Robot Sensing:
Feature Detection

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 8

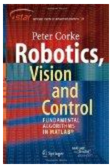
metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

September 14, 2016

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (*& Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	**14-Sep**	**Robot Sensing: Multiple View Geometry & Feature Detection**
9	21-Sep	Probabilistic Robotics: Localization & SLAM
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Planning & Control
12	19-Oct	State-Space Modelling
13	26-Oct	Shaping the Dynamic Response/LQR + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

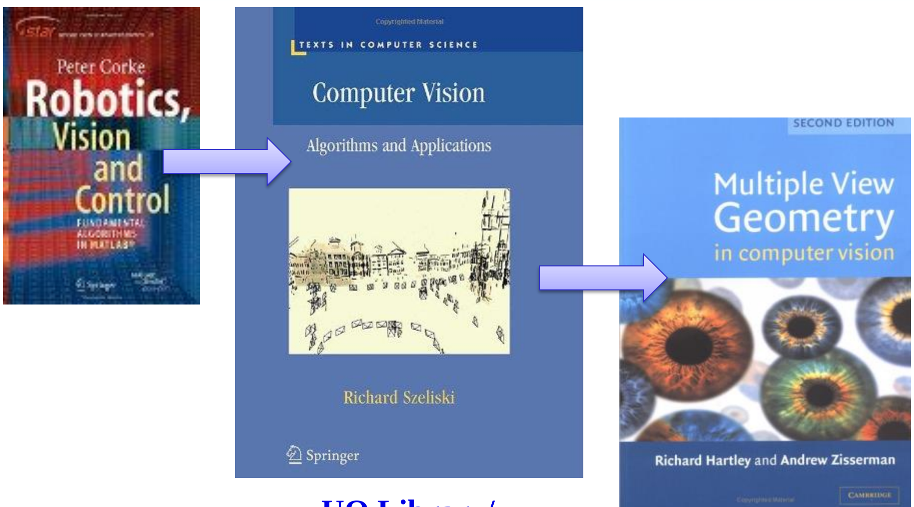
➔ **Sensing and Vision** 🡐

- Multiple View Geometry
  - P. 47
  - Hartley & Zisserman:
    Chapter 6: Camera Models
    Chapter 7: Camera Matrix

- Localization
  - Chapter 6: Localization

Next Time

Reference Material



UQ Library/ SpringerLink

UQ Library
(ePDF)

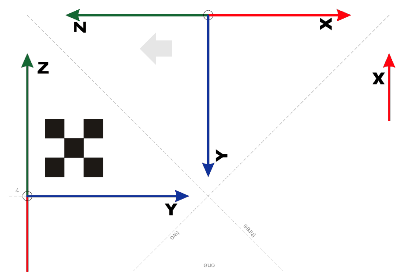
Announcement: Monday Lab → Demo Thur | Fri

- Monday, October 3:
  - Queens Birthday Public Holiday
- "Makeup" Lab
  on Friday, October 7
  from 4-6pm
- Monday Prac students
  may demo on Thursday (Oct 6)
  or Friday (Oct 7)
- Thursday Prac students to demo
  on Thursday (Oct 6)

SIFT / Corners for the {Frame} finder

To find the Frame, Consider:

- Structure
  - Corners
  - SIFT
  - ???
- Calibration Sequence
- Thought Experiment:
  How do you make this
  traceable back to the
  {camera frame}



Camera matrix calibration

- Advantages:
  - very simple to formulate and solve
  - can recover K [R | t] from M using QR decomposition [Golub & VanLoan 96]

- Disadvantages:
  - doesn't compute internal parameters
  - more unknowns than true degrees of freedom
  - need a separate camera matrix for each new view

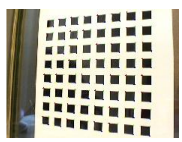
From Szeliski, *Computer Vision: Algorithms and Applications*

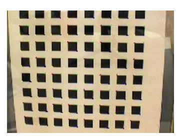
Multi-plane calibration

- Use several images of planar target held at unknown orientations [Zhang 99]
  - Compute plane homographies

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim \mathbf{HX}$$

  - Solve for K-TK-1 from Hk's
    - 1plane if only f unknown
    - 2 planes if (f,uc,vc) unknown
    - 3+ planes for full K
  - Code available from Zhang and OpenCV

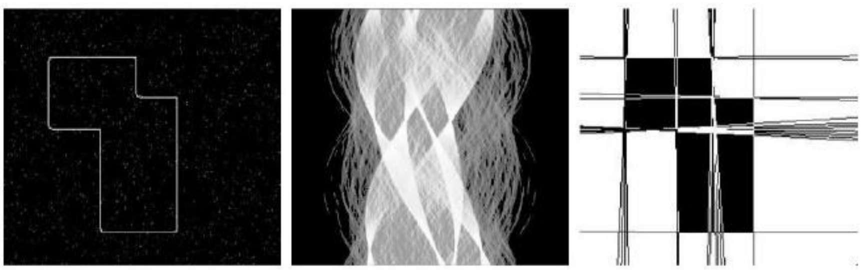




From Szeliski, *Computer Vision: Algorithms and Applications*

Lines (Recap)

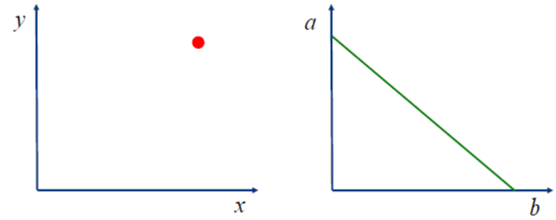
Hough Transform



- Uses a voting mechanism
- Can be used for other lines and shapes (not just straight lines)

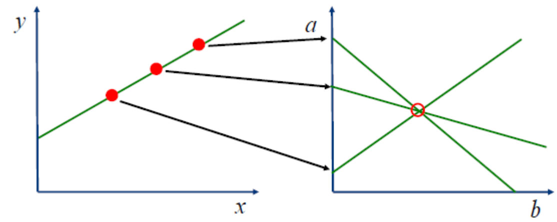
Hough Transform: Voting Space

$y = ax + b$ $\qquad$ $a = -\frac{1}{x}b + \frac{y}{x}$



- Count the number of lines that can go through a point and move it from the "*x-y*" plane to the "a-b" plane
- There is only a one-"infinite" number (a line!) of solutions (not a two-"infinite" set – a plane)

Hough Transform: Voting Space



- In practice, the polar form is often used

$$a = x\cos a + y\sin b$$

- This avoids problems with lines that are nearly vertical

Hough Transform: Algorithm

1. Quantize the parameter space appropriately.

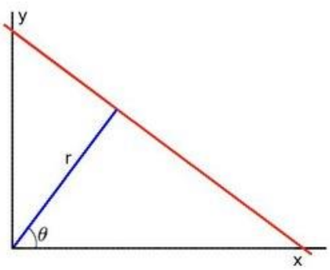
2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

3. For each point (x,y) in the (visual & range) image space, increment by 1 each of the accumulators that satisfy the equation.

4. Maxima in the accumulator array correspond to the parameters of model instances.

Line Detection – Hough Lines [1]

- A line in an image can be expressed as two variables:
  - **Cartesian coordinate system:** m,b
  - **Polar coordinate system:** r, θ
    ➔ **<u>avoids problems with vert. lines</u>**



y=**m**x+**b** ➔

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

- For each point (x_1, y_1) we can write:

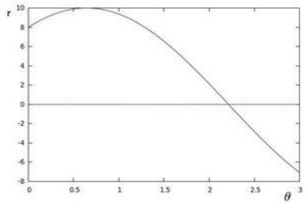
$$r = x_1 \cos\theta + y_1 \sin\theta$$

- Each pair (r,θ) represents a line that passes through (x_1, y_1)

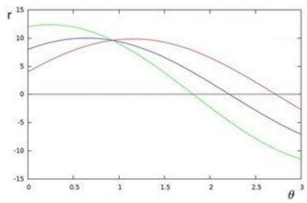
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



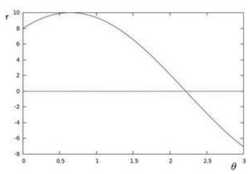
- Repeating for all points on the image



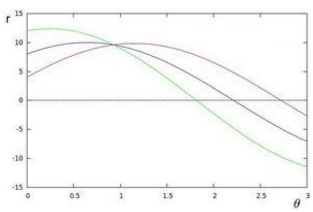
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [3]

- Thus a given point gives a sinusoid



- Repeating for all points on the image

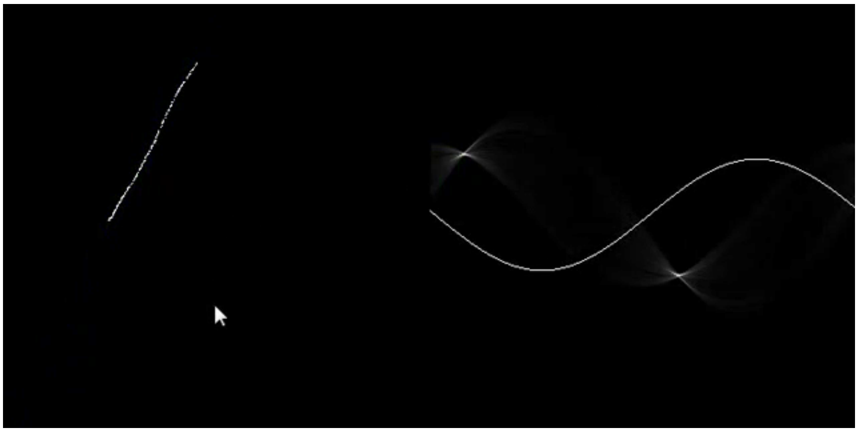


- NOTE that an intersection of sinusoids represents (**a point**) represents **a line** in which pixel points lay.

➔ Thus, a line can be *detected* by finding the number of Intersections between curves

See also OpenCV documentation (cv::HoughLines)

“Cool Robotics Share” -- Hough Transform



- http://www.activovision.com/octavi/doku.php?id=hough_transform

RANdom SAmple Consensus

1. Repeatedly select a small (minimal) subset of correspondences
2. Estimate a solution (in this case a the line)
3. Count the number of “inliers”, $|e|<\Theta$
   (for LMS, estimate med($|e|$)
4. Pick the *best* subset of inliers
5. Find a complete least-squares solution

- Related to least median squares
- See also:
  MAPSAC (Maximum *A Posteriori* SAmple Consensus)

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature Detection

"A Rose By Any Other Name?

3 8 1 7 6 7 4 7 8 3 5 9 5 3 6 3 7 4 4 6 9 3 8 7 9 0 3 6 3 2 6 6 5 6 0 3 4 2 6 8 3 8 1…
7 6 7 4 7 8 3 5 9 5 3 6 3 7 4 4 6 9 3 8 7 9 0 3 6 3 2 6 6 5 6 0 3 4 2 6 8

– SIFT

How to get Matching Points? **Features**

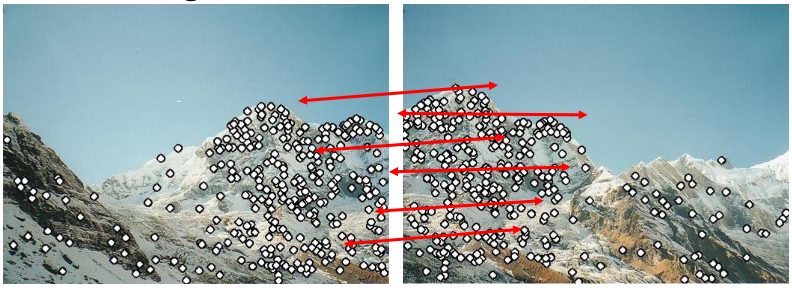
- ~~Colour~~
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF, ORB…

In OpenCV: The following detector types are supported:

- "FAST" – FastFeatureDetector
- "STAR" – StarFeatureDetector
- "SIFT" – SIFT (nonfree module)
- "SURF" – SURF (nonfree module)
- "ORB" – ORB
- "BRISK" – BRISK
- "MSER" – MSER
- "GFTT" – GoodFeaturesToTrackDetector
- "HARRIS" – GoodFeaturesToTrackDetector with Harris detector enabled
- "Dense" – DenseFeatureDetector
- "SimpleBlob" – SimpleBlobDetector

Why extract features?

- **Object detection**
- Robot Navigation
- Scene Recognition



- Steps:
  - Extract Features
  - Match Features

Adopted drom S. Lazebnik, Gang Hua (CS 558)

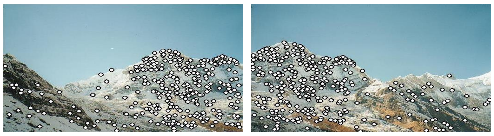
Why extract features? [2]

- Panorama stitching…
  - → Step 3: Align images

Adopted from S. Lazebnik, Gang Hua (CS 558)

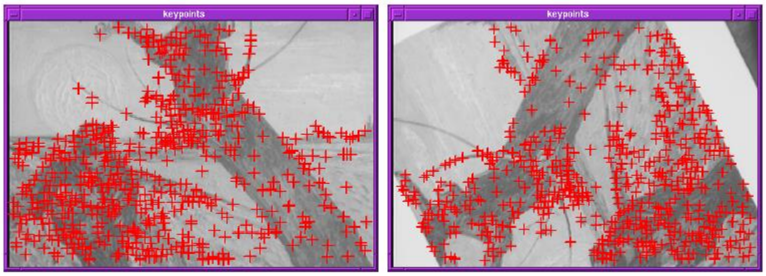
Characteristics of good features

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion



Adopted from S. Lazebnik, Gang Hua (CS 558)

Finding Corners



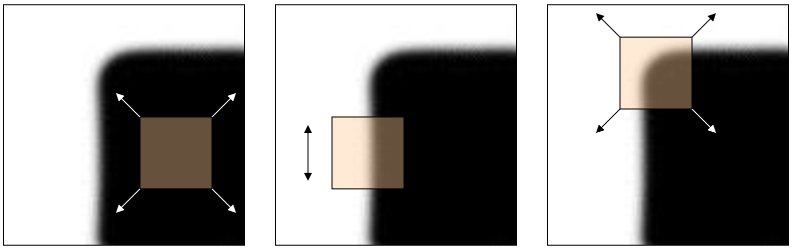
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

C.Harris and M.Stephens. "A Combined Corner and Edge Detector.“ *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Adopted from S. Lazebnik, Gang Hua (CS 558)

Corner Detection: Basic Idea

- Look through a window
- Shifting a window in any direction should give a large change in intensity



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

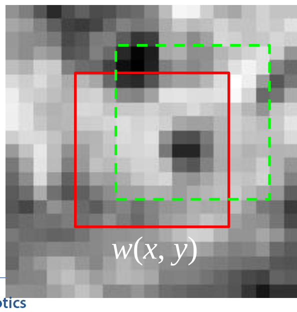
Source: A. Efros

Corner Detection: Mathematics

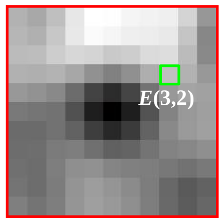
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$



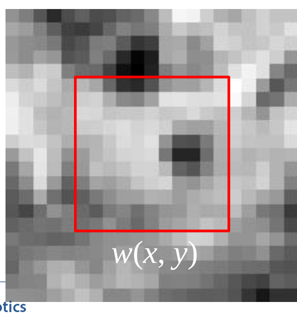
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

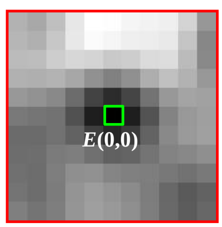
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$



Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

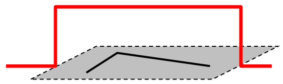
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

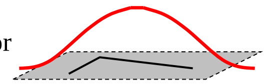
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



1 in window, 0 outside

Gaussian

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Source: R. Szeliski

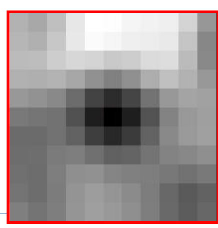
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

We want to find out how this function behaves for small shifts

$E(u, v)$



Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

We want to find out how this function behaves for small shifts

$$E(u,v)\approx E(0,0)+[u\ \ v]\begin{bmatrix}E_u(0,0)\\E_v(0,0)\end{bmatrix}+\frac{1}{2}[u\ \ v]\begin{bmatrix}E_{uu}(0,0) & E_{uv}(0,0)\\E_{uv}(0,0) & E_{vv}(0,0)\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}$$

Local quadratic approximation of $E(u,v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v)\approx E(0,0)+[u\ \ v]\begin{bmatrix}E_u(0,0)\\E_v(0,0)\end{bmatrix}+\frac{1}{2}[u\ \ v]\begin{bmatrix}E_{uu}(0,0) & E_{uv}(0,0)\\E_{uv}(0,0) & E_{vv}(0,0)\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}$$

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

$$E_{uu}(u,v)=\sum_{x,y} 2w(x,y)I_x(x+u,y+v)I_x(x+u,y+v)$$
$$+\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v)=\sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_x(x+u,y+v)$$
$$+\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_{xy}(x+u,y+v)$$

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

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

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris detector: Steps

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix M in a Gaussian window around each pixel
- Compute corner response function R
- Threshold R
- Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

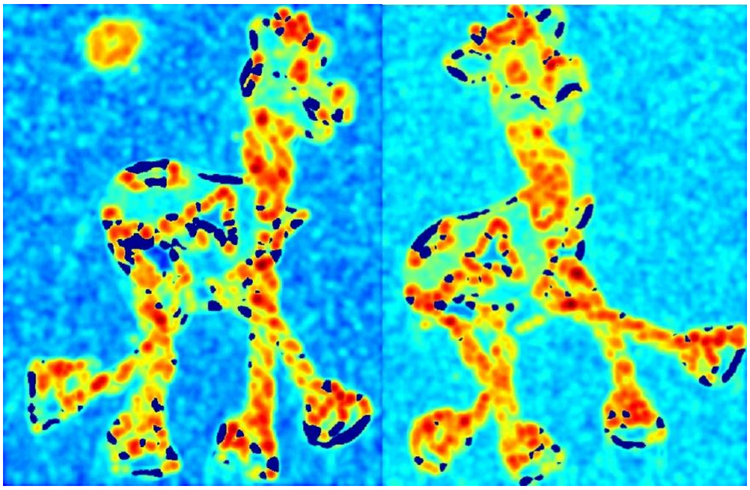
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris Detector: Steps

Adopted from S. Lazebnik, Gang Hua ([CS 558](CS 558))

Harris Detector: Steps

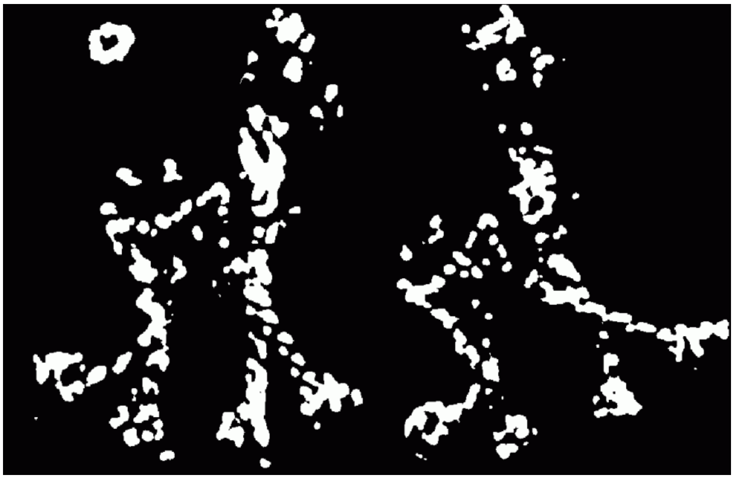
Compute corner response *R*



Adopted from S. Lazebnik, Gang Hua ([CS 558](CS 558))

Harris Detector: Steps

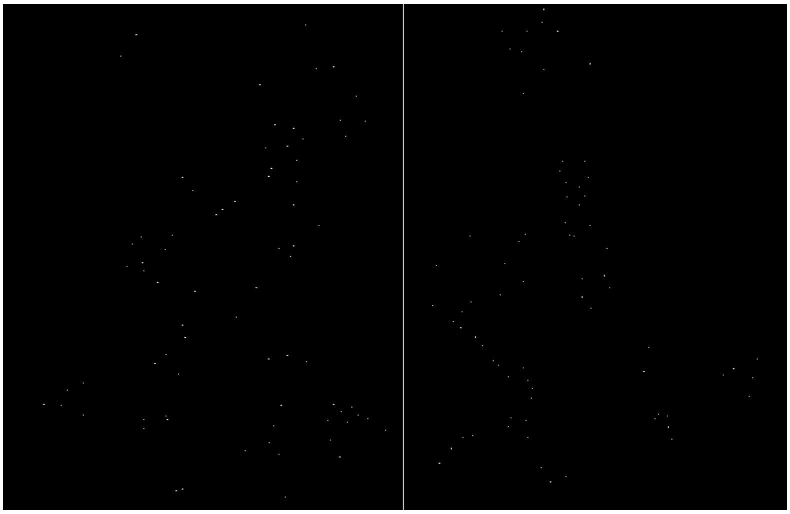
Find points with large corner response: R>threshold



Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

Take only the points of local maxima of R



Adopted from S. Lazebnik, Gang Hua (CS 558)

Harris Detector: Steps

Adopted from S. Lazebnik, Gang Hua (CS 558)

Invariance and covariance

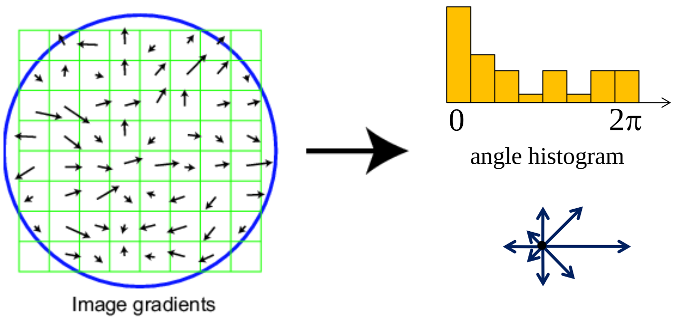
- We want corner locations to be invariant to photometric transformations and covariant to geometric transformations
  - Invariance: image is transformed and corner locations do not change
  - Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations

Adopted from S. Lazebnik, Gang Hua (CS 558)

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

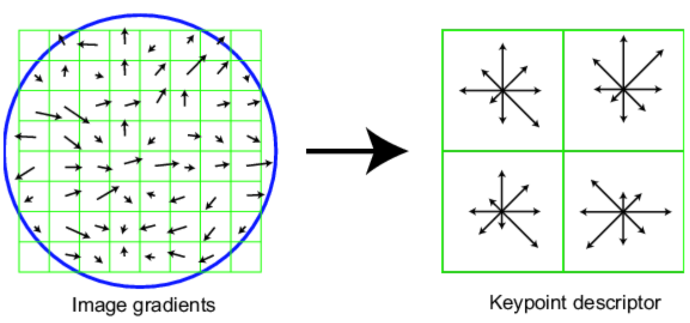


Adapted from slide by David Lowe

SIFT descriptor

Full version

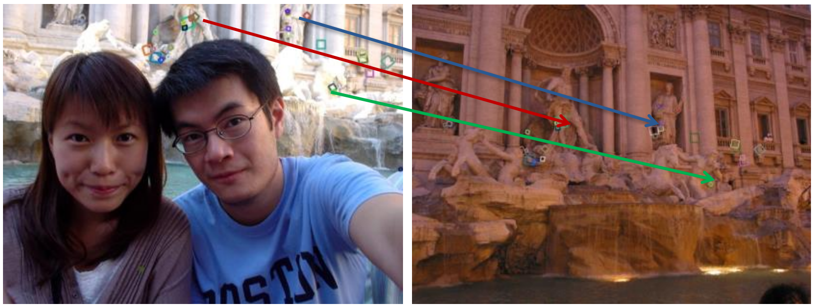
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

Properties of SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available
    - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



From David Lowe and Szeliski, *Computer Vision: Algorithms and Applications*

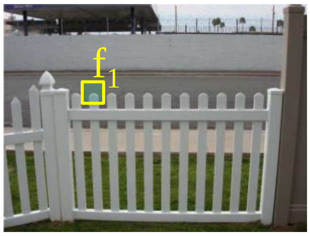
Feature matching

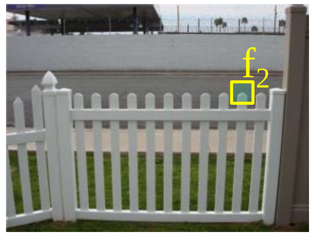
- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with min distance

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

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



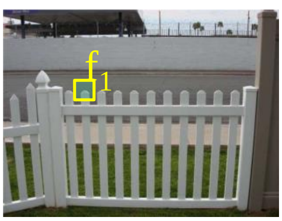


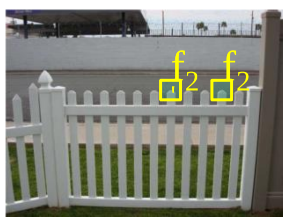
I_1 I_2

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

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



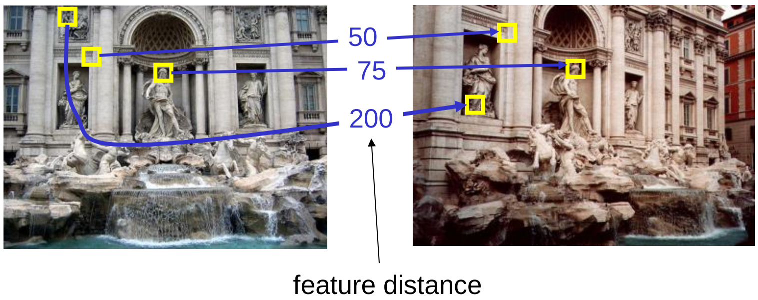


I_1 I_2

From Szeliski, *Computer Vision: Algorithms and Applications*

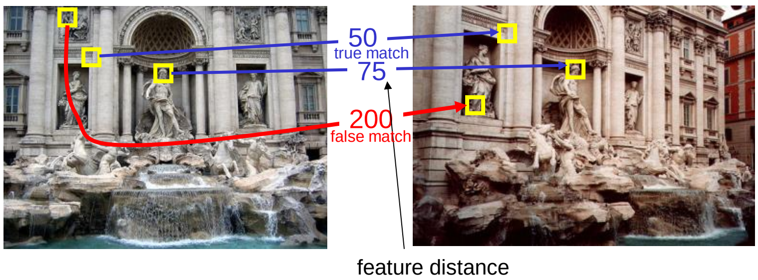
Evaluating the results

- How can we measure the performance of a feature matcher?



From Szeliski, *Computer Vision: Algorithms and Applications*

True/false positives



- The distance threshold affects performance
  - True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  - False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

- Iterative non-linear least squares [Press'92]
  - Linearize measurement equations

$$\begin{aligned}\hat{u}_i &= f(\mathbf{m}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m} \\ \hat{v}_i &= g(\mathbf{m}, \mathbf{x}_i) + \frac{\partial g}{\partial \mathbf{m}} \Delta \mathbf{m}\end{aligned}$$

  - Substitute into log-likelihood equation: quadratic cost function in Dm

$$\sum_i \sigma_i^{-2} (\hat{u}_i - u_i + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m})^2 + \cdots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

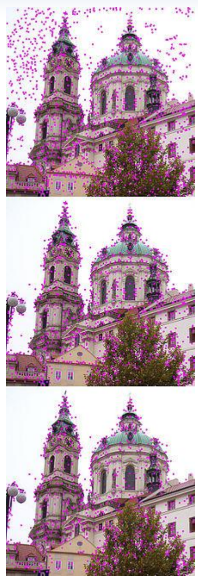
- What if it doesn't converge?
  - Multiply diagonal by (1 + l), increase l until it does
  - Halve the step size Dm (my favorite)
  - Use line search
  - Other ideas?
- Uncertainty analysis: covariance S = A-1
- Is maximum likelihood the best idea?
- How to start in vicinity of global minimum?

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature Based Vision Extras

Scale Invariant Feature Transforms

- Goal was to define an algorithm to describe an image with features
- This would enable a number of different applications:
  - Feature Matching
  - Object / Image Matching
  - Orientation / Homography Resolution



Wikipedia: Scale Invariant Feature Transforms (2014)

SIFT: Feature Definition

- SIFT features are defined as the local extrema in a Difference of Gaussian (D) Scale Pyramid.

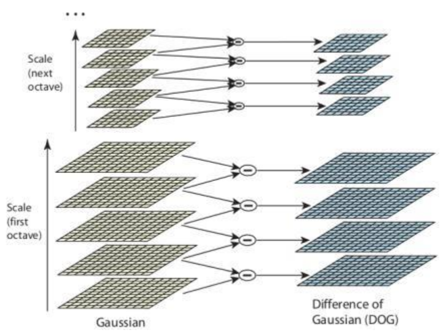
$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_i\sigma)$$

Where

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

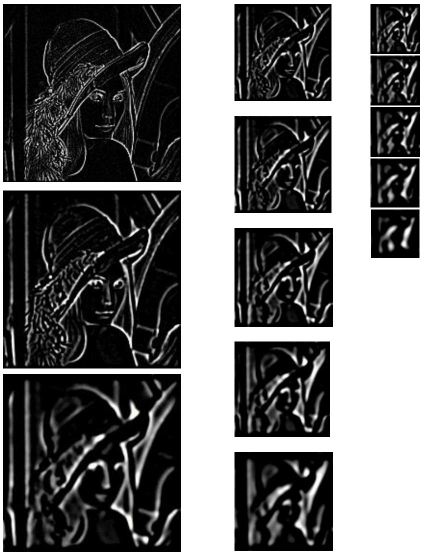
SIFT: Scale Pyramid

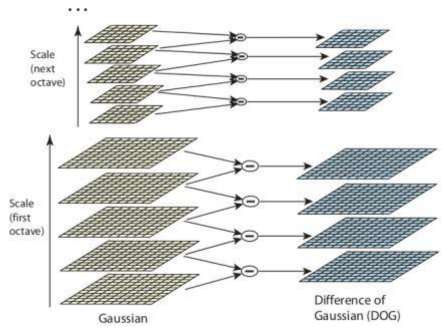
- D images are organised into a pyramid of progressively blurred images.
- Separated into octaves and scale levels per octave.
- Between octaves image is decimated by a factor of 2.



Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

SIFT: Scale Pyramid

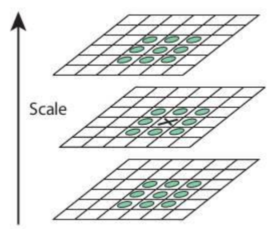




Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

SIFT: Feature Detection

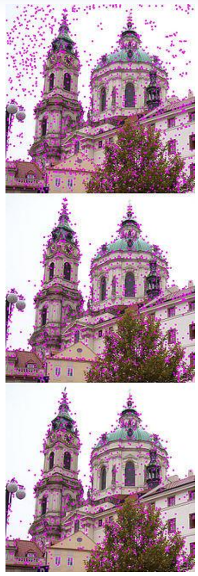
- Each scale level in the image is evaluated for features.
- A feature is defined as a local maximum or minimum.
- For efficiency the 26 surrounding points are evaluated.



Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

SIFT: Feature Reduction

- Initial feature detection over detects features descriptive of the image.
- Initially remove features with low contrast.
- Then evaluate features to remove any edge responses.

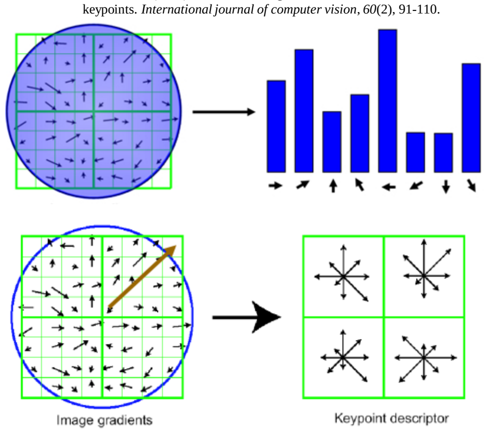


Wikipedia: Scale Invariant Feature Transforms (2014)

SIFT: Feature Description

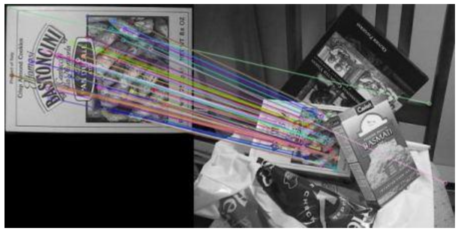
- Features are described using the pixel gradients in a 16x16 square centring on the feature point.
- These gradients are then segmented into 4x4 boxes. An 8 bin orientation histogram is created to define the box.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.



SIFT: Feature Matching

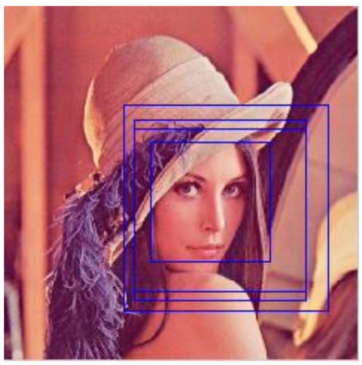
- A match is defined as a pair of features with the closest Euclidian distance to each other.
- Matches above a threshold are culled to improve match.



OpenCV: Feature Matching (2014)

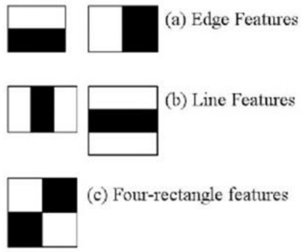
Boosted Cascade Haar-like Weak Classifiers

- Fast object detector designed primarily for use in face detection.
- Uses a cascade of weak classifiers to define object match.



Viola Jones: Feature Definition

- Feature is classified as being the difference between the average intensity of two or more image sections.
- Can be any arithmetic combination of section values.



Viola Jones: Efficient Calculation of Features

- Fast calculation of the feature value is obtained by calculating the integral image.
- This leaves at most 4 sum operations to calculate a feature.

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

input image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

integral image

Viola Jones: Boosting

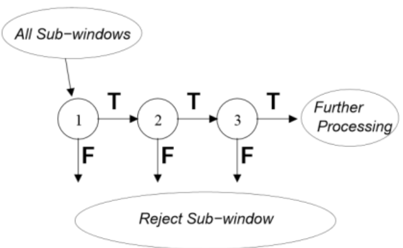
- Iteratively selects best classifier for detection.
- Assigns weights to each classifier to indicate likelihood of classifier indicating positive detection
- If the sum of the weights of positive classifier responses is above a threshold then there is a positive detection.

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \dots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier , h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features

Viola Jones: Boosted Cascades

- Effective boosted classifiers require a high number of weak classifiers.
- However, simple low count classifiers offer high rejection rate.
- Solution is to use cascaded classifiers.



Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features