Computer Vision 1

THOMAS MENSINK

MASTER AI

UNIVERSITY OF AMSTERDAM

LECTURES/THEORY

- 06-02-2018, 17:00-19:00, C0.05, **Introduction** (*Szeliski 1*)
- 13-02-2018, 17:00-19:00, C0.05, **Image Formation** (*Szeliski: 2.1.1 + 2.1.2 + 2.2 + 2.3.2 + 2.3.3*)
- 20-02-2018, 17:00-19:00, C0.05, **Color and Image Processing** (*Szeliski: 3.1 + 3.2 + 3.3*)
- 27-02-2018, 17:00-19:00, C0.05, **Feature Detection, Motion and Classification** (*Szeliski: 4, 8.1.1 + 8.1.3 + 8.2.1 + 8.4; Bengio: 4 + 5.1 + 5.2 + 5.3 + 5.7 + 5.8 + 5.9*)
- 06-03-2018, 17:00-19:00, C0.05, **Object Recognition: BoW and ConvNets** (*Szeliski: 5.1.1 + 5.1.4 + 5.1.5 + 5.2 + 5.3 + 5.4, 6.1 + 6.3, 14.1 + 14.2.1 + 14.3 + 14.4.1; Bengio: 7.2 + 7.4 + 9.1 + 9.2 + 9.3*)
- 13-03-2018, 17:00-19:00, C0.05, **Deep Learning, Stereo and 3D Reconstruction** (*Szeliski: 11.1 + 11.2 + 11.3 + 11.4, 12.1 + 12.2; Bengio: 12.1 + 12.2*)
- 20-03-2018, 17:00-19:00, C0.05, **Applications** (Szeliski: 12.6.2 + 12.6.3 + 12.2.4)
- 26-03-2018, Monday, 9:00-12:00, **Written Exam**

TODAY: FEATURE DETECTION, MOTION AND CLASSIFICATION

Focus

Szeliski:

- 4.1 Point & Patches
- 4.2 Edges
- 4.3 Lines
- 4.4 Additional reading / exercises
- 8.1 Translational alignment (-Fourier)
- 8.2.1 Parametric motion: Application Video Stabilisation
- 8.4 Optical Flow

If we need it …

Bengio:

- 4 Numerical Computation
- 5 Machine Learning Basics

Note: Should've been covered in ML1

LAST WEEK

Class

- Color invariants
- Image processing
- Filtering and convolutions

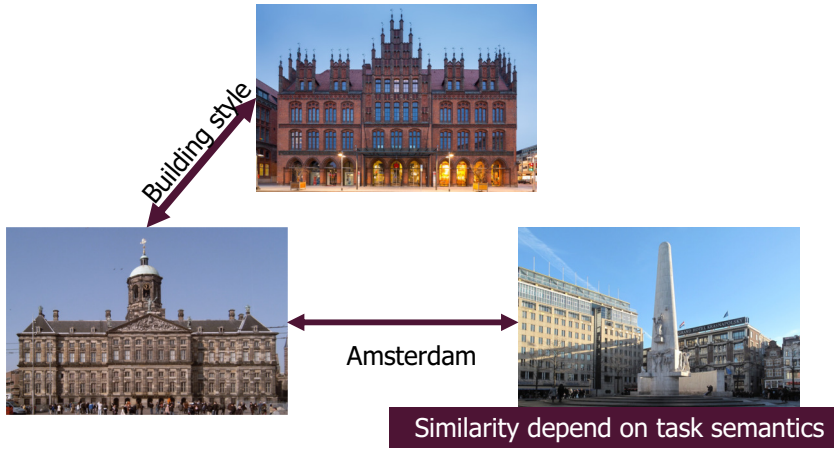
Lab

- Matlab assignment
- Piazza

TODAY'S CLASS

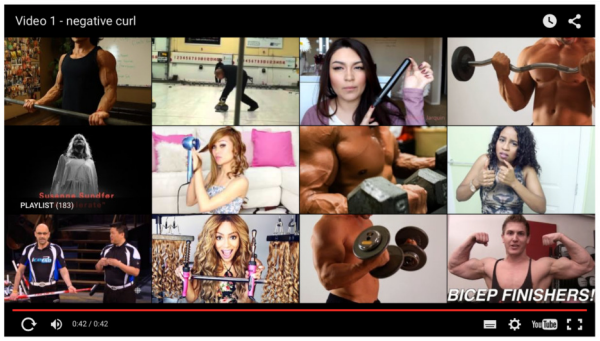
- Image Similarity
- Feature Detection
- Motion

IMAGE SIMILARITY



Similarity depend on task semantics

VISUAL SIMILARITY EXAMPLE

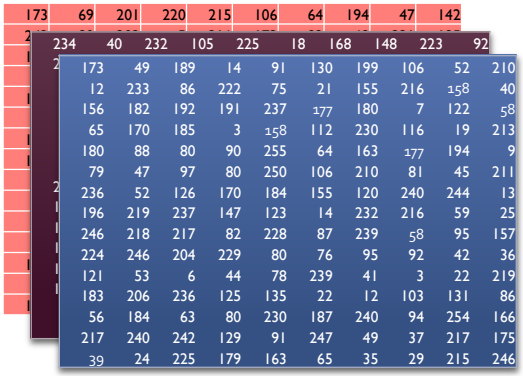


Source: NOS.nl | 21st Oct 2015

HOW TO REPRESENT AN IMAGE?

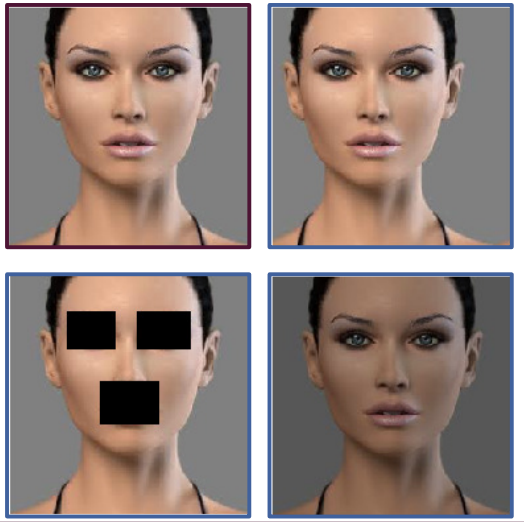
PIXEL VALUES

Rescale Image
64 x 64



Concatenate to 12.888 dim vector

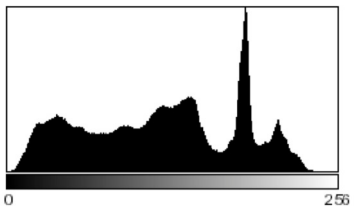
PIXEL VALUES - SIMILARITIES



These three images have equal distance to (green) query image

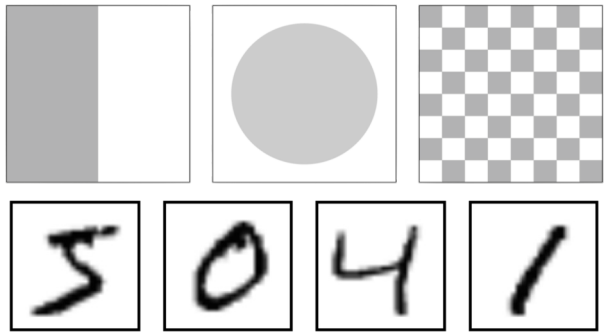
Raw pixel values do not reflect visual similarity

GLOBAL HISTOGRAMS

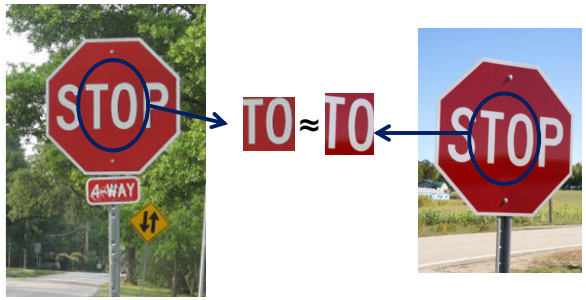


Count: 1920000 Min: 0
Mean: 118.848 Max: 251
StdDev: 59.179 Mode: 184 (30513)

GLOBAL HISTOGRAMS: EXAMPLES



LOCAL CORRESPONDENCE



Features are local, so robust to occlusion and clutter

TODAY'S CLASS

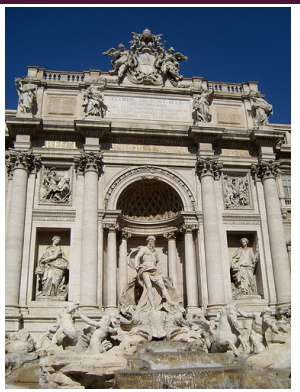
- Image Similarity
- **Feature Detection**
- Motion

FEATURE DETECTION

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Indexing and database retrieval
- Object recognition

IMAGE MATCHING

by Diva Sian



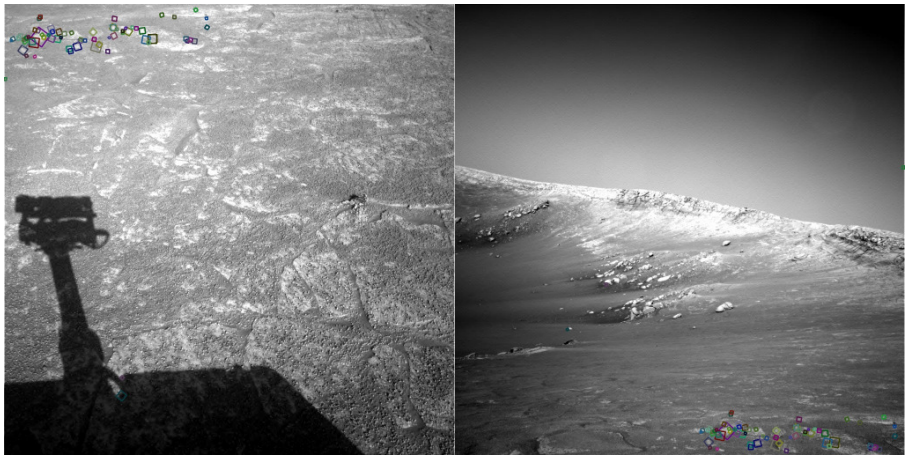
by swashford

IMAGE MATCHING

by Diva Sian

by scgbt

EVEN HARDER, STILL POSSIBLE?



NASA Mars Rover images

SIFT matches by Noah Snavely

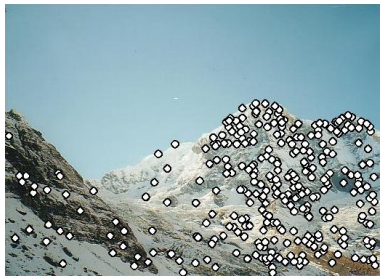
Local features and alignment

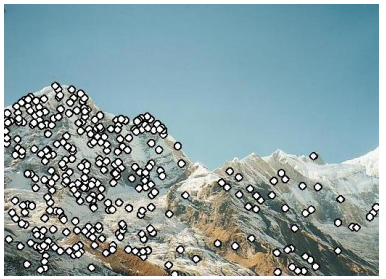
- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax effects. So look for local features that match well.
- How would you do it by eye?

[Darya Frolova and Denis Simakov]

Local features and alignment

- Detect feature points in both images

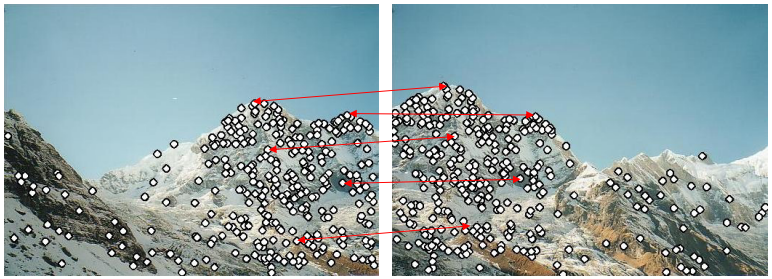




[Darya Frolova and Denis Simakov]

Local features and alignment

- Detect feature points in both images
- Find corresponding pairs



[Darya Frolova and Denis Simakov]

Local features and alignment

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

[Darya Frolova and Denis Simakov]

Local features and alignment

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

no chance to match!

We need a **repeatable** detector

[Darya Frolova and Denis Simakov]

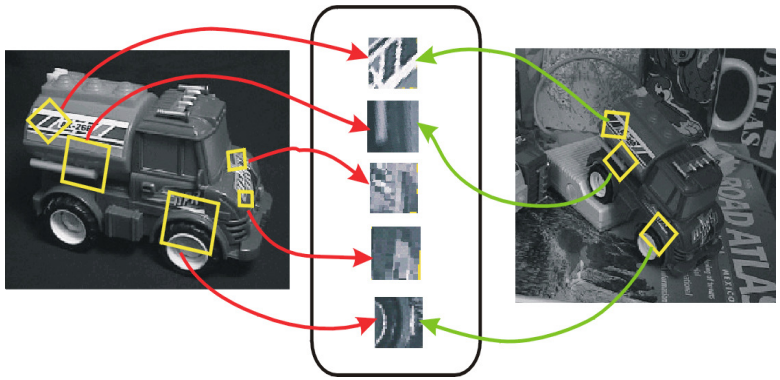
Local features and alignment

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

We need a **reliable** descriptor

[Darya Frolova and Denis Simakov]

INVARIANT LOCAL DESCRIPTORS



Local descriptors should transform image content to feature space: invariant to translation, rotation, scale, and other imaging parameters

Local features and alignment

- Problem 3:
  - Detect distinctive points in both images

no chance to find matching pairs!

We need a **distinctive** detector/descriptor

[Darya Frolova and Denis Simakov]

MAIN QUESTIONS

1. Where will the interest points come from?
2. How to *describe* a local region?
3. How to establish *correspondences*, i.e., compute matches?

Note: *interest points* are *key-points* (and sometimes called *features*)

Many applications

- tracking: which points are good to track?
- recognition: find patches likely to tell us something about object category
- 3D reconstruction: find correspondences across different views

EXAMPLE: STRUCTURE FROM MOTION

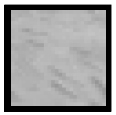
https://www.youtube.com/watch?v=4cEQZreQ2zQ

EXAMPLE: STRUCTURE FROM MOTION

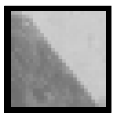
https://www.youtube.com/watch?v=4cEQZreQ2zQ

Where will interest points come from?

INTEREST POINTS



0D structure: single points
➡ not useful for matching



1D structure: lines
➡ edge, can be localised in 1D, subject to the aperture problem

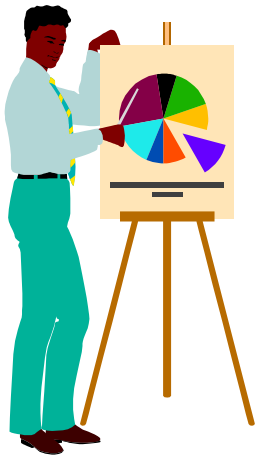
2D structure: corners, blobs
➡ corner, or **interest point**, can be localised in 2D, good for matching

Interest Points have **2D** structure.

INTEREST POINT DETECTORS

1. Edges
2. Corners
3. Blobs
4. Templates

Canny Edge Detector



DESIGNING AN EDGE DETECTOR

Criteria for a good edge detector:

- Good detection
  - the optimal detector should find all real edges, ignoring noise or other artifacts
- Good localization
  - the edges detected must be as close as possible to the true edges
  - the detector must return one point only for each true edge point

Cues of edge detection

- Differences in color, intensity, or texture across the boundary
- Continuity and closure
- High-level knowledge

Source: L. Fei-Fei

CANNY EDGE DETECTOR

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

PIPELINE

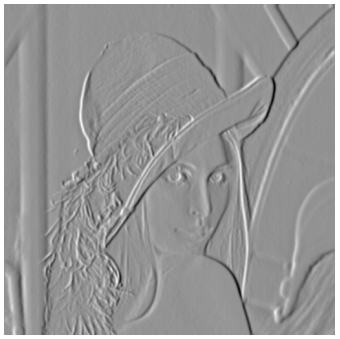
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: edge(image, ‘canny’)

Source: D. Lowe, L. Fei-Fei

FINDING CANNY EDGES

COMPUTE GRADIENTS

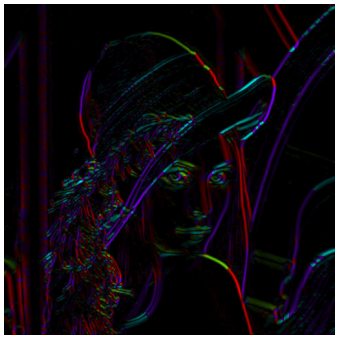


X-Derivative of Gaussian

Y-Derivative of Gaussian

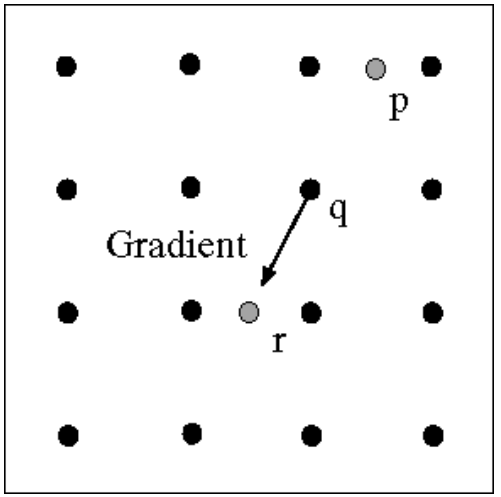
GET MAGNITUDE AND ORIENTATION

Gradient Magnitude

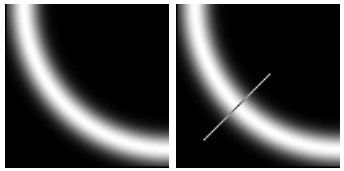


theta = atan2(gy, gx)

NON-MAXIMUM SUPPRESSION



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



NON-MAXIMUM SUPPRESSION

Before

After

THRESHOLDING AND LINKING

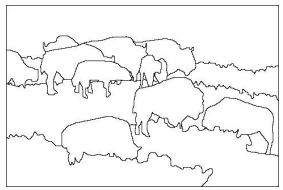
Before

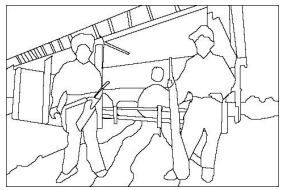
After

FINAL CANNY EDGES

WHERE DO HUMANS SEE BOUNDARIES?

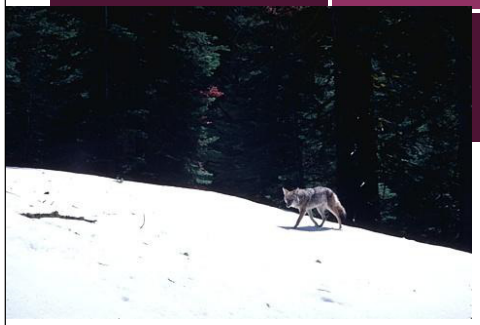
image | human segmentation | gradient magnitude





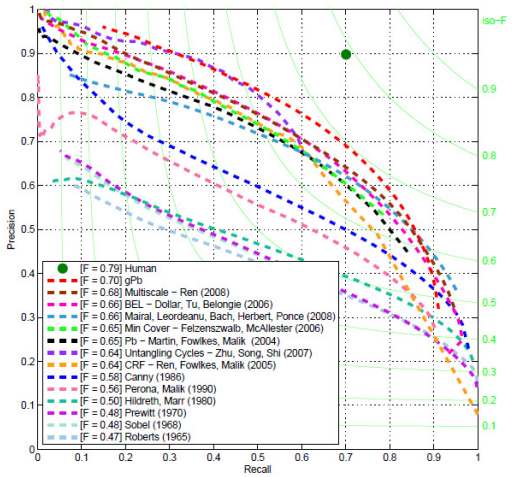
- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

RESULTS

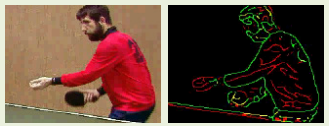


RESULTS

50 YEARS OF EDGE DETECTION



Detection and Classification of Color Edges



Edge Detection

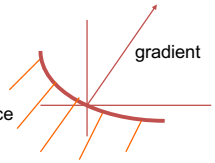
An intensity edge is defined as a point where...

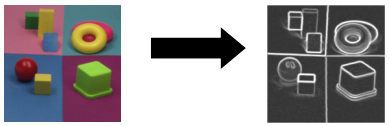
the gradient is large:

$$|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$$

Localization - Laplacian - zero-crossing:

$$\nabla^2 f(x,y) = f_{xx} + f_{yy}$$





gradient computation

zero-crossings at large gradients

Color Edge Detection

Summing the gradient magnitudes separately:

$$|\nabla C(x,y)| = \sqrt{(R_x^2 + R_y^2)} + \sqrt{(G_x^2 + G_y^2)} + \sqrt{(B_x^2 + B_y^2)}$$

or using the Euclidean metric:

$$|\nabla C(x,y)| = \sqrt{R_x^2 + R_y^2 + G_x^2 + G_y^2 + B_x^2 + B_y^2}$$

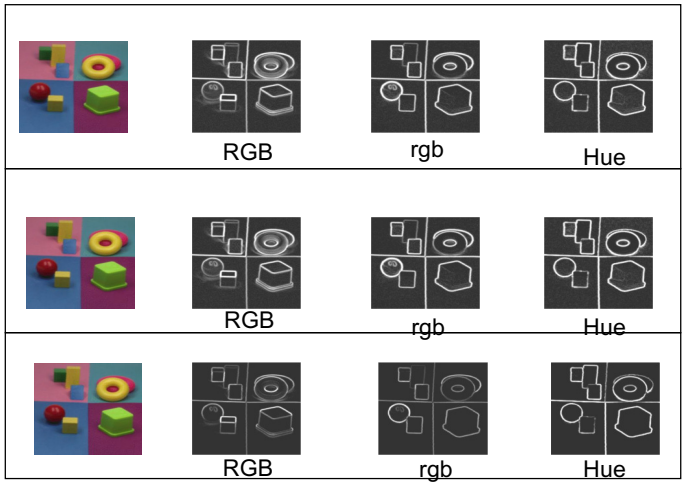
or using eigen-values: [diZenzo86], [Sapiro96]

Edge Classification

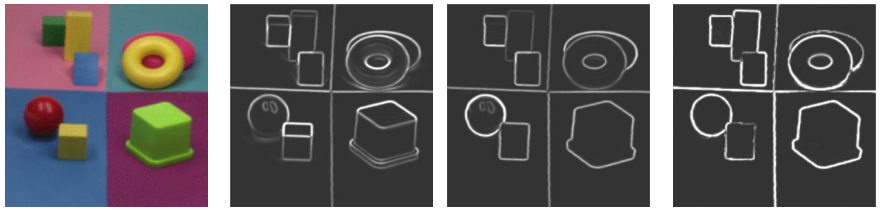
summing gradient magnitudes separately:

Euclidean:

eigen-values:



Edge Classification



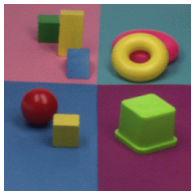
if $(|\nabla C_{rgb}| \geq t_{rgb} \,\&\, |\nabla C_H| < t_H)$ then classify as highlight edge

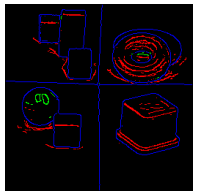
else

if $(|\nabla C_H| \geq t_H)$ then classify as color edge

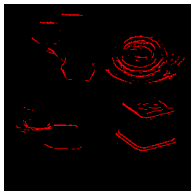
else classify as shadow/geometry edge

Edge Classification

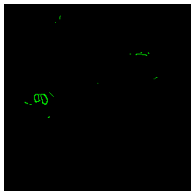




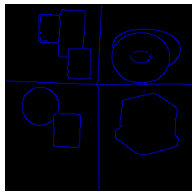
colour edge maxima by type



shadows and geometry



highlights

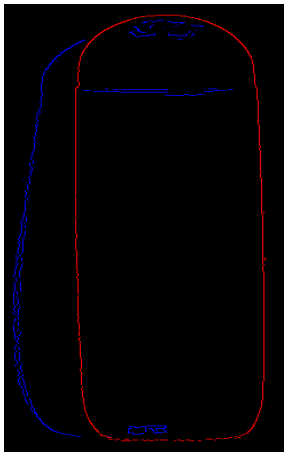


colour edges

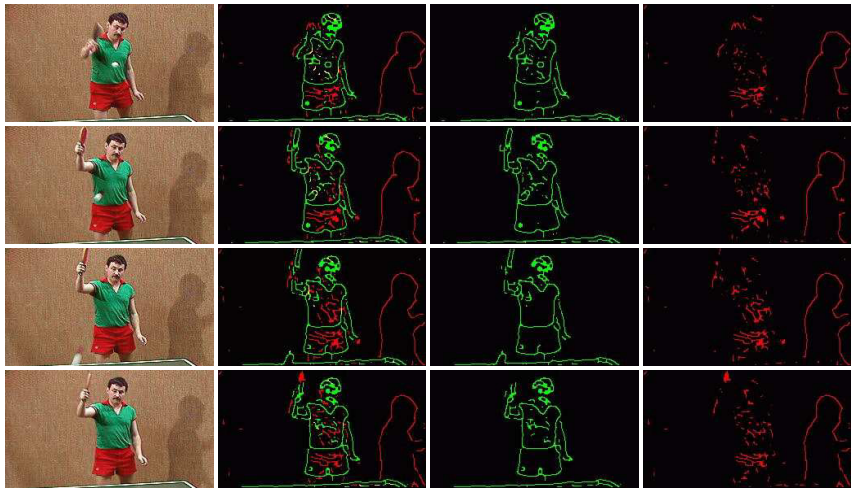
Edge Classification (1)

material

shadow or geometry



Edge Classification (2)

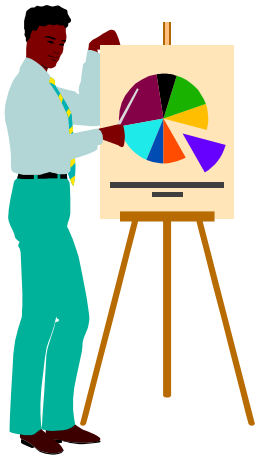


Reflectance-based Classification of Color Edges , Th. Gevers, ICCV 2003

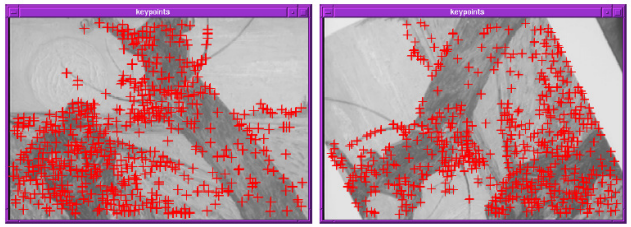
INTEREST POINT DETECTORS

1. Edges
2. Corners
3. Blobs
4. Templates

Harris Corner Detector



Finding Corners



Key property: in the region around a corner, image gradient has two or more dominant directions

Corners are repeatable and **distinctive**

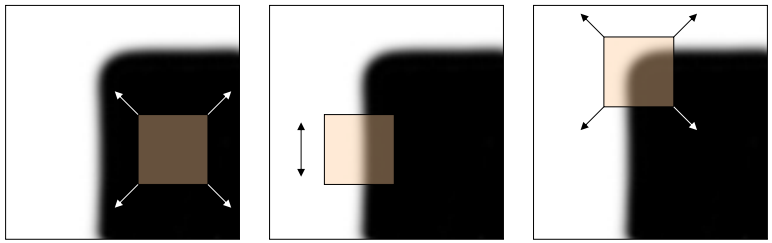
C.Harris and M.Stephens. "A Combined Corner and Edge Detector."
Proceedings of the 4th Alvey Vision Conference: pages 147--151.

Source: Lana Lazebnik

Corners as distinctive interest points

We should easily recognize the point by looking through a small 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

Harris Detector formulation

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

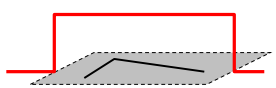
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

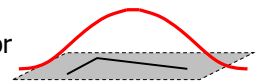
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

Source: R. Szeliski

Measuring the "Properties" of E()

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

$$= \sum_{x,y} [I(x,y) - I_x u - I_y v + \varepsilon - I(x,y)]^2 \approx$$

$$\approx \sum_{x,y} [-I_x u - I_y v]^2 =$$

$$= \sum_{x,y} I_x^2 u^2 + I_x u I_y v + I_y v I_x u + I_y^2 v^2 =$$

$$= \sum_{x,y} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix} \qquad M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

M depends on image properties

Harris Detector formulation

This measure of change can be approximated by:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

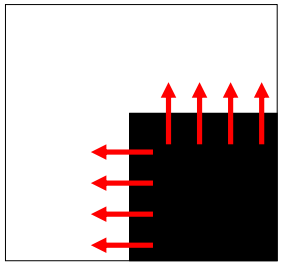
Gradient with respect to x, times gradient with respect to y

Sum over image region – area we are checking for corner

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y]$$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

What if we have a corner that is not aligned with the image axes?

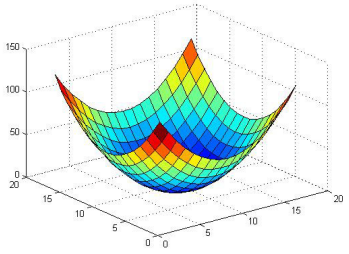
Slide credit: David Jacobs

Quadratic Form and its Eigenvalue

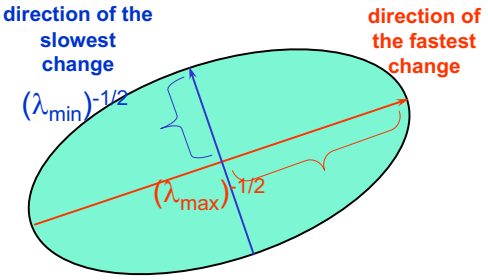
$$E(u,v) \cong [u,v] \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

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

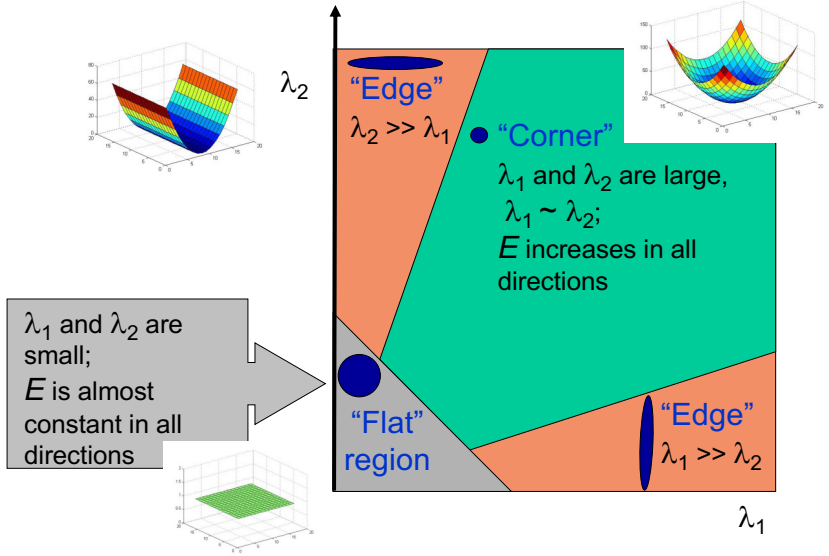
λ_1, λ_2 – eigenvalues of M



Ellipse $E(u,v)$ = const



Classification of Image Points ~~using Eigenvalues of *M*~~:



Harris Detector: Mathematics

Measure of corner response:

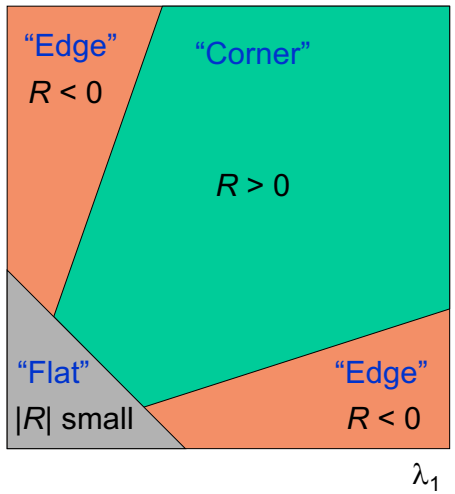
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

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

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

Harris Detector

- 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



$$R(x_0, y_0) = \det M - k\left(\operatorname{trace} M\right)^2$$

Summary of the Harris detector

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x.I_x \quad I_{y2} = I_y.I_y \quad I_{xy} = I_x.I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma\prime} * I_{x2} \quad S_{y2} = G_{\sigma\prime} * I_{y2} \quad S_{xy} = G_{\sigma\prime} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y2}(x,y) \end{bmatrix}$$

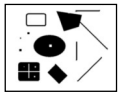
5. Compute the response of the detector at each pixel

$$R = Det(H) - k(Trace(H))^2$$

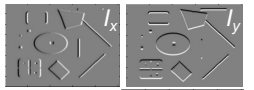
6. Threshold on value of R. Compute nonmax suppression.

Harris Detector [Harris88]

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



1. Image derivatives



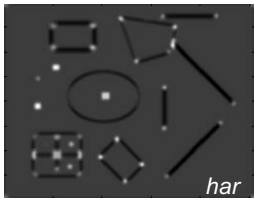
2. Square of derivatives

3. Gaussian filter $g(s_i)$

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

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

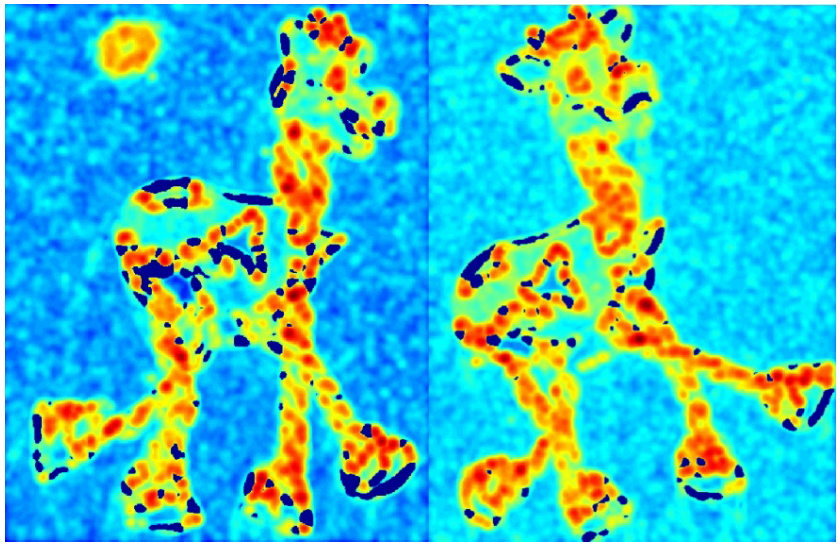


5. Non-maxima suppression

Harris Detector: Workflow

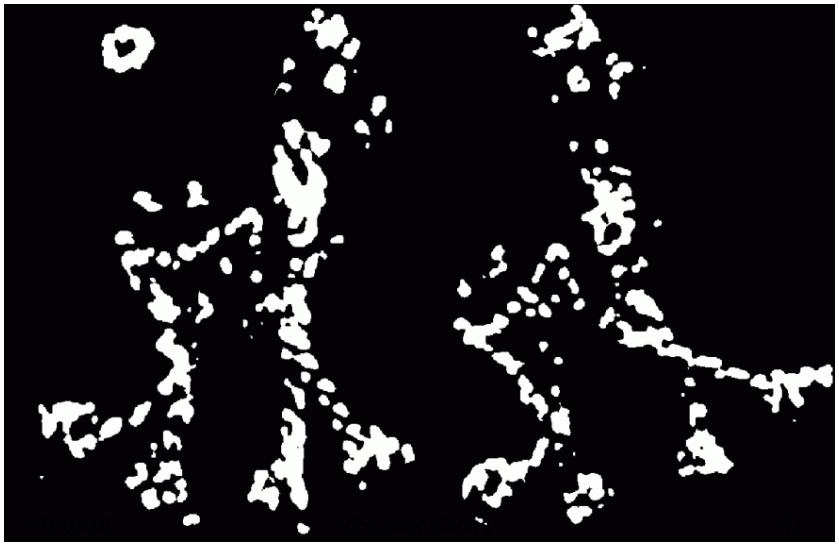
Harris Detector: Workflow

Compute corner response R



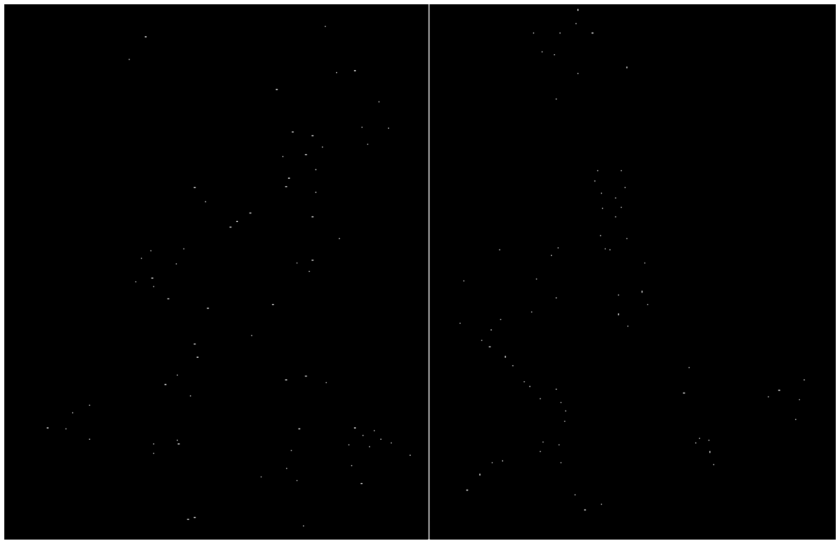
Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

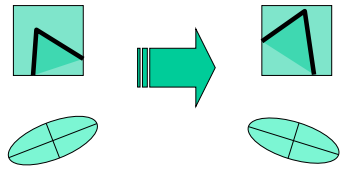
Take only the points of local maxima of R



Harris Detector: Workflow

Harris Detector: Properties

Rotation invariance



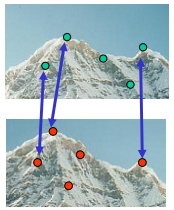
Ellipse rotates but its shape (i.e. eigenvalues) remains the same

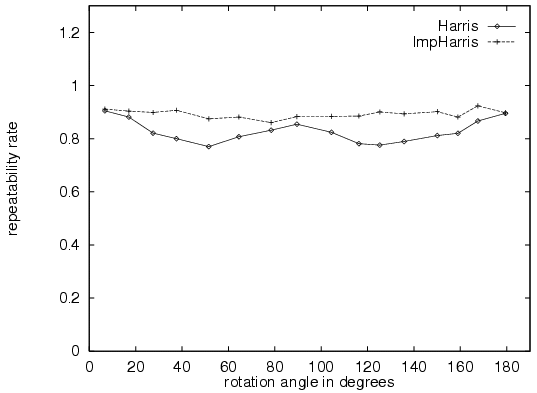
Corner response R is invariant to image rotation

Harris Detector: Properties

Repeatability rate:

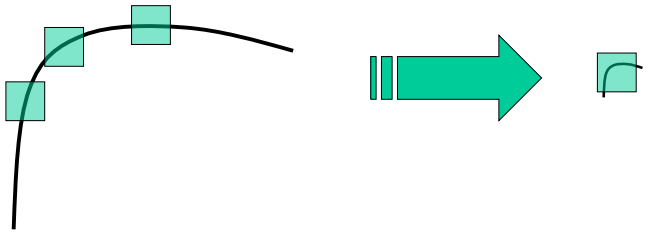
$$\frac{\text{\# correspondences}}{\text{\# possible correspondences}}$$





Harris Detector: Properties

Not invariant to image scale



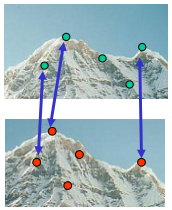
All points will be classified as edges

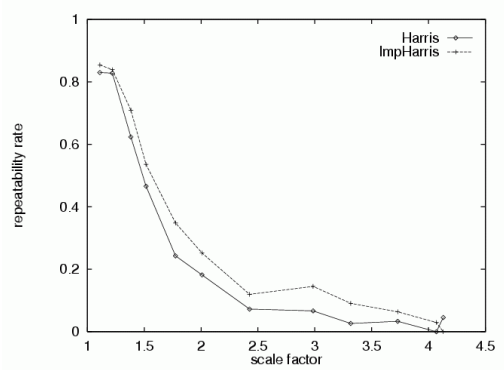
Corner !

Harris Detector: Properties

Repeatability rate:

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

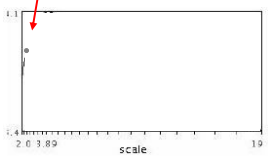




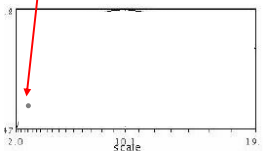
How can we detect **scale invariant** interest points?

Automatic Scale Selection

Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

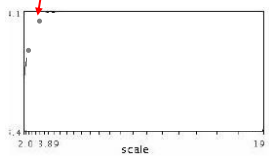


$f(I_{i_1\ldots i_m}(x',\sigma))$

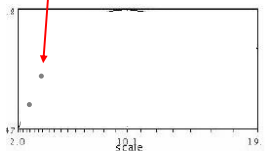
K. Grauman, B. Leibe

Automatic Scale Selection

Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

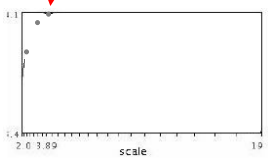


$f(I_{i_1\ldots i_m}(x',\sigma))$

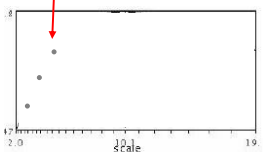
K. Grauman, B. Leibe

Automatic Scale Selection

Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

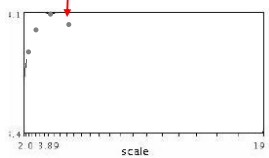


$f(I_{i_1\ldots i_m}(x',\sigma))$

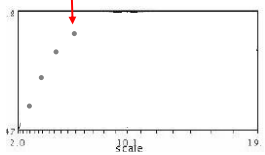
K. Grauman, B. Leibe

Automatic Scale Selection

Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

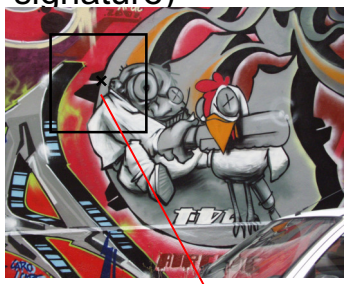


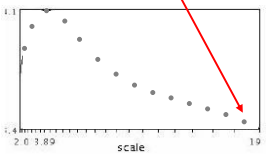
$f(I_{i_1\ldots i_m}(x',\sigma))$

K. Grauman, B. Leibe

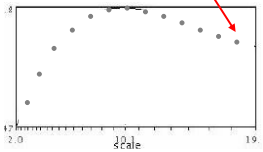
Automatic Scale Selection

Function responses for increasing scale (scale signature)





$f(I_{i_1\ldots i_m}(x,\sigma))$

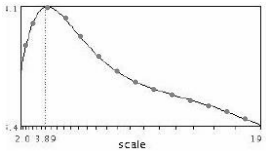


$f(I_{i_1\ldots i_m}(x',\sigma))$

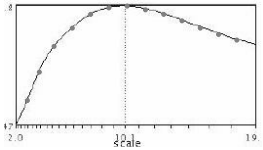
K. Grauman, B. Leibe

Automatic Scale Selection

Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

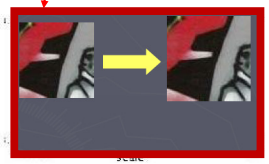


$f(I_{i_1\ldots i_m}(x',\sigma'))$

K. Grauman, B. Leibe

Automatic Scale Selection

Function responses for increasing scale (scale signature)



$f(I_{i_1\ldots i_m}(x,\sigma))$

$f(I_{i_1\ldots i_m}(x',\sigma'))$

K. Grauman, B. Leibe

INTEREST POINT DETECTORS

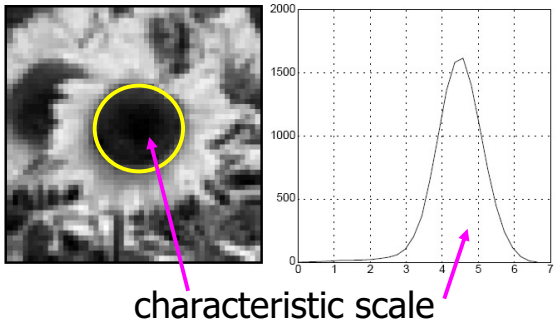
1. Edges
2. Corners
3. **Blobs**
4. Templates

Scale-space Blob Detector: Example

Image credit: Lana Lazebnik

Characteristic scale

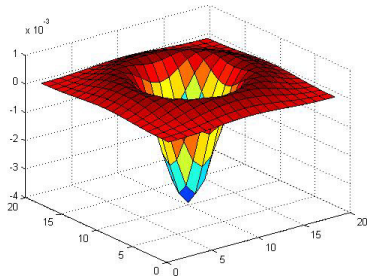
- We define the *characteristic scale* as the scale that produces peak of Laplacian response

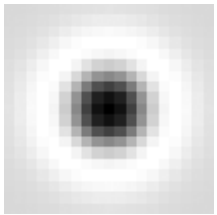


T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* **30** (2): pp 77--116. Source: Lana Lazebnik

Laplacian of Gaussian

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D



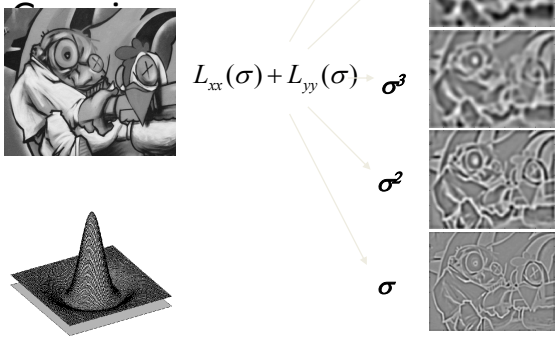


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Laplacian-of-Gaussian (LoG)

- Interest points:

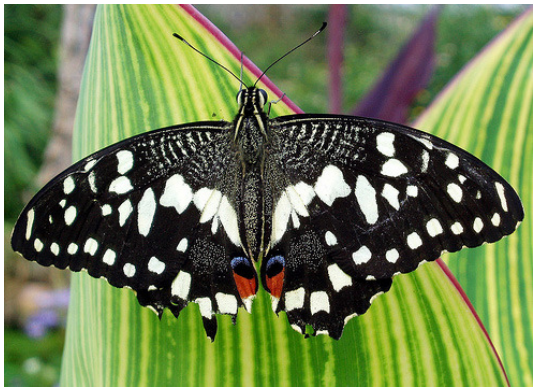
  Local maxima in scale space of Laplacian-of-



⇒ **List of** ***(x, y, σ)***

K. Grauman, B. Leibe
K. Grauman, B. Leibe

Scale-space blob detector: Example



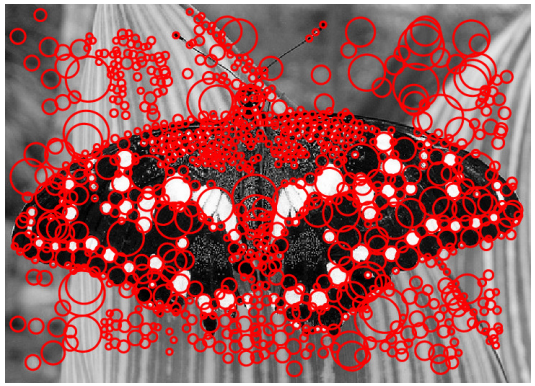
Source: Lana Lazebnik

Scale-space blob detector: Example

sigma = 11.9912

Source: Lana Lazebnik

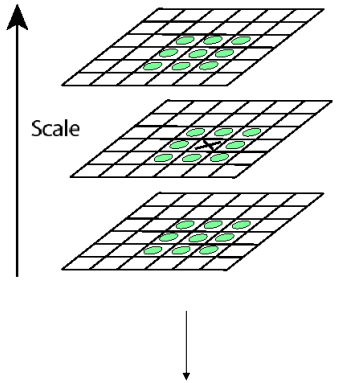
Scale-space blob detector: Example



Source: Lana Lazebnik

Key point localization with DoG

- Detect maxima of difference-of-Gaussian (DoG) in scale space
- Then reject points with low contrast (threshold)
- Eliminate edge responses

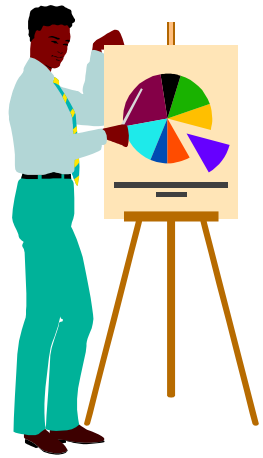


Candidate keypoints:
list of (x,y,σ)

INTEREST POINT DETECTORS

1. Edges
2. Corners
3. Blobs
4. Templates

Template Matching

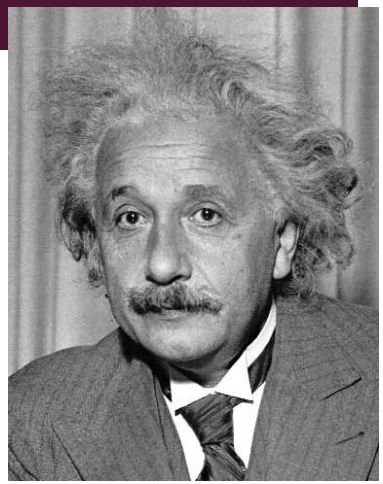


TEMPLATE MATCHING

- Goal: find  in image

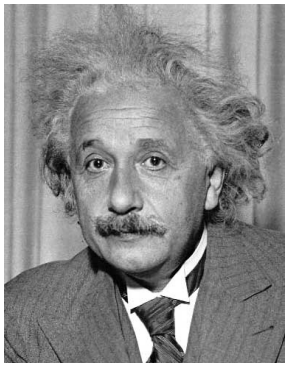
Main challenge:

- What is a good similarity or distance measure between two patches?
  - Sum Square Difference
  - Normalized Cross Correlation
  - (Matlab: normxcorr2)



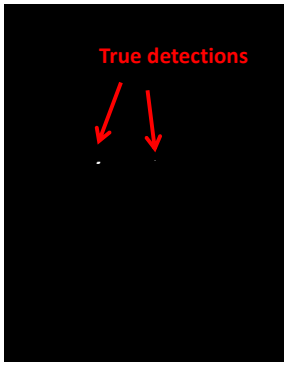
Template Matching

- Goal: find  in image
- SSD $h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$



Input

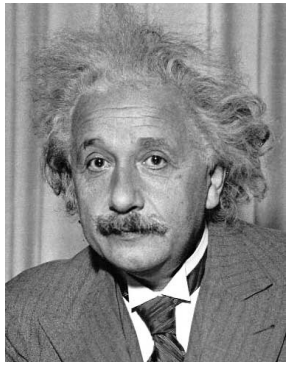
1- sqrt(SSD)



Thresholded Image

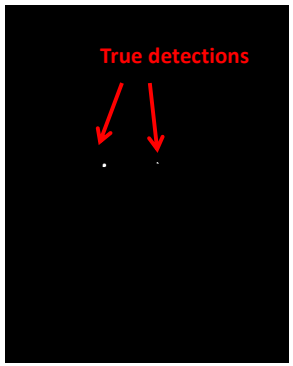
Template Matching

- Goal: find  in image
- Method 2: Normalized cross-correlation



Input

Normalized X-Correlation



Thresholded Image

Slide: Hoiem

WHERE WILL INTEREST POINTS COME FROM?

1. Edges
2. Corners
3. Blobs
4. Templates

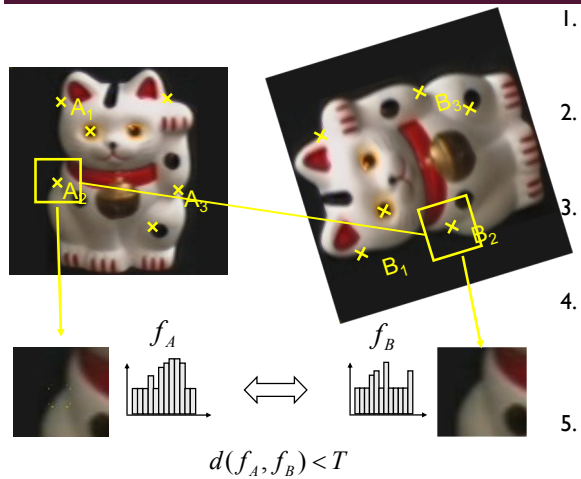
Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

From a study of Tuytelaars & Mikolajczyk

OVERVIEW OF KEYPOINT MATCHING



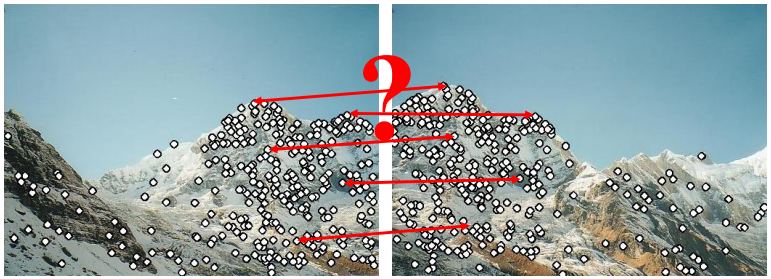
1. Find a set of distinctive keypoints
2. Define a local region around keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

How to describe interest points?

Local descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



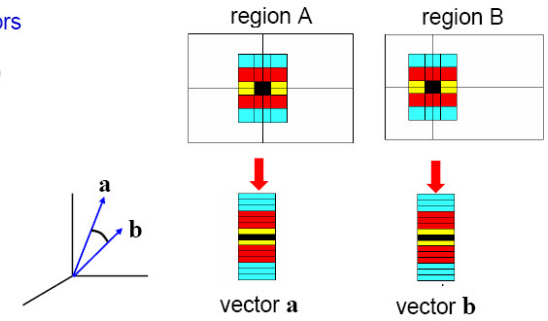
Point descriptor should be:
1. Invariant
2. Distinctive

Local descriptors

- Simplest descriptor: list of intensities within a patch.
- What is this going to be invariant to?

Write regions as vectors

$A \rightarrow \mathbf{a},\ B \rightarrow \mathbf{b}$

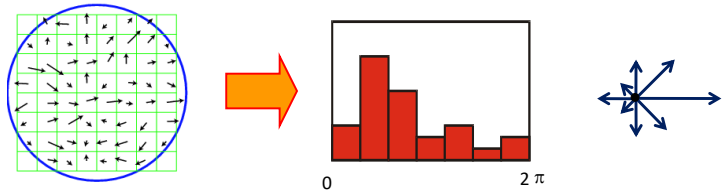


Feature descriptors

Disadvantage of patches as descriptors:

- Small shifts can affect matching score a lot

Solution: histograms



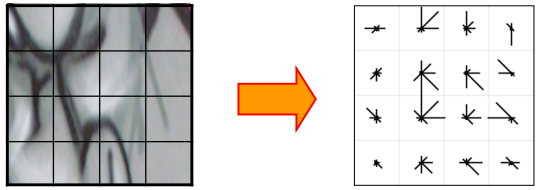
Source: Lana Lazebnik

Feature descriptors: SIFT

Scale **I**nvariant **F**eature **T**ransform

Descriptor computation:

- Divide patch into 4x4 sub-patches: 16 cells
- Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
- Resulting descriptor: 4x4x8 = 128 dimensions



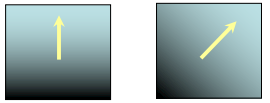
David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Source: Lana Lazebnik

Rotation Invariant Descriptors

- Find local orientation

  Dominant direction of gradient for the image patch



- Rotate patch according to this angle

  This puts the patches into a canonical orientation.

Rotation Invariant Descriptors

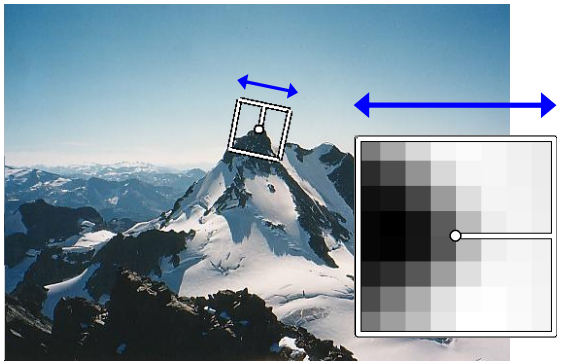
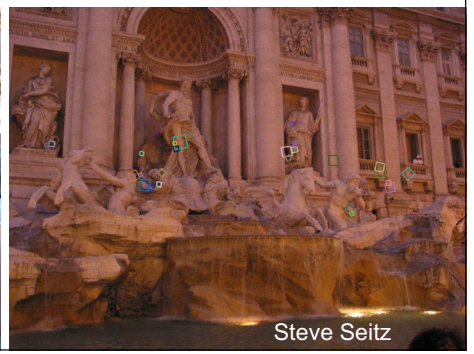


Image from Matthew Brown

Feature descriptors: SIFT

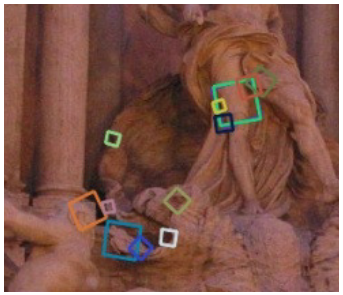
Extraordinarily robust matching technique

- 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



Working with SIFT descriptors

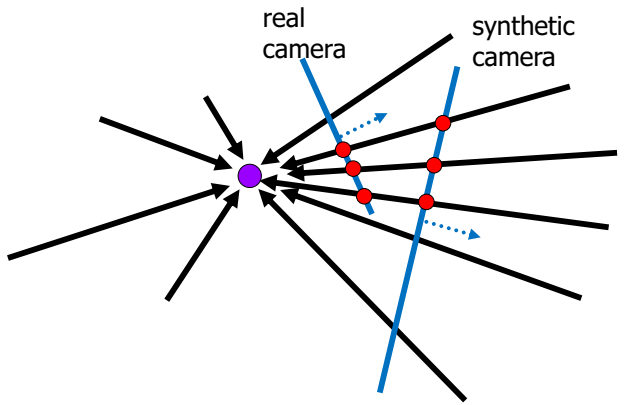
- One image yields:
  - n 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
    - [n x 128 matrix]
  - n scale parameters specifying the size of each patch
    - [n x 1 vector]
  - n orientation parameters specifying the angle of the patch
    - [n x 1 vector]
  - n 2d points giving positions of the patches
    - [n x 2 matrix]



How to match interest points?

Building a Panorama

Panoramas: generating synthetic views

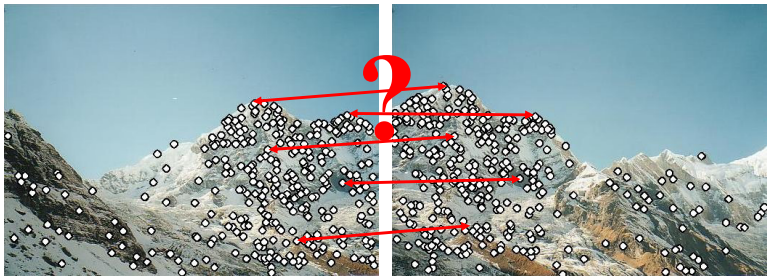


Can generate any synthetic camera view as long as it has **the same center of projection**!

Source: Alyosha Efros

FEATURE MATCHING

We know how to detect and describe good points

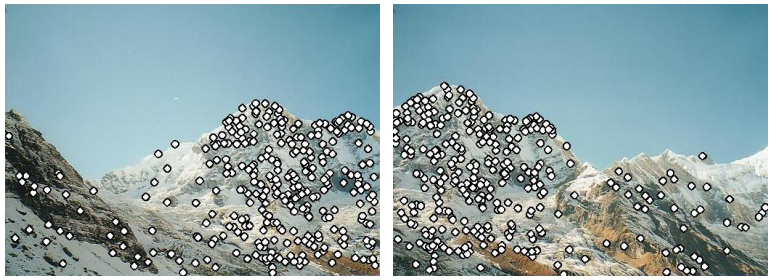


Given a feature in Img1, how to find the best match in Img2?

- Define distance function that compares two descriptors
- Test all the features in I2, find the one with min distance

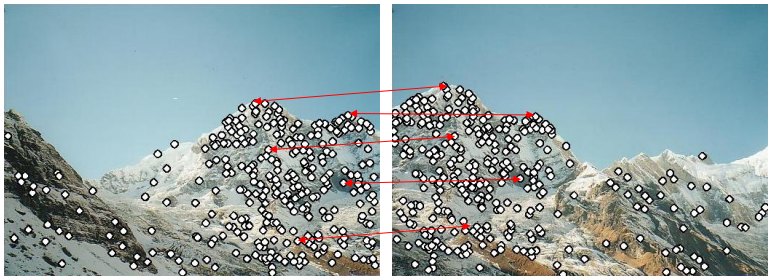
Building a Panorama

1) Detect feature points in both images



Building a Panorama

1. Detect feature points in both images
2. Find corresponding pairs



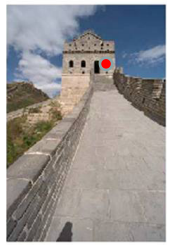
Building a Panorama

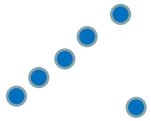
1. Detect feature points in both images
2. Find corresponding pairs
3. Find a parametric transformation (e.g. homography)
4. Warp (right image to left image)

$$\forall \text{ matching pair } \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{left} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{right}$$

Outliers

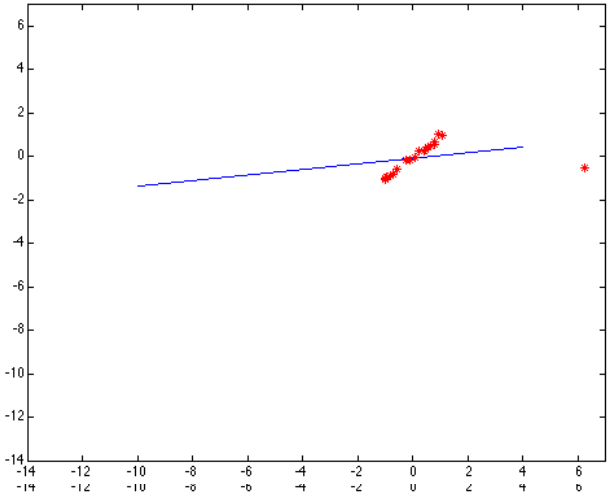
- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn't belong to the line we are fitting.





Grauman

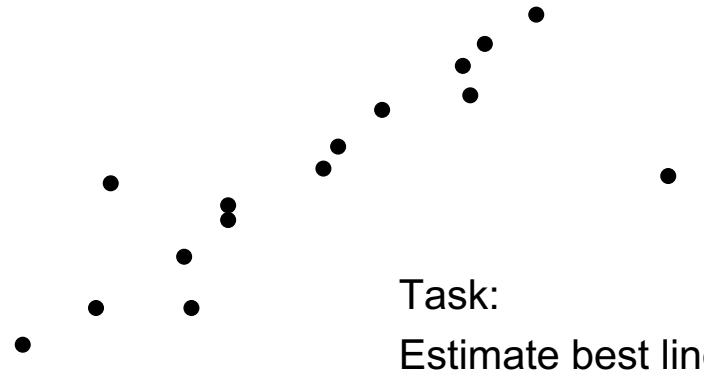
Example: Outliers in Linear Regression



RANSAC

- RANdom Sample Consensus
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

RANSAC Line Fitting Example

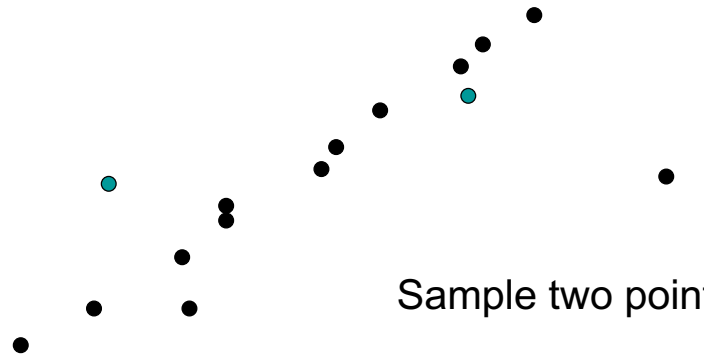


Task:

Estimate best line

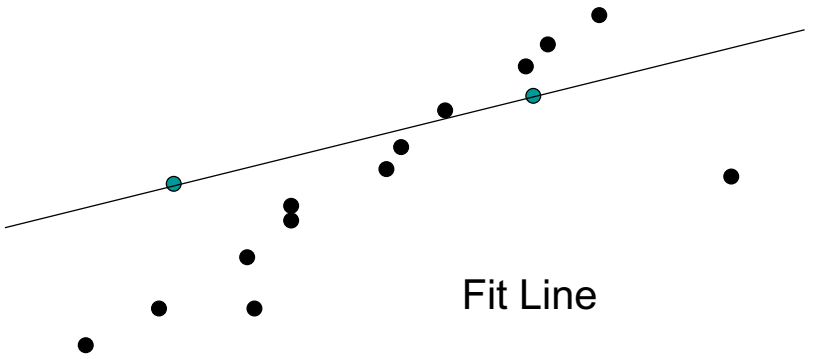
Slide credit: Jinxiang Chai, CMU

RANSAC Line Fitting Example



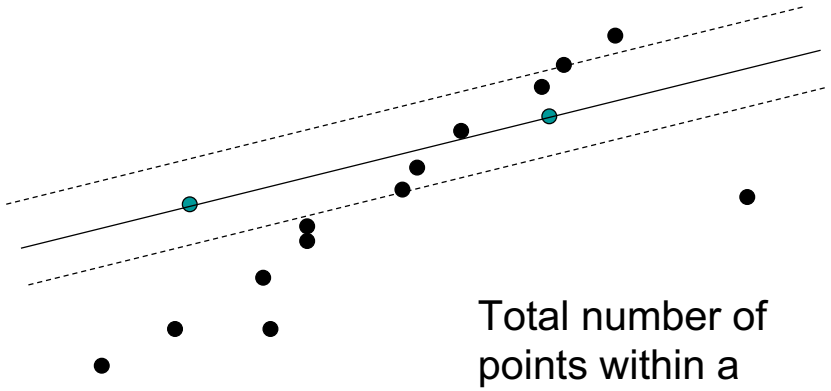
Sample two points

RANSAC Line Fitting Example



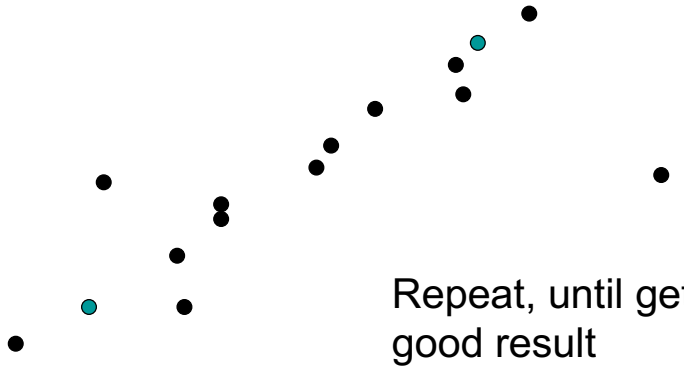
Fit Line

RANSAC Line Fitting Example



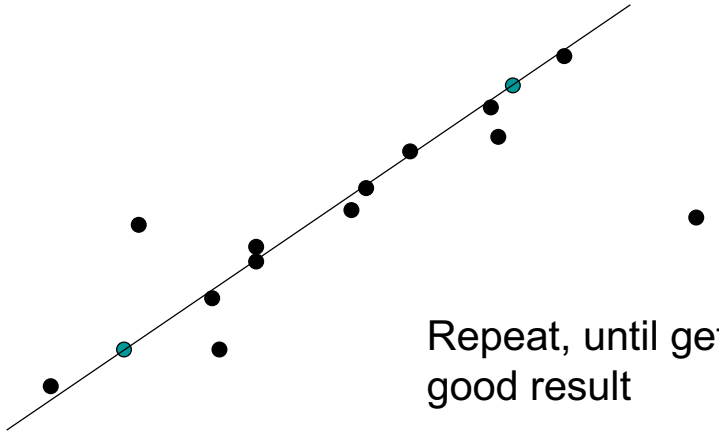
Total number of points within a threshold of line.

RANSAC Line Fitting Example



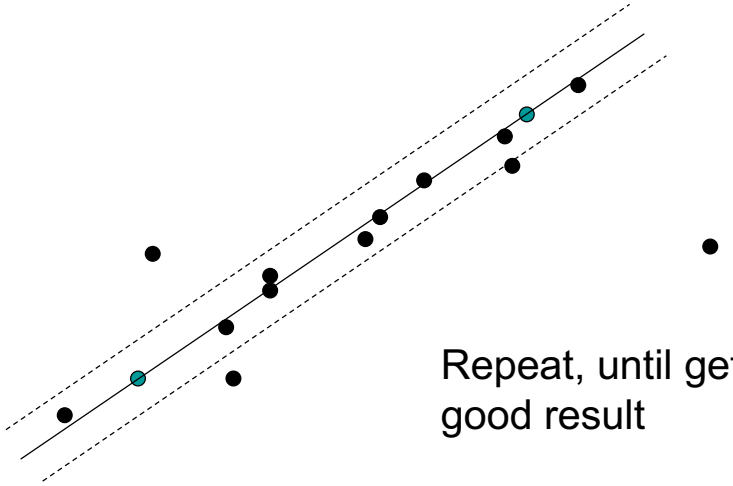
Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

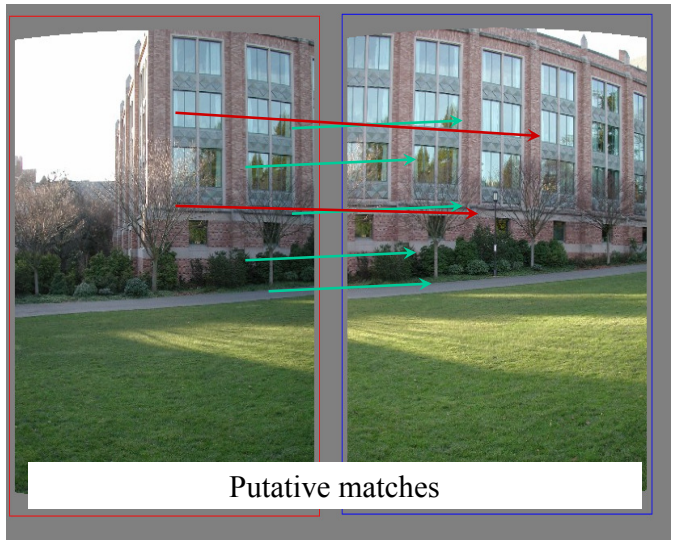
RANSAC for Estimating Homography

- RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $SSD(p_i', \mathbf{H}\, p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

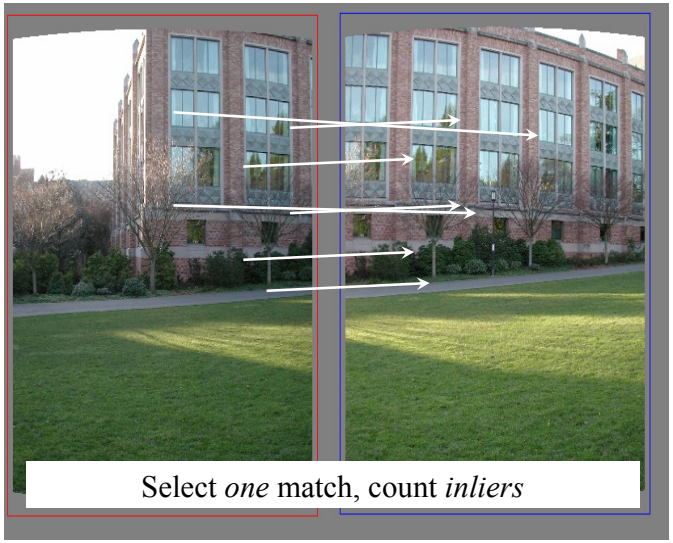
Slide: Derek Hoiem

RANSAC example: Translation

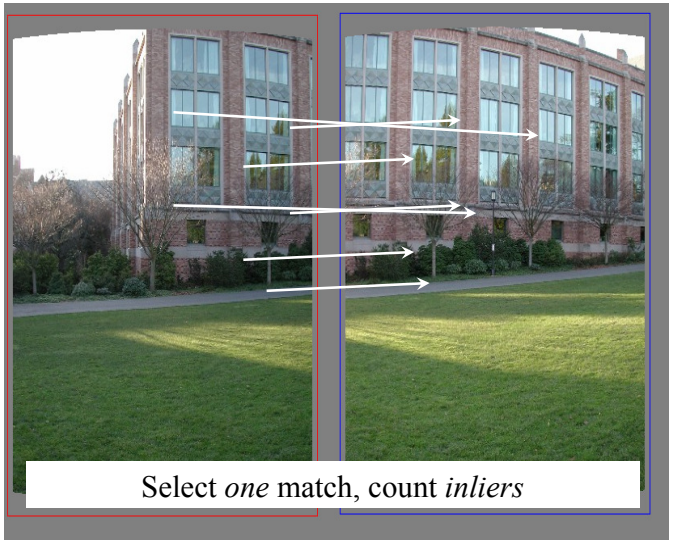


Source: Rick Szeliski

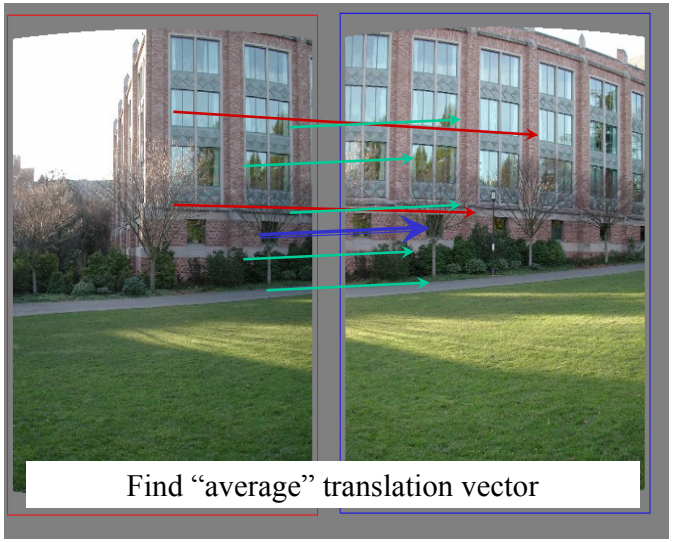
RANSAC example: Translation



RANSAC example: Translation



RANSAC example: Translation



Slide: Derek Hoiem

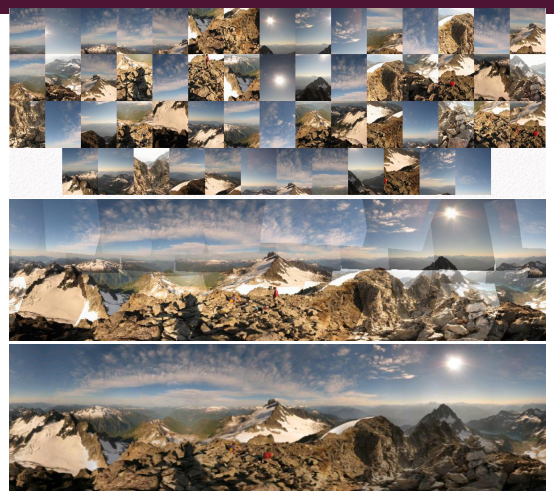
Slide: Derek Hoiem

Multi-band Blending

- Burt & Adelson 1983
  - Blend frequency bands over range $\propto l$

Slide: Derek Hoiem

YET ANOTHER EXAMPLE



http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

These 4 images show the same object (heart). Describe why a common setup of Interest point matching (eg using Harris corners / DoG Blobs and SIFT features) will have difficulties in matching these?

INTRIGUING QUESTION

TODAY'S CLASS

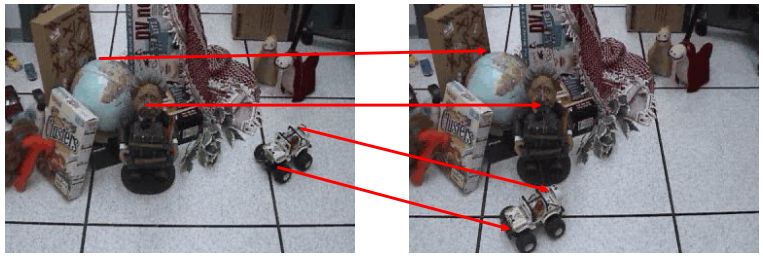
- Image Similarity
- Feature Detection
- **Motion**

MOTION & VISUAL TRACKING

- Optical Flow
- Template tracking
- Mean-shift tracking
- Foreground-Background Separation Tracking
- Tracking by Detection

OPTICAL FLOW

Where did each pixel in image 1 go to in image 2



OPTICAL FLOW (EXAMPLE VIDEO)

https://www.youtube.com/watch?v=Wgaukp4ZJ74

MOTIVATION

Stabilization

- Visual Motion can be annoying
  - Camera instabilities, jitter
  - Camera movement
- Measure it; remove it (stabilize)

Scene Recognition

- Visual Motion reveals spatial layout
  - Motion parallax

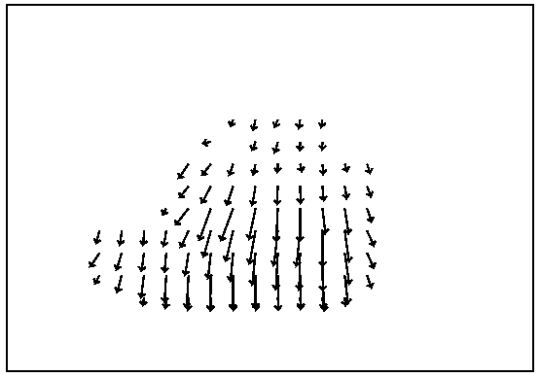
Object recognition

- Visual Motion indicates dynamics
  - How many moving objects are there.
  - Which directions are they moving in.
  - How fast are they moving.
- Track objects and analyze trajectories

Action Recognition

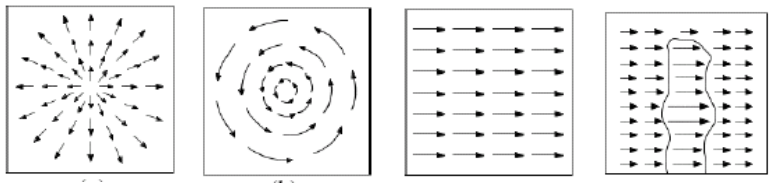
- Can we recognize their type of motion (e.g. walking, running, etc.).

FLOW FIELD



Pierre Kornprobst's Demo

Examples of Motion fields



Forward motion

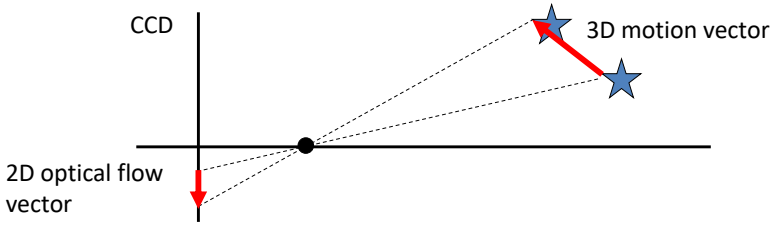
Rotation

Horizontal translation

Closer objects appear to move faster!!

Motion Field & Optical Flow Field

- Motion Field = Real world 3D motion
- Optical Flow Field = Projection of the motion field onto the 2d image



$$\vec{u} = (u, v)$$

The Optical Flow Field

- Goal:
  Find for each pixel a velocity vector
  which says:
  - In which direction it is moving
  - How quickly is the pixel moving across the image

Patch-based motion estimation

Image motion

How do we determine correspondences?

I J

Assume all change between frames is due to **motion!**

Estimating Optical Flow

Assume the image intensity I is constant

Time = *t* Time = *t+dt*

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Brightness Constancy Equation

$$I(x,y,t) = I(x+dx, y+dy, t+dt)$$

First order Taylor Expansion

$$= I(x,y,t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt$$

Assumption?

Simplify notations:

$$I_x dx + I_y dy + I_t dt = 0$$

Divide by *dt* and denote:

$$u = \frac{dx}{dt} \quad v = \frac{dy}{dt}$$

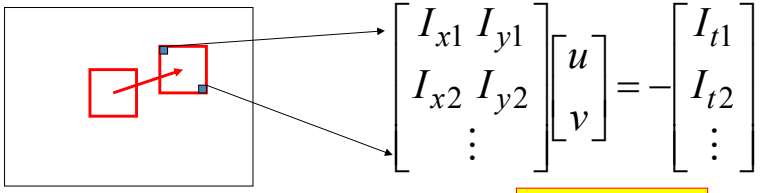
$$I_x u + I_y v = -I_t$$

Problem: One equation, two unknowns

Local Smoothness: Lucas Kanade

$$I_x u + I_y v = -I_t \quad \Rightarrow \quad \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

Assume constant (u,v) in small neighborhood



$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix}$$

$$A\vec{u} = b$$

Conditions for Solvability

– Optimal (u, v) satisfies Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

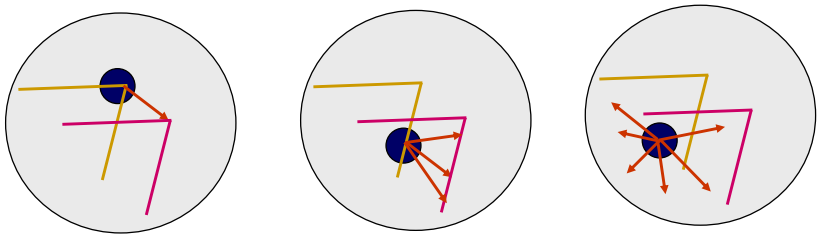
$A^T A$ $\qquad$ $A^T b$

- When is This Solvable?
  - **A**^{T}**A** should be invertible
  - **A**^{T}**A** should not be too small due to noise
    - eigenvalues l_1 and l_2 of **A**^{T}**A** should not be too small
  - **A**^{T}**A** should be well-conditioned
    - l_1 / l_2 should not be too large (l_1 = larger eigenvalue)

•* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

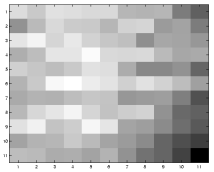
Local Patch Analysis

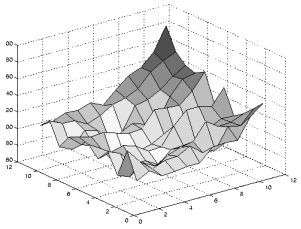
- How *certain* are the motion estimates?



Szelisk

Low Texture Region



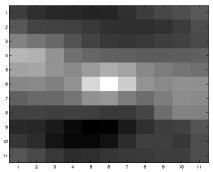


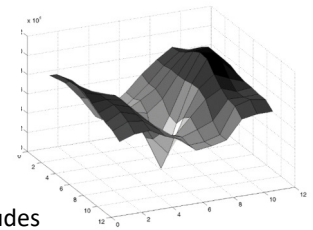
$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small l_1, small l_2

High Textured Region





$$\sum \nabla I (\nabla I)^T$$

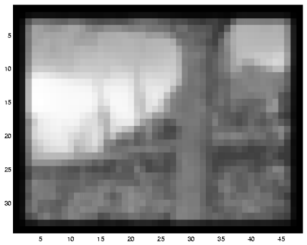
– gradients are different, large magnitudes

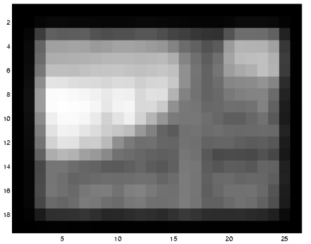
– large l_1, large l_2

Small Motion Assumption

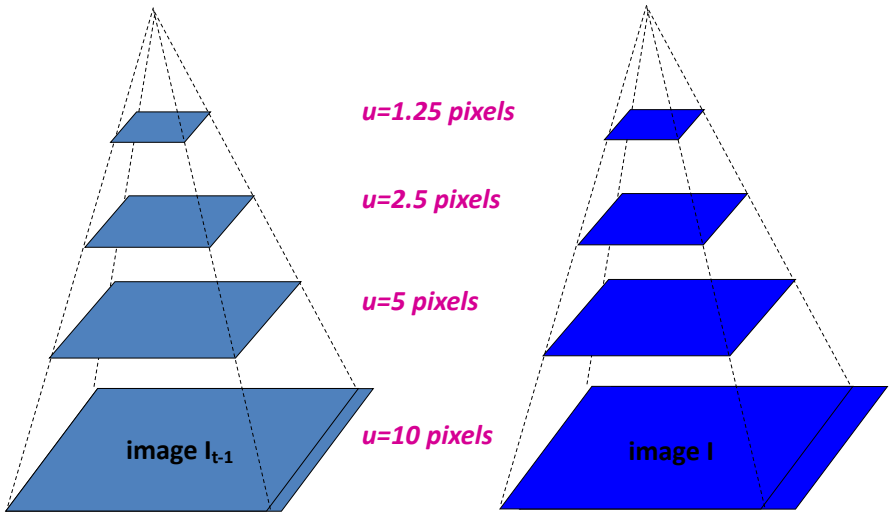
- Is this motion small enough?
  - Probably not—it's much larger than one pixel (Taylor expansion doesn't hold)
  - How might we solve this problem?

Reduce the Resolution





Coarse-to-fine Optical Flow



OPTICAL FLOW (EXAMPLE VIDEO)

https://www.youtube.com/watch?v=Wgaukp4ZJ74

OPTICAL FLOW (RECAP)

What are the (hidden) assumption behind this flow computation?

- Finding corresponding key-points in subsequent frames
- Small motion between frames
- Intensity/Color constancy between frames

Steps

1. Find corresponding patches
2. Compute flow over small neighborhood
   I. From coarse to fine
   II. Using Lucas-Kanade Patch Translation

OTHER BREAK-DOWNS

Brightness constancy is **not** satisfied

→ Correlation based methods

A point does **not** move like its neighbors

what is the ideal window size?

→ Regularization based methods

MOTION & VISUAL TRACKING

- Optical Flow
- **Template tracking**
- **Mean-shift tracking**
- **Foreground-Background Separation Tracking**
- **Tracking by Detection**

Not discussed

TODAY'S CLASS

- Image Similarity
- Feature Detection
- Motion

- Deadline lab assignment: Gabor & Gaussian Filters
- Lab session (March 1): Optical Flow & Harris
- Lecture 4 (Tuesday): Features & Classification

QUESTIONS?

The important thing is not to stop questioning; curiosity has its own reason for existing.
— Albert Einstein

SLIDE CREDITS

- Theo Gevers
- Trevor Darell (https://people.eecs.berkeley.edu/~trevor/CS280Notes/)
- Bill Freeman
- Kristen Grauman
- Tinne Tuytelaars
- **And many others…**

Computer Vision 1 (total #slides 169 | Lecture 4)

Summary

1. **Feature Detection**
2. **Motion**
3. **Classification -> next week!**