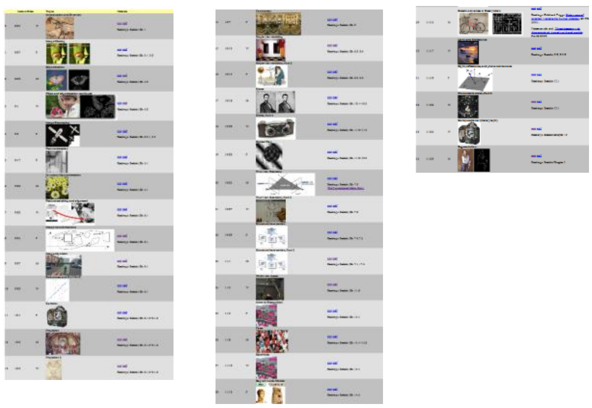
CS4670 / 5670: Computer Vision

Noah Snavely

Lecture 36: Course review



Announcements

- Project 5 due tonight, 11:59pm
- Final exam Monday, Dec 10, 9am
  - Open-note, closed-book
- Course evals
  - http://www.engineering.cornell.edu/CourseEval/

Questions?

Neat Video

Topics – image processing

- Filtering
- Edge detection
- Image resampling / aliasing / interpolation
- Feature detection
  - Harris corners
  - SIFT
  - Invariant features
- Feature matching

Topics – 2D geometry

- Image transformations
- Image alignment / least squares
- RANSAC
- Panoramas

Topics – 3D geometry

- Cameras
- Perspective projection
- Single-view modeling
- Stereo
- Two-view geometry (F-matrices, E-matrices)
- Structure from motion
- Multi-view stereo

Topics – recognition

- Skin detection / probabilistic modeling
- Eigenfaces
- Viola-Jones face detection (cascades / adaboost)
- Bag-of-words models
- Segmentation / graph cuts

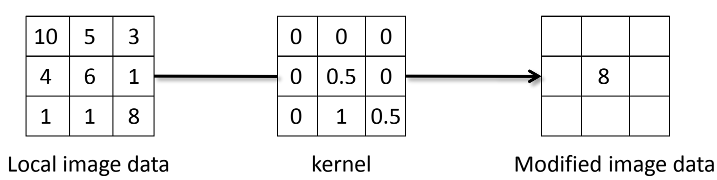
Topics – Light, reflectance, cameras

- Light, BRDFS
- Photometric stereo

Image Processing

Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)



Source: L. Zhang

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

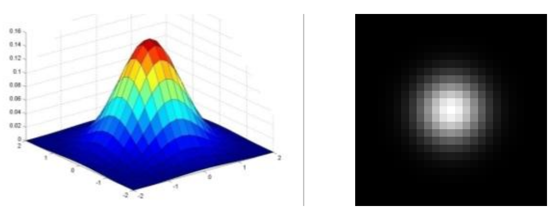
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Gaussian Kernel



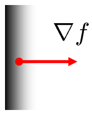
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

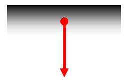
Image gradient

- The *gradient* of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

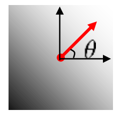
The gradient points in the direction of most rapid increase in intensity



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

The *edge strength* is given by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- how does this relate to the direction of the edge?

Source: Steve Seitz

Finding edges

gradient magnitude

Finding edges

thinning

(non-maximum suppression)

Image sub-sampling

1/2 1/4 (2x zoom) 1/8 (4x zoom)

Why does this look so crufty?

Source: S. Seitz

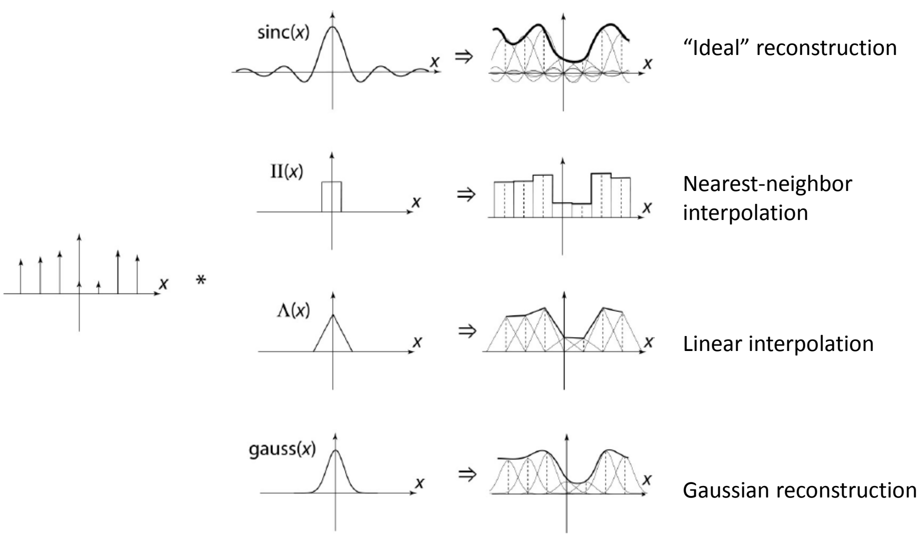
Subsampling with Gaussian pre-filtering

Gaussian 1/2 G 1/4 G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

Image interpolation



Source: B. Curless

Image interpolation

Original image:  x 10

Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation

The second moment matrix

The surface *E*(*u*,*v*) is locally approximated by a quadratic form.

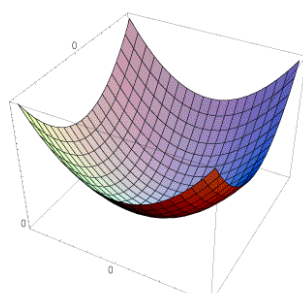
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



The Harris operator

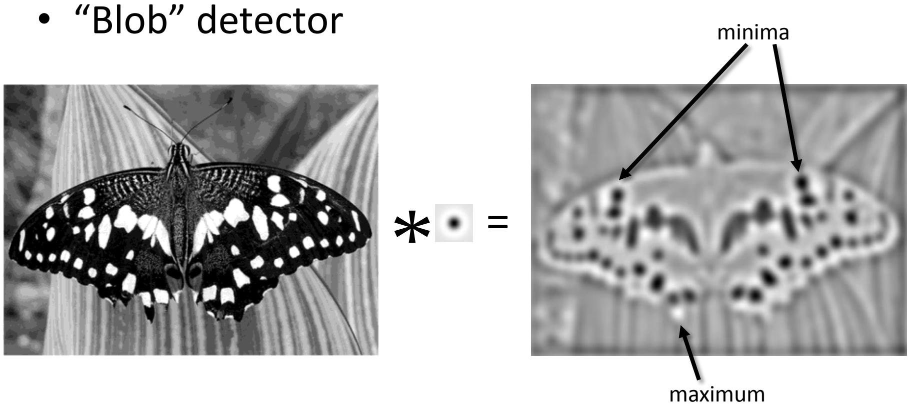
λ_{min} is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$
$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., *trace(H)* = $h_{11} + h_{22}$
- Very similar to λ_{min} but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

Laplacian of Gaussian

- "Blob" detector



- Find maxima *and minima* of LoG operator in space and scale

Scale-space blob detector: Example

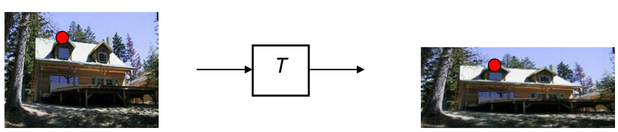
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = $||f_1 - f_2|| / ||f_1 - f_2'||$
  - 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 large values for ambiguous matches

2D Geometry

Parametric (global) warping



p = (x,y) **p'** = (x',y')

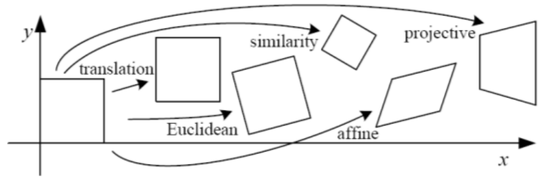
- Transformation T is a coordinate-changing machine:

  p' = *T*(p)

- What does it mean that *T* is global?
  - Is the same for any point p
  - can be described by just a few numbers (parameters)
- Let's consider *linear* xforms (can be represented by a 2D matrix):

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	3	lengths $+\cdots$	
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	4	angles $+\cdots$	
affine	$\left[\, \boldsymbol{A} \,\right]_{2\times3}$	6	parallelism $+\cdots$	
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{3\times3}$	8	straight lines	

These transformations are a nested set of groups

- Closed under composition and inverse is a member

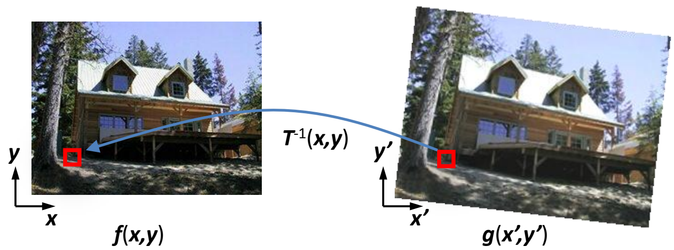
Projective Transformations aka Homographies aka Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)

Inverse Warping

- Get each pixel ***g***(***x'***,***y'***) from its corresponding location (***x***,***y***) = T^{-1}(***x***,***y***) in ***f***(***x***,***y***)
  - Requires taking the inverse of the transform



Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Affine transformations

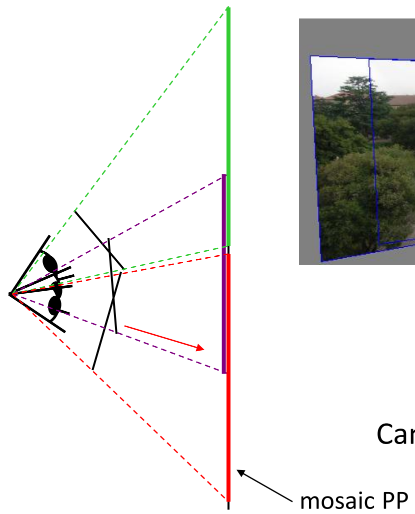
- Matrix form

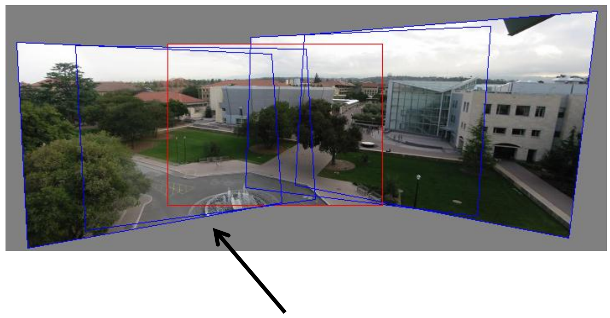
$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix}}_{\mathbf{A}\ (2n \times 6)} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}}_{\mathbf{t}\ (6 \times 1)} = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}}_{\mathbf{b}\ (2n \times 1)}$$

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

Projecting images onto a common plane



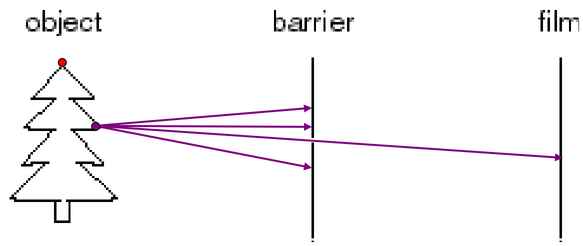


each image is warped with a homography $\mathbf{H}$

Can't create a 360 panorama this way…

3D Geometry

Pinhole camera



- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening known as the **aperture**
  - How does this transform the image?

Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

This is known as **perspective projection**

- The matrix is the **projection matrix**

Projection matrix

$$\Pi = \underbrace{\mathbf{K}}_{\text{intrinsics}} \underbrace{\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}}_{\text{translation}}}$$

$$\left[\ \mathbf{R} \ \middle| \ \underbrace{-\mathbf{R}\mathbf{c}}_{} \ \right]$$

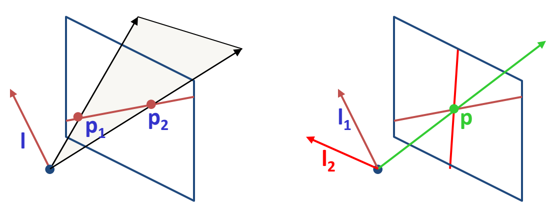
(**t** in book's notation)

$$\Pi = \mathbf{K} \left[\ \mathbf{R} \ \middle| \ -\mathbf{R}\mathbf{c} \ \right]$$

Point and line duality

– A line **l** is a homogeneous 3-vector

– It is $\perp$ to every point (ray) **p** on the line: **l p**=0



What is the line **l** spanned by rays $\mathbf{p}_1$ and $\mathbf{p}_2$?

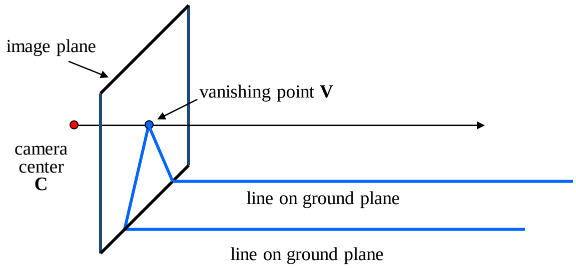
- **l** is $\perp$ to $\mathbf{p}_1$ and $\mathbf{p}_2$ $\Rightarrow$ $\mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- **l** can be interpreted as a *plane normal*

What is the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$?

- **p** is $\perp$ to $\mathbf{l}_1$ and $\mathbf{l}_2$ $\Rightarrow$ $\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$

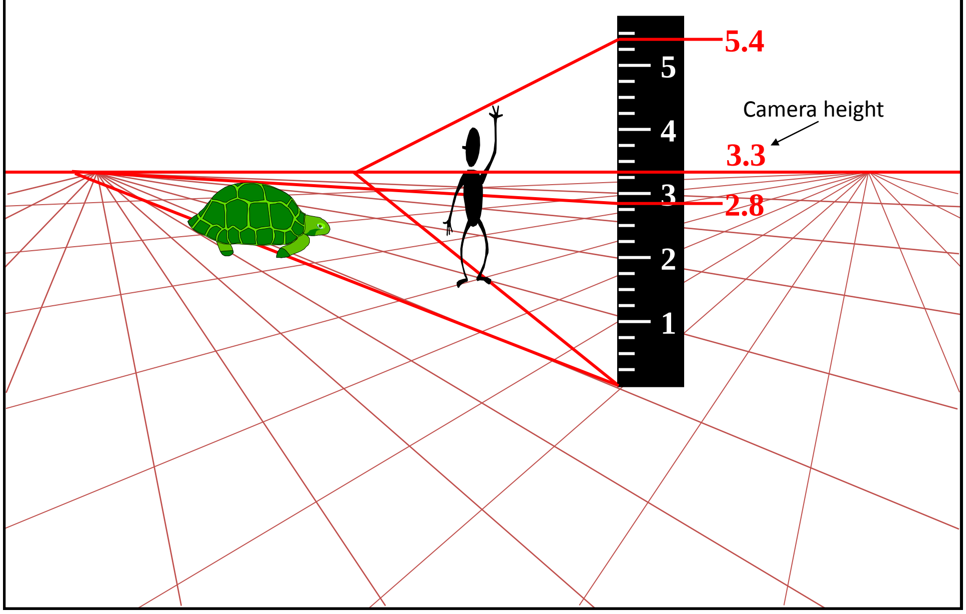
Points and lines are *dual* in projective space

Vanishing points

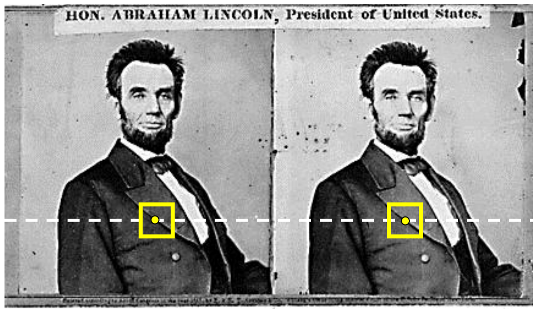


- Properties
  - Any two parallel lines (in 3D) have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point
    - in fact, every image point is a potential vanishing point

Measuring height



Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

Stereo as energy minimization

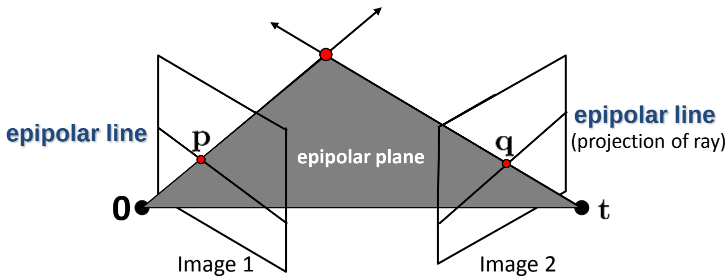
- Better objective function

$$E(d) = \underbrace{E_d(d)}_{\text{match cost}} + \lambda \underbrace{E_s(d)}_{\text{smoothness cost}}$$

Want each pixel to find a good match in the other image

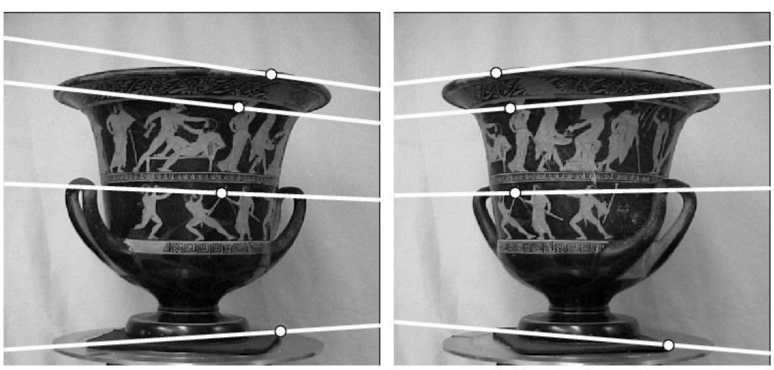
Adjacent pixels should (usually) move about the same amount

Fundamental matrix



- This *epipolar geometry* of two views is described by a Very Special 3x3 matrix $\mathbf{F}$, called the *fundamental matrix*
- $\mathbf{F}$ maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point **p** is: $\mathbf{F}\mathbf{p}$
- *Epipolar constraint* on corresponding points: $\mathbf{q}^T\mathbf{F}\mathbf{p} = 0$

Epipolar geometry demo

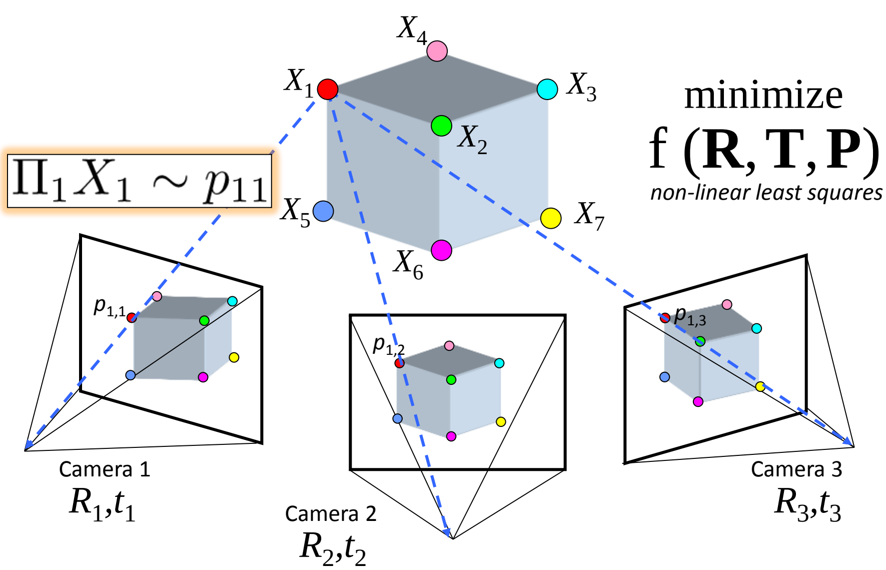


8-point algorithm

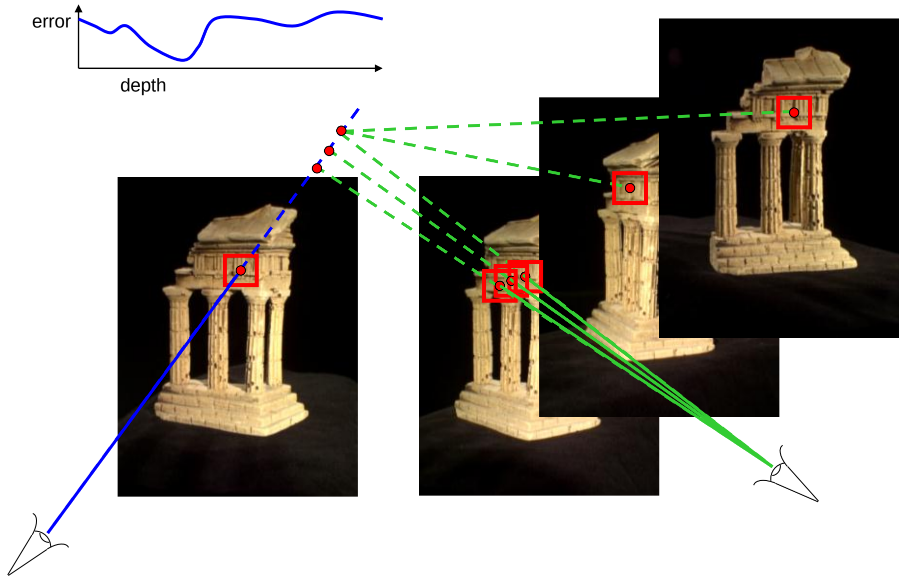
$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

Structure from motion



Stereo: another view



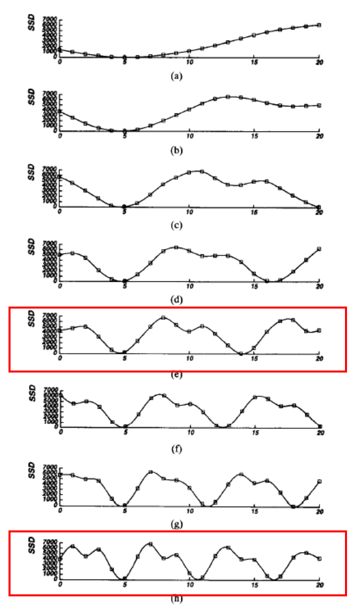


Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

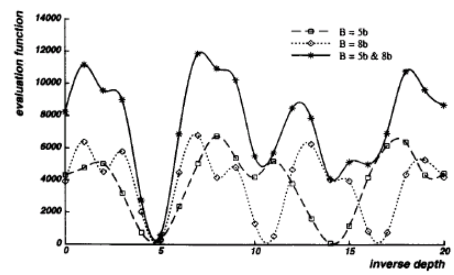


Fig. 6. Combining two stereo pairs with different baselines.

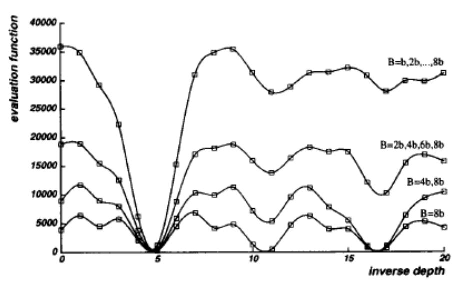


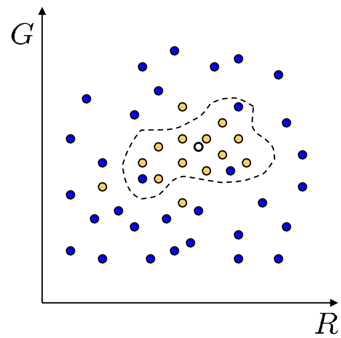
Fig. 7. Combining multiple baseline stereo pairs.

Recognition

Face detection

- Do these images contain faces? Where?

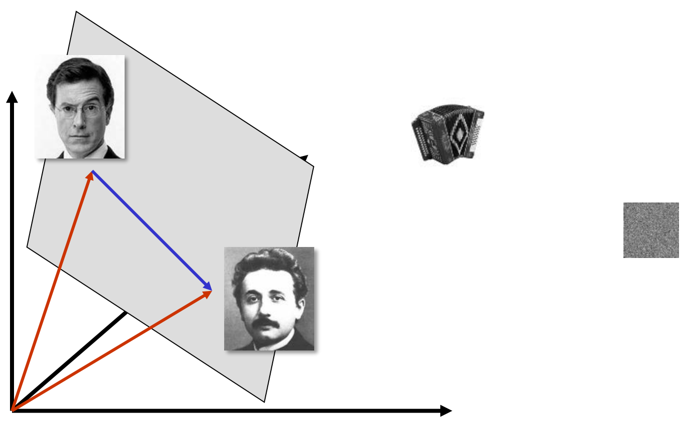
Skin classification techniques



Skin classifier

- Given X = (R,G,B): how to determine if it is skin or not?
- Nearest neighbor
  - find labeled pixel closest to X
  - choose the label for that pixel
- Data modeling
  - fit a model (curve, surface, or volume) to each class
- Probabilistic data modeling
  - fit a probability model to each class

Dimensionality reduction



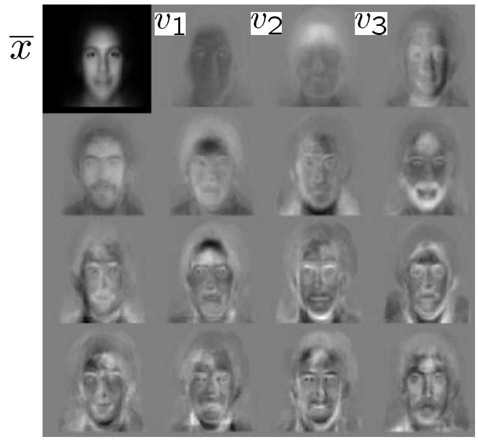
The set of faces is a "subspace" of the set of images

- Suppose it is K dimensional
- We can find the best subspace using PCA
- This is like fitting a "hyper-plane" to the set of faces
  - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_K}$
  - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

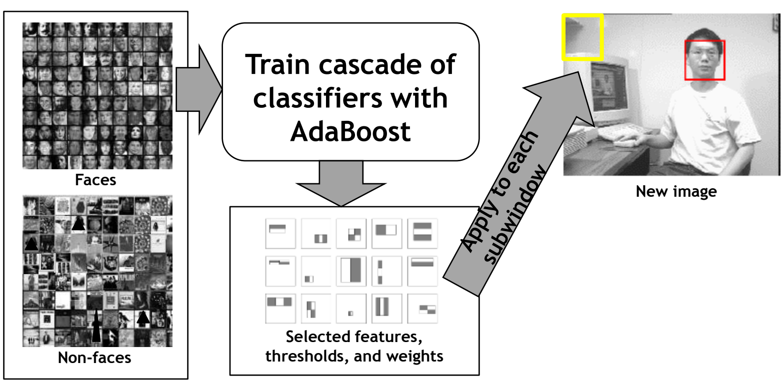
Eigenfaces

PCA extracts the eigenvectors of **A**

- Gives a set of vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, ...
- Each one of these vectors is a direction in face space
  - what do these look like?



Viola-Jones Face Detector: Summary



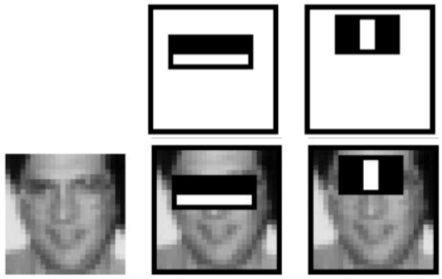
- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

Visual Object Recognition Tutorial

K. Grauman, B. Leibe

56

Viola-Jones Face Detector: Results



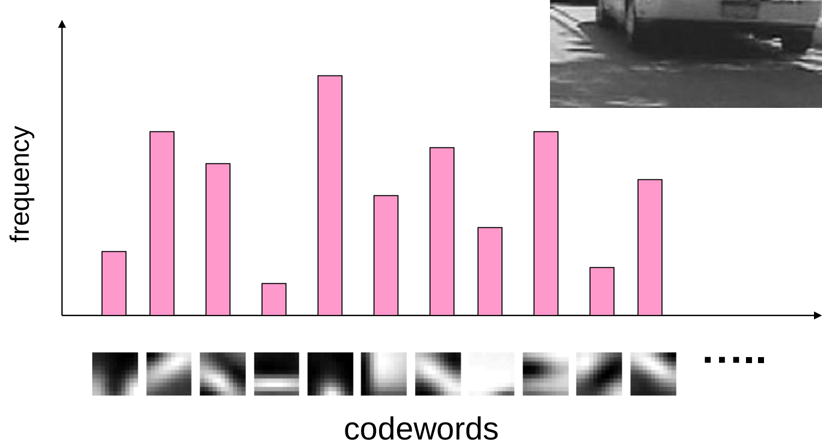
First two features selected

Visual Object Recognition Tutorial

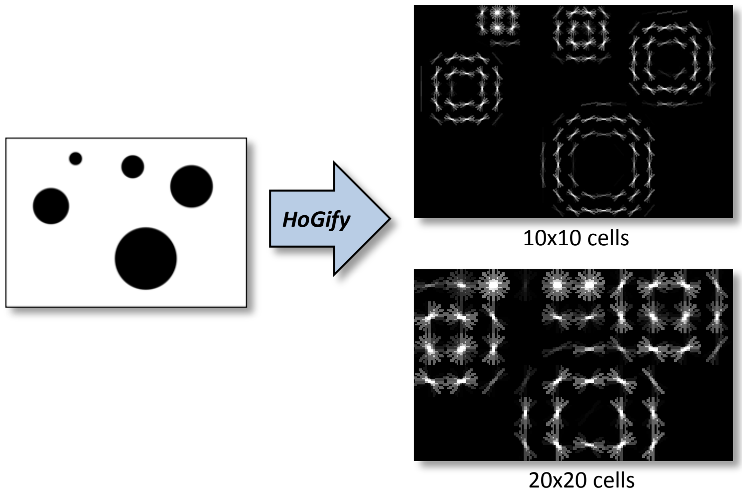
K. Grauman, B. Leibe

57

Bag-of-words models

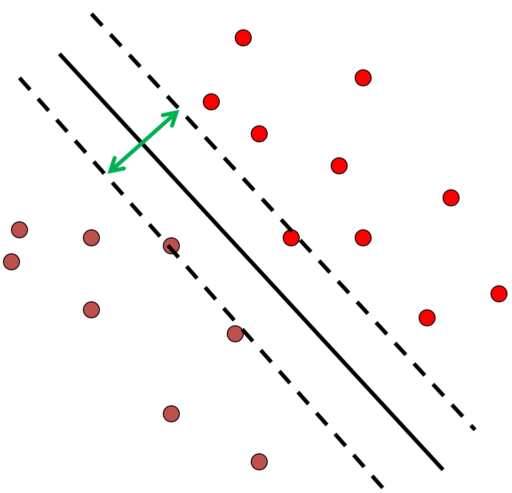


Histogram of Oriented Gradients (HoG)



[Dalal and Triggs, CVPR 2005]

Support Vector Machines (SVMs)

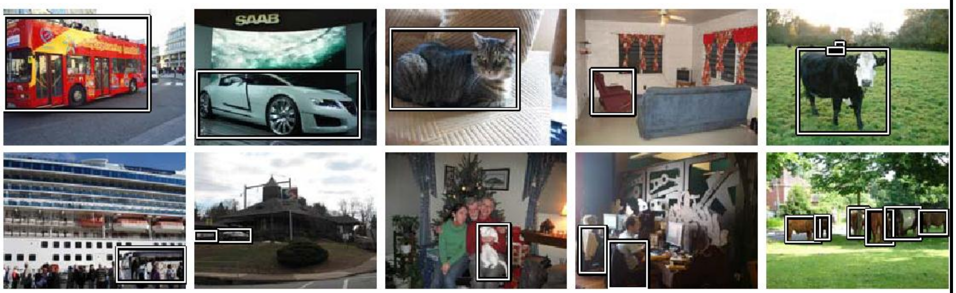


- Discriminative classifier based on *optimal separating line (for 2D case)*
- Maximize the *margin* between the positive and negative training examples

[slide credit: Kristin Grauman]

Vision Contests

- PASCAL VOC Challenge



- 20 categories
- Annual classification, detection, segmentation, … challenges

Binary segmentation

- Suppose we want to segment an image into foreground and background

Binary segmentation as energy minimization

- Define a labeling *L* as an assignment of each pixel with a 0-1 label (background or foreground)

- Problem statement: find the labeling *L* that minimizes

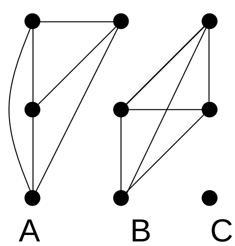
$$E(L) = \underbrace{E_d(L)}_{\text{match cost}} + \lambda \underbrace{E_s(L)}_{\text{smoothness cost}}$$

match cost

("how similar is each labeled pixel to the foreground / background?")

smoothness cost

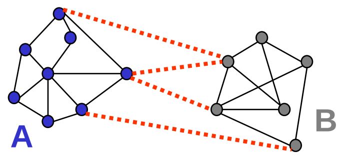
Segmentation by Graph Cuts



Break Graph into Segments

- Delete links that cross between segments
- Easiest to break links that have low cost (similarity)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Cuts in a graph



Link Cut

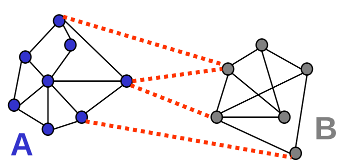
- set of links whose removal makes a graph disconnected
- cost of a cut:

$$cut(A,B) = \sum_{p \in A, q \in B} c_{p,q}$$

Find minimum cut

- gives you a segmentation

Cuts in a graph



Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

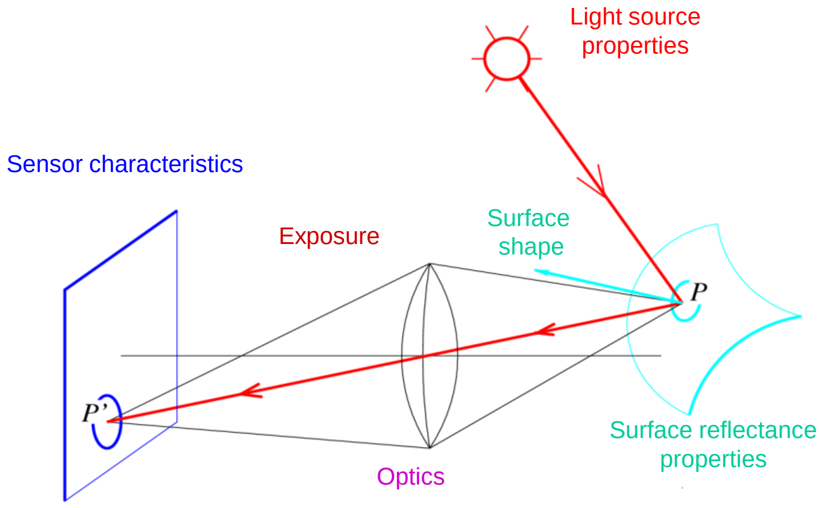
$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

- volume(A) = sum of costs of all edges that touch A

Light, reflectance, cameras

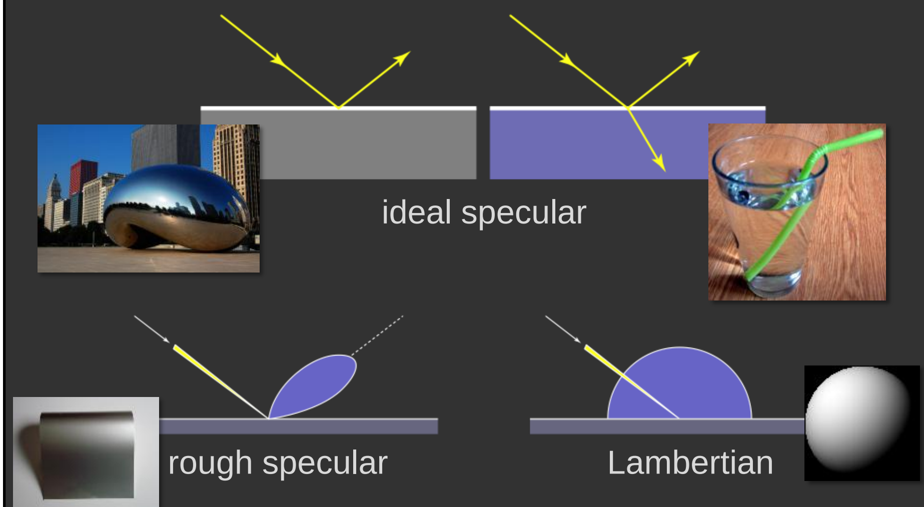
Radiometry

What determines the brightness of an image pixel?



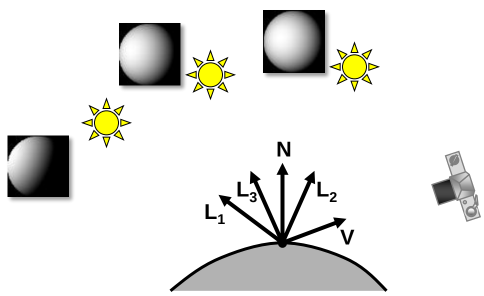
Slide by L. Fei-Fei

Classic reflection behavior



from Steve Marschner

Photometric stereo

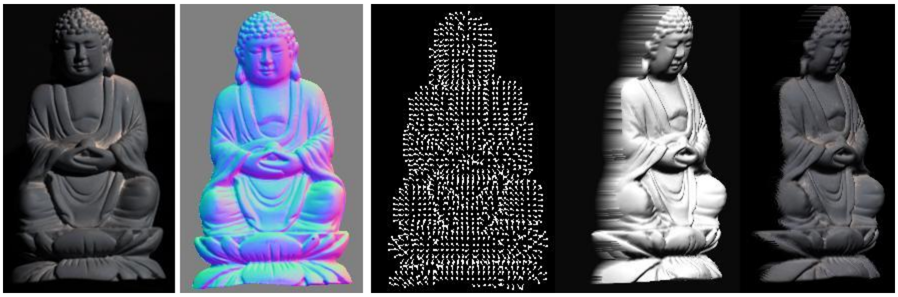


$$I_1 = k_d \mathbf{N} \cdot \mathbf{L_1}$$
$$I_2 = k_d \mathbf{N} \cdot \mathbf{L_2}$$
$$I_3 = k_d \mathbf{N} \cdot \mathbf{L_3}$$

Can write this as a matrix equation:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = k_d \begin{bmatrix} \mathbf{L_1}^T \\ \mathbf{L_2}^T \\ \mathbf{L_3}^T \end{bmatrix} \mathbf{N}$$

Example



Questions?

Good luck!