1/2}}$$

92

Sum-of-Squared-Difference (SSD)

- Perfect match: $SSD = 0$
- If $N = M + c$, $SSD = c^2 n^2$, so sensitive to constant illumination change in image N . Fix by grayscale normalization of N before SSD

93

Cross-Correlation (CC)

- CC measure takes on values in the range [0, 1] (or [0, $\sqrt{\Sigma \Sigma M^2}$] if first term in denominator removed)
 - ♦ it is 1 if and only if $N = cM$ for some constant c
 - ♦ so N can be uniformly brighter or darker than the template, M , and the correlation will still be high
 - ♦ SSD is sensitive to these differences in overall brightness
 - ♦ The first term in the denominator, $\Sigma \Sigma M^2$, depends only on the template, and can be ignored because it is constant
 - ♦ The second term in the denominator, $\Sigma \Sigma N^2$, can be eliminated if we first normalize the gray levels of N so that their total value is the same as that of M - just scale each pixel in N by $\Sigma \Sigma M / \Sigma \Sigma N$
 - ♦ practically, this step is sometimes ignored, or M is scaled to have average gray level of the big image from which the unknown images, N , are drawn

94

Cross-Correlation

- Suppose that $N(i,j) = cM(i,j)$

$$\begin{aligned}
 CC &= \frac{\sum_{i=1}^n \sum_{j=1}^n M(i,j)N(i,j)}{[\sum_{i=1}^n \sum_{j=1}^n M(i,j)^2 \sum_{i=1}^n \sum_{j=1}^n N(i,j)^2]^{1/2}} \\
 &= \frac{\sum_{i=1}^n \sum_{j=1}^n cN(i,j)N(i,j)}{[\sum_{i=1}^n \sum_{j=1}^n c^2 N(i,j)^2 \sum_{i=1}^n \sum_{j=1}^n N(i,j)^2]^{1/2}} \\
 &= \frac{c \sum_{i=1}^n \sum_{j=1}^n N(i,j)^2}{c [\sum_{i=1}^n \sum_{j=1}^n N(i,j)^2 \sum_{i=1}^n \sum_{j=1}^n N(i,j)^2]^{1/2}} \\
 &= 1
 \end{aligned}$$

95

Cross-Correlation

- Alternatively, we can rescale both M and N to have unit total intensity
 - ♦ $N'(i,j) = N(i,j) / \Sigma \Sigma N$
 - ♦ $M'(i,j) = M(i,j) / \Sigma \Sigma M$
- Now, we can view these new images, M' and N' as unit vectors of length n^2
- The correlation measure $\Sigma \Sigma M'(i,j)N'(i,j)$ is the familiar dot product between the two n^2 vectors M' and N' . Recall that the dot product is the cosine of the angle between the two vectors
 - ♦ it is equal to 1 when the vectors are the same vector, or the normalized images are identical
- These are BIG vectors

96

Cross-Correlation Example 1

- Template $M = \begin{matrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{matrix}$ Image $N = \begin{matrix} 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 \end{matrix}$
 - $\Sigma \Sigma NM = \begin{matrix} 0 \\ 0 & 1 & 2 & 3 & 2 & 1 & 0 \\ 0 \end{matrix}$ $\Sigma \Sigma N^2 = \begin{matrix} 0 \\ 0 \\ 0 & 1 & 2 & 3 & 2 & 1 & 0 \\ 0 \\ 0 \end{matrix}$
 - $\Sigma \Sigma NM / \sqrt{\Sigma \Sigma N^2} = 0 \ 1 \ \sqrt{2} \ \sqrt{3} \ \sqrt{2} \ 1 \ 0$
- NOTE: Many near misses

97

Cross-Correlation Example 2

- Template $M = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ Image $N = \begin{bmatrix} 0 & & & \\ 4 & 4 & 4 & \\ 0 & & & \end{bmatrix}$
- $\sum \sum NM = \begin{bmatrix} 0 & & & \\ 4 & 8 & 12 & 8 & 4 \\ 0 & & & \end{bmatrix}$ $\sum \sum N^2 = \begin{bmatrix} 0 & & & \\ 16 & 32 & 48 & 32 & 16 \\ 0 & & & \end{bmatrix}$
- $\sum \sum NM / \sqrt{\sum \sum N^2} = \begin{bmatrix} 1 & \sqrt{2} & \sqrt{3} & \sqrt{2} & 1 \end{bmatrix}$

98

Cross-Correlation Example 3

- Template $M = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ Image $N = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- $\sum \sum NM = \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$ $\sum \sum N^2 = \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$

99

Example 3 (cont.)

- $\sum \sum NM / \sqrt{\sum \sum N^2} = \begin{bmatrix} 0 & & & & \\ 1 & \sqrt{6}/2 & 1 & & \\ 0 & 0 & 1 & 0 & 0 \\ 1 & \sqrt{6}/2 & 1 & & \\ 0 & & & & \end{bmatrix}$
- Lots of near misses!

100

Reducing the Computational Cost of Correlation Matching

- A number of factors lead to large costs in correlation matching:
 - ♦ the **image N is much larger than the template M** , so we have to perform correlation matching of M against every $n \times n$ window of N
 - ♦ we might have **many templates, M_i** , that we have to compare against a given image N
 - ♦ face recognition - have a face template for **every** known face; this might easily be tens of thousands
 - ♦ character recognition - template for each character
 - ♦ we might not know the **orientation** of the template in the image
 - ♦ template might be rotated in the image N - example: someone tilts their head for a photograph
 - ♦ would then have to perform correlation of rotated versions of M against N

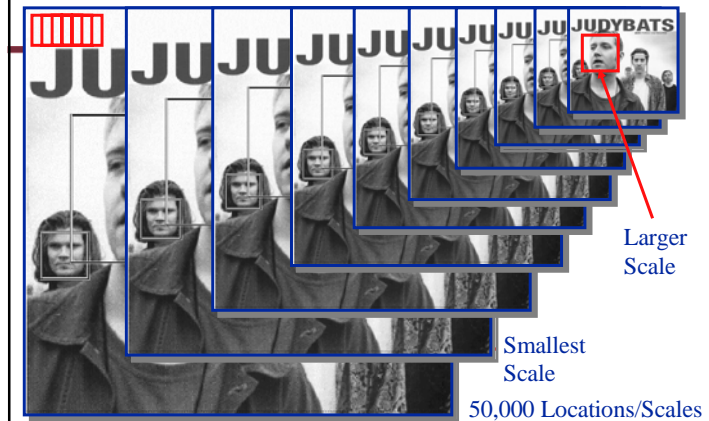
101

Reducing the Computational Cost of Correlation Matching

- A number of factors lead to large costs in correlation matching:
 - ◆ we might not know the **scale**, or size, of the template in the unknown image
 - ◆ the distance of the camera from the object might only be known approximately
 - ◆ would then have to perform correlation of scaled versions of M against N
- Most generally, the image N contains some mathematical transformation of the template image M
 - ◆ if M is the image of a planar pattern, like a printed page or (approximately) a face viewed from a great distance, then the transformation is an **affine transformation**, which has **six** degrees of freedom

102

The Classical Face Detection Process



Slide courtesy of Paul Viola

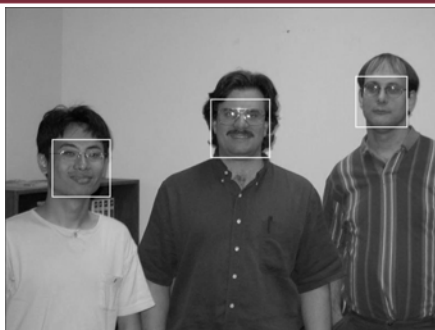
Larger Scale

Smallest Scale

50,000 Locations/Scales

103

Faces as Rare Events



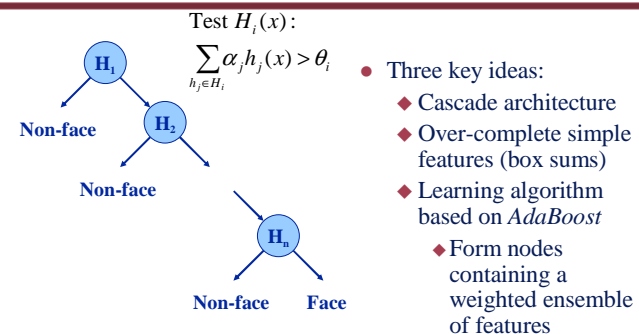
Scanning over all positions and scales for faces requires

2.6 million window evaluations...

...for 3 faces

104

Viola-Jones Face Detector

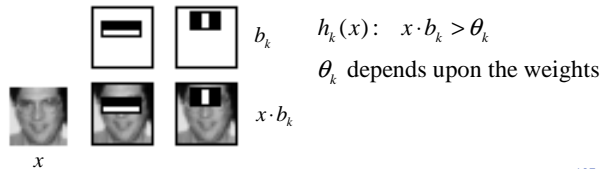


Paul Viola and Michael J. Jones, "Robust Real-Time Face Detection"
Intl. J. Computer Vision, 57(2): 137-154, 2004

105

Weak Classifiers

- Weak classifiers formed from simple "box sum" features applied to input
 - Classifier is trained by setting a threshold, which depends on the training data
- Efficient computation



107

Definition of Simple Features



3 rectangular features types:

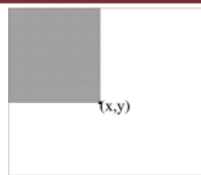
- two-rectangle feature type (horizontal/vertical)
- three-rectangle feature type
- four-rectangle feature type

Using a 24 x 24 pixel detection window, with all the possible combinations of horizontal and vertical locations and scales of these feature types, the full set of features has 49,396 features

The motivation behind using rectangular features, as opposed to more expressive steerable filters is due to their computational efficiency

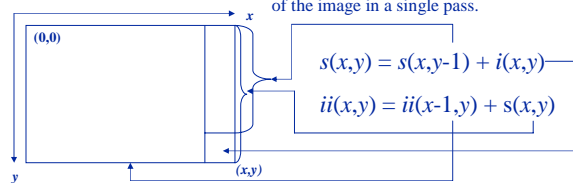
108

Integral Image



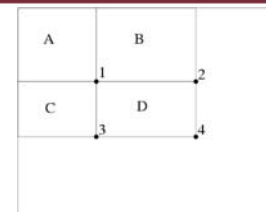
Def: The *integral image* at location (x,y) , is the sum of the pixel values above and to the left of (x,y) , inclusive.

Using the following two recurrences, where $i(x,y)$ is the pixel value of original image at the given location and $s(x,y)$ is the cumulative column sum, we can calculate the integral image representation of the image in a single pass.



109

Rapid Evaluation of Rectangular Features



Using the integral image representation one can compute the value of any rectangular sum in constant time.

For example the integral sum inside rectangle D we can compute as:

$$ii(4) + ii(1) - ii(2) - ii(3)$$

As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references, respectively

110

Experiments (Dataset for Training)

- 4,916 positive training example were hand picked aligned, normalized, and scaled to a base resolution of 24 x 24
- 10,000 negative examples were selected by randomly picking sub-windows from 9,500 images which did not contain faces



116

Experiments (Structure of the Detector Cascade)

- The final detector had 32 layers and 4297 features total

Layer number	1	2	3 to 5	6 and 7	8 to 12	13 to 32
Number of features	2	5	20	50	100	200
Detection rate	100%	100%	-	-	-	-
Rejection rate	60%	80%	-	-	-	-

- Speed of the detector ~ total number of features evaluated
- On the MIT-CMU test set the average number of features evaluated is 8 (out of 4297)
- The processing time of a 384 by 288 pixel image on a conventional PC is about .067 seconds
- Processing time should linearly scale with image size, hence processing of 3.1 megapixel images should take about 2 seconds

117

Correlation Matching

- Let $T(p_1, p_2, \dots, p_r)$ be the class of mathematical transformations of interest
 - ◆ For rotation, we have $T(\theta)$
 - ◆ For scaling, we have $T(s)$
- General goal is to find the values of p_1, p_2, \dots, p_r for which
 - ◆ $C(T(p_1, p_2, \dots, p_r)M, N)$ is "best"
 - ◆ highest for normalized cross-correlation
 - ◆ smallest for SSD

118

Reducing the Computational Cost of Correlation Matching

- Two basic techniques for reducing the number of operations associated with correlation
 - ◆ reduce the number of pixels in M and N
 - ◆ Multi-resolution image representations
 - ◆ principal component or "feature selection" reductions
 - ◆ match a subset of M (i.e., sub-template) against a subset of N
 - ◆ random subsets
 - ◆ boundary subsets

119

Multi-Resolution Correlation

- Multi-resolution template matching
 - ◆ reduce resolution of both template and image by creating a **Gaussian pyramid**
 - ◆ match small template against small image
 - ◆ identify locations of strong matches
 - ◆ expand the image and template, and match higher resolution template selectively to higher resolution image
 - ◆ iterate on higher and higher resolution images
- Issue:
 - ◆ how to choose detection thresholds at each level?
 - ◆ too low will lead to too much cost
 - ◆ too high will miss match

120

Coarse-to-Fine Hierarchical Search

- Selectively process only relevant regions of interest (foveation) and scales
- Iterative refinement
- Variable resolution analysis
- Based on fine-to-coarse operators for computing complex features over large neighborhoods in terms of simpler features in small neighborhoods (e.g., Gaussian pyramid, Laplacian pyramid, texture pyramid, motion pyramid)

121

Efficiency of Multi-Resolution Processing

- For an $n \times n$ image and an $m \times m$ template, correlation requires $O(m^2 n^2)$ arithmetic operations
- To detect at a finer scale, either
 - ◆ Increase scale of template by s , resulting in $O(s^2 m^2 n^2)$ operations
 - ◆ Decrease scale of image by s , resulting in $O(m^2 n^2 / s^2)$ operations
- ◆ These two approaches differ in cost by s^4

122

Pyramid Processing Example

- Goal: Detect moving objects from a stationary video camera
- For each pair of consecutive image frames do:
 - ◆ Compute difference image $D = I_1 - I_2$; compute “energy-change” features
 - ◆ Compute Laplacian pyramid, L , from D ; decompose D into bandpass components
 - ◆ Square values in L ; enhance features
 - ◆ Compute Gaussian pyramid, G , from level k in L ; local integration of feature values which “pools” energy-change within neighborhoods of increasing size -- measures “local energy”
 - ◆ Threshold values in G to determine positions and sizes of detected moving objects

123

Subset (Sub-Template) Matching Techniques

- Sub-template/template matching
 - ◆ choose a subset of the template
 - ◆ match it against the image
 - ◆ compare the remainder of the template at positions of high match
 - ◆ can add pieces of the template iteratively in a multi-stage approach
- Key issues:
 - ◆ what piece(s) to choose?
 - ◆ want pieces that are rare in the images against which we will perform correlation matching so that non-match locations are identified quickly
 - ◆ choose pieces that define the geometry of the object
 - ◆ how to choose detection thresholds at each stage?

124

Subset Matching Methods - Edge Correlation

- Reduce both M and N to **edge maps**
 - ◆ binary images containing "1" where edges are present and "0" elsewhere
 - ◆ associated with each "1" in the edge map we can associate
 - ◆ location (implicitly)
 - ◆ orientation from the edge detection process
 - ◆ color on the "inside" of the edge for the model, M, and on both sides of the edge for the image, N



125

Edge Template Matching

- Simple case
 - ◆ N and M are binary images, with 1 at edge points and 0 elsewhere
 - ◆ The match of M at position (i, j) of N is obtained by
 - ◆ placing M(0, 0) at position N(i, j)
 - ◆ counting the number of pixels in M that are 1 and are coincident with 1's in N - **binary correlation**



$$C(i, j) = \sum_{r=1}^n \sum_{s=1}^n M(r, s) \times N(r + i, s + j)$$

126

Observations

- Complexity of matching M against N is $O(n^2m^2)$ for an $n \times n$ template and $m \times m$ image
 - ◆ to allow rotations of M, must match rotated versions of M against N
 - ◆ to allow for scale changes in M, must match scaled versions of M against N
- Small distortions in the image can give rise to very bad matches
 - ◆ can be overcome by "binary smoothing" (expanding) either the template or the image
 - ◆ but this also reduces the "specificity" of the match



127

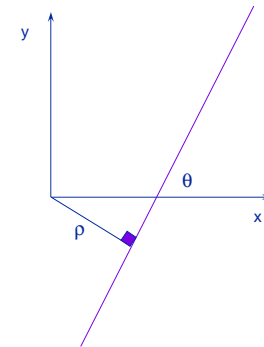
Hough Transform for Line Detection

- Consider the following simple problem:
 - Given: a binary image
 - Find
 - (a) the largest collinear subset of 1's in that binary image
 - (b) all collinear subsets of size greater than a threshold t
 - (c) a set of disjoint collinear subsets of size greater than a threshold t
- Representation of lines
 - $y = mx + b$
 - m is the slope
 - b is the y-intercept
 - problems
 - m is unbounded
 - cannot represent vertical lines

128

Parametric Representation of Lines (ρ, θ)

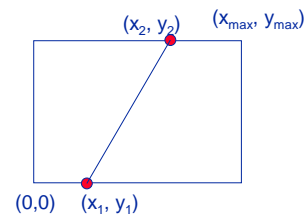
- $\rho = x \cos\theta + y \sin\theta$
- ρ is an unbounded parameter in the representation, but is bounded for any finite image
- θ , the slope parameter, is bounded in the interval $[0, 2\pi]$



129

Parametric Representation of Lines (x, y, x', y')

- Encode a line by the coordinates of its 2 intersections with the boundary of the image
- all parameters are bounded by the image size
- but now we have 4 rather than 2 parameters



130

Brute-Force Solution to Line Detection

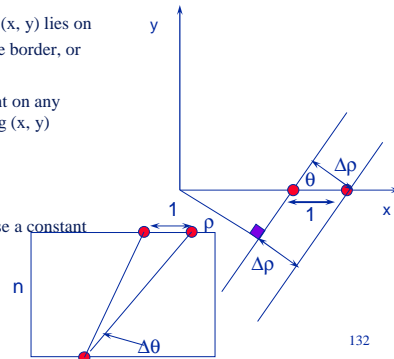
- Brute-force algorithm enumerates L , the set of "all" lines passing through B , the binary input image
 - for each line in L it generates the image pixels that lie on that line
 - it counts the number of those pixels in B that are 1's
 - for problem (a) it remembers the maximal count (and associated line parameters) greater than the required threshold
 - for problem (b) it remembers all that satisfy the threshold requirement.
- So, how do we
 - enumerate L
 - given an element, λ , of L , enumerate the pixels in B that lie on λ

131

Brute-Force Solution

- Enumeration of L

- ◆ (x, y, x', y') - easy: each (x, y) lies on
 - ◆ one side of the image border, or
 - ◆ a corner
 - ◆ (x', y') can be a point on any border not containing (x, y)
- ◆ (ρ, θ) - much harder
 - ◆ $\Delta\rho = \sin \theta$
 - ◆ $\Delta\theta \equiv 1/n$
 - ◆ practically, would use a constant quantization of ρ



132

Generating the Pixels on a Line

- Standard problem in computer graphics
- Compute the intersections of the line with the image boundaries
 - ◆ let the intersection be $(x_1, y_1), (x_2, y_2)$
 - ◆ Compute the “standard” slope of the line
 - ◆ special cases for near vertical line
 - ◆ if the slope is < 1 , then the y coordinate changes more slowly than x, and the algorithm steps through x coordinates, computing y coordinates - depending on slope, might obtain a run of constant y but changing x coordinates
 - ◆ if the slope ≥ 1 , then x changes more slowly than y and the algorithm will step through y coordinates, computing x coordinates

133

Drawbacks of the Brute-Force Algorithm

- The complexity of the algorithm is the sum of the lengths of all of the lines in L
 - ◆ consider the $[(x_1, y_1), (x_2, y_2)]$ algorithm
 - ◆ there are about $3n$ possible locations for (x_1, y_1) and there are $2n$ possible locations for (x_2, y_2) once (x_1, y_1) is chosen (this avoids generating lines twice). This is $6n^2$ lines
 - ◆ It is hard to compute the average length of a line, but it is $O(n)$
 - ◆ So, the brute-force algorithm is $O(n^3)$
- Many of these lines pass through all or almost all 0's
 - ◆ practically, the 1's in our binary image were generated by an edge or feature detector
 - ◆ for typical images, about 3-5% of the pixels lie on edges
 - ◆ so most of the work in generating lines is a waste of time

134

Hough Transform

- Original application was detecting lines in time lapse photographs of bubble chamber experiments
 - ◆ elementary particles move along straight lines, collide, and create more particles that move along new straight trajectories
 - ◆ Hough was the name of the physicist who invented the method
- Turn the algorithm around and *loop on image coordinates* rather than line parameters
- Brute-force algorithm:
 - ◆ For each possible line, generate the line and count the 1's
- Hough transform
 - ◆ For each possible “1” pixel at coordinates (x, y) in B, generate the set of all lines passing through (x, y)

135

Hough Transform

- Algorithm uses an array of accumulators, or counters, H , to tally the number of 1's on any line
 - size of this array is determined by the quantization of the parameters in the chosen line representation
 - we will use the (ρ, θ) representation, so a specific element of H will be referenced by $H(\rho, \theta)$
 - when the algorithm is completed, $H(\rho, \theta)$ will contain the number of points from B that satisfy the equation (i.e. lie on the line) $\rho = x \cos\theta + y \sin\theta$
- Algorithm scans B . Whenever it encounters a "1" at a pixel coordinates (x, y) it performs the following loop:
 - for $\theta := 0$ to 2π step $\Delta\theta$
 - $\rho := x \cos\theta + y \sin\theta$
 - $H[\text{norm}(\rho), \text{norm}(\theta)] := H[\text{norm}(\rho), \text{norm}(\theta)] + 1$
 - norm turns the floats into valid array indices

136

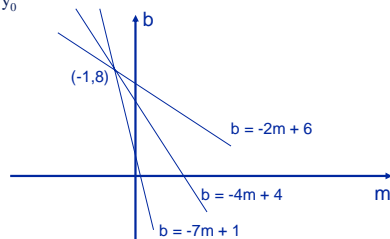
Hough Transform Algorithm

- Quantize parameter space (ρ, θ)
- Create Accumulator Array, $H(\rho, \theta)$
- Initialize H to 0
- Apply voting procedure for each "1" in B
- Find local maxima in H

137

Hough Transform Example

- Let input image B have "1"s at coordinates $(7,1)$, $(6,2)$, and $(4,4)$
- Using slope-intercept parameterization, we have
 - $b = -x_0 m + y_0$



138

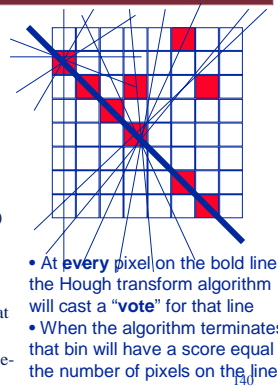
Hough Transform Properties

- Hough space** (*aka parameter space*) has dimensionality equal to the number of degrees of freedom of the parameterized object
- A point in input image maps to a line in (m, b) parameter space, and to a sinusoidal curve in (ρ, θ) parameter space
- A point in H corresponds to a line in image B
- $H(x_0, y_0) = z_0 \Rightarrow z_0$ points are collinear along line in B
- Works when image points are disconnected
- Relatively insensitive to occlusion
- Effective for simple shapes

139

Hough Transform

- What is the computational complexity of the Hough transform?
 - ◆ Scanning the image is $O(n^2)$ and if we encounter a fixed percentage of 1's, we still need to nontrivially process $O(n^2)$ pixels
 - ◆ At each pixel, we have to generate $O(n)$ lines that pass through the pixel
 - ◆ So it is **also** $O(n^3)$ in the worst case
 - ◆ But practically, the Hough transform only does work for those pixels in B that are 1's
 - ◆ This makes it **much** faster than the brute-force algorithm



Solving the Original Problems

- Problem (a) - Find the line having maximal score
 - ◆ Compute the Hough transform
 - ◆ Scan array H for the maximal value; resolve ties arbitrarily
 - ◆ **Problem:** scanning H can be time consuming
 - ◆ Alternatively, can keep track of the location in H having maximal tally as the algorithm proceeds
- Problem (b) - Find all lines having score $> t$
 - ◆ Compute the Hough array
 - ◆ Scan the array for all values $> t$
 - ◆ **Problem:** also requires scanning the array
 - ◆ Can maintain a data structure of above threshold elements of H and add elements to this data structure whenever the algorithm **first** sends an entry of H over t
 - ◆ k-d tree or a point quadtree

141

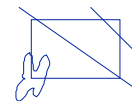
Solving the Original Problems

- Problem (c) - find a set of disjoint lines all of which have size greater than a threshold t
 - ◆ Compute the Hough transform, H
 - ◆ Scan H for the highest value; if it is $< t$, halt. If it is $\geq t$, add it to the set (*)
 - ◆ Remove the "votes" cast by the points on that line
 - ◆ use our line generation algorithm to enumerate the image points on that line
 - ◆ subtract the votes cast for all elements of H by the 1's on that line
 - ◆ this ensures that a point in the image will contribute to the score for one and only one line as the lines are extracted
 - ◆ go back to (*)
- It is difficult to see how to avoid the scanning of H after iteration 1

142

Other Practical Problems

- Algorithm is biased towards long lines
 - ◆ The number of pixels on the intersection of a line and the image varies with ρ and θ
 - ◆ When we generalize this algorithm to detect other types of shapes, the bias will be introduced by the border of the image **clipping** the shapes for certain placements of the shapes in the image
- A Solution
 - ◆ Can precompute, for each (ρ, θ) , the number of pixels on the line $\rho = x \cos \theta + y \sin \theta$ and place these in a normalization array, η , which is exactly the same size as H
 - ◆ After the accumulator array is completed, we can divide each entry by the corresponding entry in η to obtain the percentage of pixels on the line that are 1 in B
 - ◆ Similar tricks can be developed to avoid scanning H



143

Asymptotic Complexity

- In the worst case, the Hough transform algorithm is an $O(n^3)$ algorithm, just like the brute-force algorithm
- Consider the following alternative approach
 - ◆ Generate all pairs of pixels in B that have value 1
 - ◆ these define the set of all line segments that will have counts > 1 after running the conventional Hough transform algorithm
 - ◆ For each pair, compute the parameters of the line joining that pair of points
 - ◆ not necessary to quantize the parameters for this version of the algorithm
 - ◆ Generate the set of pixels on this line and count the number of 1's in B in this set. This is the number of 1's in B that fall on this line
 - ◆ Generate a data structure of all such lines, sorted by count or normalized count. Can be easily used to solve problems (a) and (b)

144

Asymptotic Complexity

- What is the complexity of this algorithm?
 - ◆ Again, if there are $O(n)$ 1's in B, then we generate n^2 lines
 - ◆ Each of these has $O(n)$ points on it that have to be examined from B
 - ◆ So the algorithm is still $O(n^3)$
- Suppose that we **sample** the 1's in B and compute the lines joining only pairs from this sample
 - ◆ If our sample is small - say only the square root of the number of 1's in B, then we will be generating only $O(n)$ lines - one for each pair of points from a set of size $O(n^{1/2})$
 - ◆ Incredibly, it can be shown that with **very high probability** any such random sample of size $n^{1/2}$ will contain at least two of the points from any "long" line
 - ◆ This method reduces the asymptotic complexity to $O(n^2)$

145

Using More Image Information

- Practically, the 1's in B were computed by applying an edge detector to some grayscale image
- ◆ This means that we could also associate with each 1 in B the **gradient direction** measured at that edge point
 - ◆ this direction can be used to limit the range of θ considered at each 1 in B - for example, we might only generate lines for θ in the range $[\phi + \pi/4, \phi + 3\pi/4]$, where ϕ is the gradient direction at a pixel
 - ◆ this will further reduce the computational cost of the algorithm
- ◆ Each edge also has a **gradient magnitude**
 - ◆ could use this magnitude to differentially weight votes in the Hough transform algorithm
 - ◆ complicates peak finding
 - ◆ generally not a good idea - isolated high contrast edges can lead to unwanted peaks



146

Circle Detection

- Circle parameterized by
 - ◆ $(x_i - a)^2 + (y_i - b)^2 = r^2$
- If r known, 2D Hough space, $H(a, b)$, and an image point at coordinates (x_p, y_p) votes for a **circle** of points of radius r centered at (x_p, y_p) in H
- If r unknown, 3D Hough space, $H(a, b, r)$, and an image point at coordinates (x_p, y_p) votes for a **right circular cone** of points in H

147

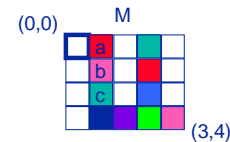
Generalized Hough Transform (GHT)

- Most of the comparisons performed during edge template matching match 0's in the image N against points in M
 - ◆ This is similar to the situation in the brute-force line finder, which generates lines containing mostly 0's in B
- The Generalized Hough transform avoids comparing the 0's in the image against the edge template
 - ◆ Similar to the Hough transform, the outermost loop of the algorithm will perform computations only when encountering a 1 in N
- Let $H(i, j)$ be an array of counters
 - ◆ Whenever we encounter a 1 in N we will efficiently determine all placements of M in N that would cause 1 in M to be aligned with this point of N. These placement will generate indices in H to be incremented

148

Template Representation for the Generalized Hough Transform

- Rather than represent M as a binary array, we will represent it as a list of coordinates, M'



M'

a (0, -1)
 b (-1, -1)
 c (-2, -1)
 (-3, -1)
 (-3, -2)
 (-3, -3)
 (-2, -3)
 (-1, -3)
 (0, -3)

• If we place pixel a over location (i, j) in N, then the (0, 0) location of the template will be at position (i, j-1)

• If we place pixel c over location (i, j) in N, then the (0, 0) location of the template will be at position (i-2, j-1)

149

GHT - Basic Algorithm

- Scan N until a 1 is encountered at position (x, y)
 - ◆ Iterate through each element (i, j) in M'
 - ◆ The placement of M over N that would have brought $M(i, j)$ over $N(x, y)$ is the one for which the origin of M is placed at position $(x+i, y+j)$
 - ◆ Therefore, we increment $H(x+i, y+j)$ by 1
 - ◆ And move on to the next element of M'
- And move on to the next 1 in N
- When the algorithm completes, $H(i, j)$ counts the number of template points that would overlay a "1" in N if the template were placed at position (i, j) in N

150

GHT - Generalizations

- Suppose we want to detect instances of M that vary in **orientation** in the image
 - ◆ need to increase the dimensionality of H by adding a dimension, θ , for orientation
 - ◆ Now, each time we encounter a "1" during the scan of N we must consider all possible rotations of M with respect to N - will result in incrementing one counter in each θ plane of H for each point in M
 - ◆ For each (i, j) from M
 - ◆ For each quantized θ
 - ◆ Determine the placement (r, s) of the rotated template in N that would bring (i, j) onto (x, y) and increment $H(r, s, \theta)$
- For **scale** we would have to add one more dimension to H and another loop that considers possible scales of M

151

Other Generalizations

- Match patterns of linear and curvilinear features against images from which such features have been detected
- Impose a hierarchical structure on M, and match pieces and compositions of pieces
 - ◆ At lowest level one finds possible matches to small pieces of M
 - ◆ A second GHT algorithm can now find combinations of pieces that satisfy other spatial constraints
 - ◆ Example: Square detection

152

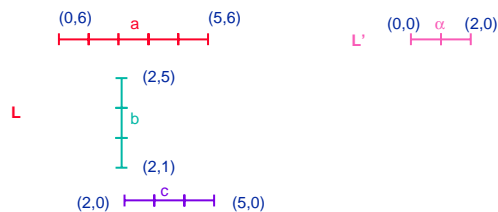
Hough Transform for Line Matching

- Let $L = \{L_1, \dots, L_n\}$ be the set of line segments which define M
- Let $L' = \{L'_1, \dots, L'_m\}$ be the set of observed line segments from N
- Define $L_i - L_j$ as follows:
 - ◆ If L_j is a **subsegment** of L_i , then $L_i - L_j = l_j$, where l_j is the length of L_j
 - ◆ Otherwise, $L_i - L_j = 0$
- Let F be a set of transformations that maps lines to lines
- Given F, L and L', find f in F that maximizes

$$v(f) = \sum_{L_i \in L} \sum_{L_j \in L'} [L_i - f(L_j)]$$

153

Example - Translation Only



- Which translations get incremented?
 - ◆ α -a: (0,6), (1,6), (2,6), (3,6) incremented by 2
 - ◆ α -b: none
 - ◆ α -c: (2,0), (2,1) incremented by 2

154

Representing High-Dimensional Hough Arrays

- Problems with high-dimensional arrays
 - ◆ Storage
 - ◆ Initialization and searching for high values after algorithm
- Possible solutions
 - ◆ Hierarchical representations
 - ◆ first match using a coarse-resolution Hough array
 - ◆ then selectively expand parts of the array having high matches
 - ◆ Projections
 - ◆ Instead of having one high-dimensional array, store a few 2D projections with common coordinates (e.g., store (x, y), (y, θ), (θ , s) and (s, x))
 - ◆ Find consistent peaks in these lower dimensional arrays

155

GHT Generate-and-Test

- Peaks in Hough array do not reflect spatial distribution of points underlying match
 - ◆ typical to “test” the quality of peak by explicitly matching template against image at the peak
 - ◆ hierarchical GHT’s also provide control over parts of template that match the image
- Controlling the generate-and-test framework
 - ◆ construct the complete Hough array, find peaks, and test them
 - ◆ test as soon as a point in the Hough space passes a threshold
 - ◆ if the match succeeds, points in I that matched can be eliminated from further testing
 - ◆ test as soon as a point in the Hough space is incremented **even once**

156

Chamfer Matching

- Given
 - ◆ Binary image, B, of edge and local feature locations
 - ◆ Binary “template” image, T, of shape we want to match
- Let D be an image in registration with B such that $D(i, j)$ is the distance to the nearest “1” in B
 - ◆ D is the **distance transform** of B
- Goal: Find placement of T in D that minimizes the sum, M, of the distance transform multiplied by the pixel values in T
 - ◆ If T is an exact match to B at location (i, j) then $M(i, j) = 0$
 - ◆ But if the edges in B are slightly displaced from their ideal locations in T, we still get a good match using the distance transform technique

157

Computing the Distance Transform

- Brute force, exact algorithm, is to scan B and find, for each “0”, its closest “1” using the Euclidean distance
 - ◆ expensive in time
- Various approximations to Euclidean distance can be made that are efficient to compute
- Goal: find a simple method to assign distance values to pixels that approximates ratios of Euclidean distances
 - ◆ horizontal and vertical neighbors in an image separated by distance 1
 - ◆ but diagonal neighbors separated by distance $\sqrt{2}$
 - ◆ This is “almost” a ratio of 3:4

158

Computing the Distance Transform

- Parallel algorithm
 - ◆ Initially, set $D(i, j) = 0$ where $B(i, j) = 1$, else set $D(i, j) = \infty$
 - ◆ Iterate the following until there are no further changes

$$D_k(i, j) = \min(D_{k-1}(i-1, j-1) + 4, D_{k-1}(i-1, j) + 4, \\ D_{k-1}(i+1, j-1) + 4, D_{k-1}(i+1, j) + 4, D_{k-1}(i, j-1) + 3, \\ D_{k-1}(i, j+1) + 3, D_{k-1}(i-1, j) + 3, D_{k-1}(i+1, j) + 3, D_{k-1}(i, j))$$

4	3	4
3	0	3
4	3	4

159

Computing the Distance Transform

- Two-pass sequential algorithm
- Same initial conditions
- Forward pass
 - ◆ $D(i,j) = \min[D(i-1,j-1) + 4, D(i-1,j) + 3, D(i-1,j+1) + 4, D(i,j-1) + 3, D(i,j)]$
- Backward pass
 - ◆ $D(i,j) = \min[D(i,j+1) + 3, D(i+1,j-1) + 4, D(i+1,j) + 3, D(i+1,j+1) + 4, D(i,j)]$

160

Hausdorff Distance Matching

- Let t be a transformation of the template T into the image
- $H(B, t(T)) = \max(h(B, t(T)), h(t(T), B))$, where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

- ◆ $\| \cdot \|$ is a norm like the Euclidean norm
- $h(A, B)$ is called the directed Hausdorff distance
 - ◆ ranks each point in A based on distance to nearest point in B
 - ◆ most mis-matched point of A is measure of match, i.e., measures distance of the point of A that is farthest from any point of B
 - ◆ if $h(A, B) = d$, then all points in A must be within distance d of B
 - ◆ generally, $h(A, B) \circ h(B, A)$
 - ◆ easy to compute Hausdorff distances from Distance Transform

161

Computing the Hausdorff Distance

$$\begin{aligned} H(A, B) &= \max(h(A, B), h(B, A)) \\ &= \max(\max_{a \in A} \min_{b \in B} \|a - b\|, \max_{b \in B} \min_{a \in A} \|a - b\|) \\ &= \max(\max_{a \in A} d(a), \max_{b \in B} d'(b)) \end{aligned}$$

- where $d(x) = \min_{b \in B} \|x - b\| = \text{DistanceTransform}(B)$
- and $d'(x) = \min_{a \in A} \|a - x\| = \text{DistanceTransform}(A)$
- For translation only, $H(A, B+t) = \text{maximum of translated copies of } d(x) \text{ and } d'(x)$
- $O(pq(p+q) \log pq)$ time, where $|A|=p, |B|=q$

162

Fast Template Matching

- **Simulated Annealing** approach
 - ◆ Let $T_{\theta,s}$ be a rotated and scaled version of T
 - ◆ For a random θ and s , and a random (i, j) match $T_{\theta,s}$ at position (i, j) of I
 - ◆ Now, randomly perturb θ, s, i and j by perturbations whose magnitudes will be reduced in subsequent iterations of the algorithm to obtain θ', s', i', j'
 - ◆ Match $T_{\theta',s'}$ at position (i', j') . If the match is better, “move” to that position in the search space. If the match is worse, move with some probability to that position **anyway!**
 - ◆ Iterate using smaller perturbations, and smaller probabilities of moving to worse locations
 - ◆ the rate at which the probability of taking “bad” moves decreases is called the “cooling schedule” of the process
 - ◆ This has also been demonstrated with deformation parameters that mimic projection effects for planar patterns

163