Perception / Computer Vision

"What's an object?"

3817674783595363744693879036326656034268381...
76747835953637446938790363266560342683 8 1...

METR 4202: Advanced Control & **Robotics**

Dr Surya Singh

Lecture # 7

September 6, 2013

metr4202@itee.uq.edu.au

http://itee.uq.edu.au/~metr4202/

RoboticsCourseWare.org

RoboticsCourseWare Contributor

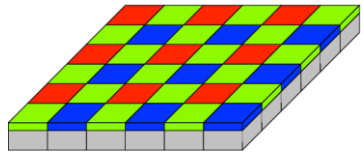
Schedule

Week	Date	Lecture (F: 9-10:30, 42-212)
1	26-Jul	Introduction
2	2-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	9-Aug	Robot Kinematics
4	16-Aug	Robot Dynamics & Control
5	23-Aug	Robot Trajectories & Motion
6	30-Aug	Sensors & Measurement
7	**6-Sep**	**Perception / Computer Vision**
8	13-Sep	Localization and Navigation
9	20-Sep	State-Space Modelling
	27-Sep	State-Space Control
10	4-Oct	*Study break*
11	11-Oct	Motion Planning
12	18-Oct	Vision-based control (+ Prof. P. Corke or Prof. M. Srinivasan)
13	25-Oct	Applications in Industry (+ Prof. S. LaValle) & Course Review

Features

- Colour
- Corners
- Edges
- Lines
- Statistics on Edges: SIFT, SURF

Features -- Colour Features



Bayer Patterns

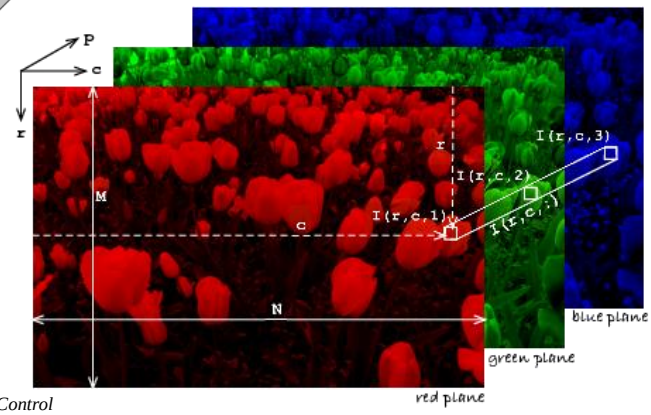
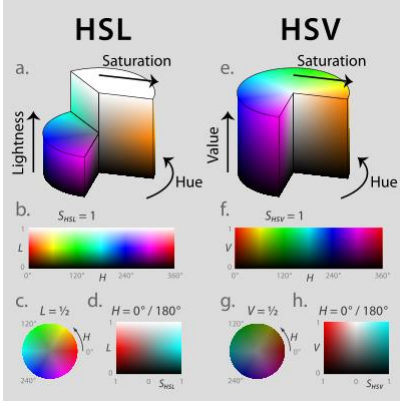


Fig: Ch. 10, *Robotics Vision and Control*

Colour Spaces

- HSV



Source: Wikipedia – HSV and YCrCb

- YCrCb

➔ Gamma Corrected Luma (Y) + Chrominance

➔ BW → Colour TVs : Just add the Chrominance

➔ γ Correction: CRTs γ=2.2-2.5

$$\begin{aligned} Y' &= 16 + (65.481 \cdot R' + 128.553 \cdot G' + 24.966 \cdot B') \\ C_B &= 128 + (-37.797 \cdot R' - 74.203 \cdot G' + 112.0 \cdot B') \\ C_R &= 128 + (112.0 \cdot R' - 93.786 \cdot G' - 18.214 \cdot B') \end{aligned}$$

Edge Detection

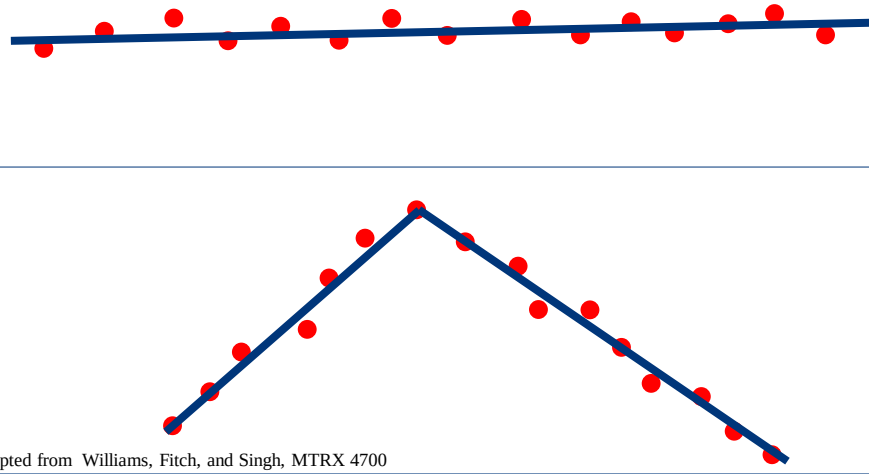
- Canny edge detector:

Fig: Ch. 10, *Robotics Vision and Control*

Edge Detection

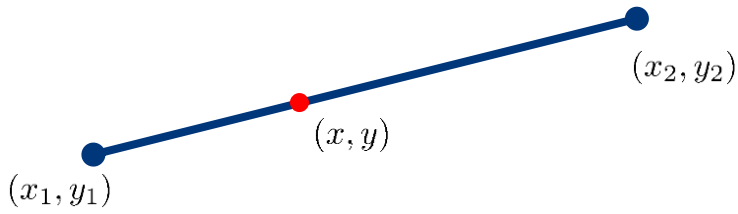
- Canny edge detector:

Line Extraction and Segmentation



Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Formula



$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\mathbf{y} = m\mathbf{x} + b$$

Adopted from Williams, Fitch, and Singh, MTRX 4700

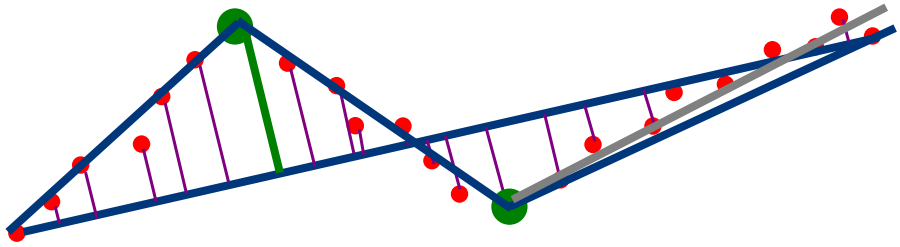
Line Estimation

Least squares minimization of the line:

- Line Equation: $\mathbf{y} - m\mathbf{x} - b = 0$
- Error in Fit: $\sum_i (y_i - mx_i - b)^2$
- Solution: $\begin{pmatrix} \bar{x}\bar{y} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} \overline{x^2} & \bar{x} \\ \bar{x} & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix}$

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Splitting / Segmentation



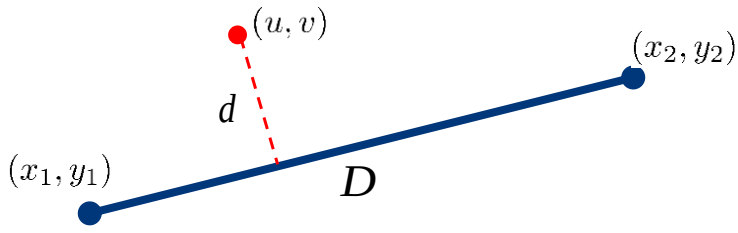
- What about corners?

➔Split into multiple lines (via expectation maximization)

1. Expect (assume) a number of lines N (say 3)
2. Find "breakpoints" by finding nearest neighbours upto a threshold or simply at random (RANSAC)
3. How to know N? (Also RANSAC)

Adopted from Williams, Fitch, and Singh, MTRX 4700

⊥ of a Point from a Line Segment

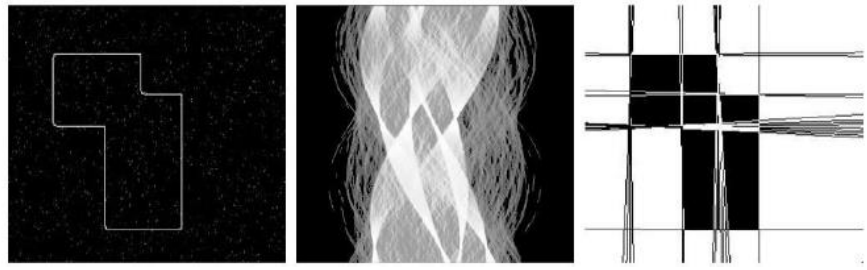


$$r = u(y_1 - y_2) + v(x_2 - x_1) + y_2x_1 - y_1x_2$$

$$d = \frac{r}{D}$$

Adopted from Williams, Fitch, and Singh, MTRX 4700

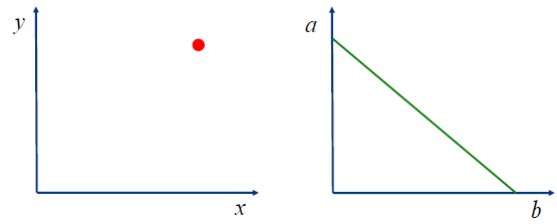
Hough Transform



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

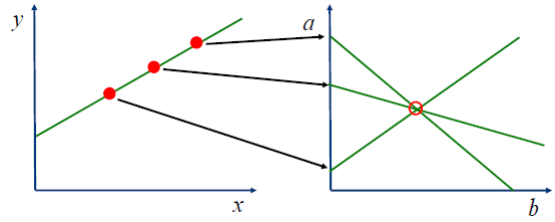
Hough Transform: Voting Space

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



- 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.

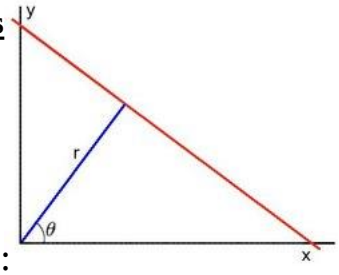
2. Assume that each cell in the parameter space is an accumulator. Initialize all cells to zero.

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:

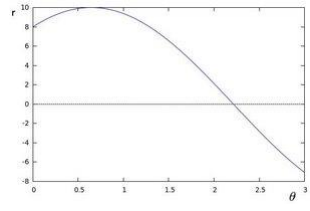
$$r = x_1\cos\theta + y_1\sin\theta$$

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

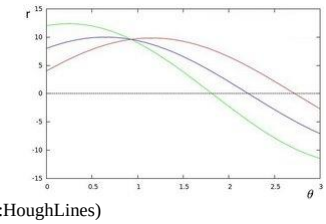
See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [2]

- Thus a given point gives a sinusoid



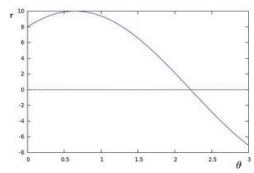
- Repeating for all points on the image

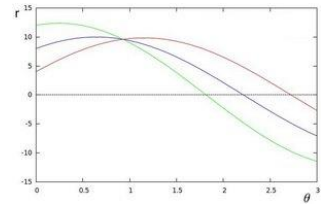


See also OpenCV documentation (cv::HoughLines)

Line Detection – Hough Lines [3]

- Thus a given point gives a sinusoid





- Repeating for all points on the image

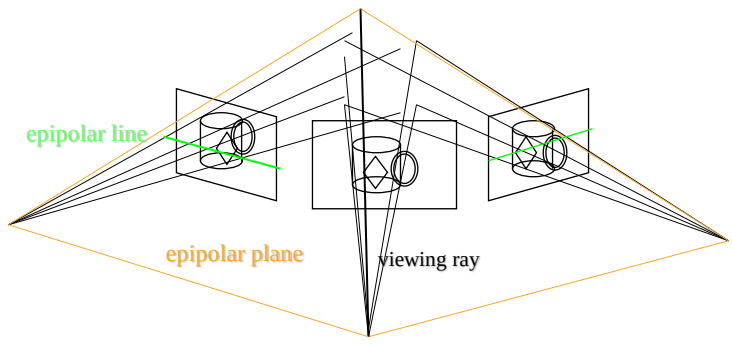
- NOTE that an intersection of sinusoids represents (**a point**) represents **a line** in which pixel points lay.

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

See also OpenCV documentation (cv::HoughLines)

Stereo: Epipolar geometry

- Match features along epipolar lines



Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Stereo: epipolar geometry

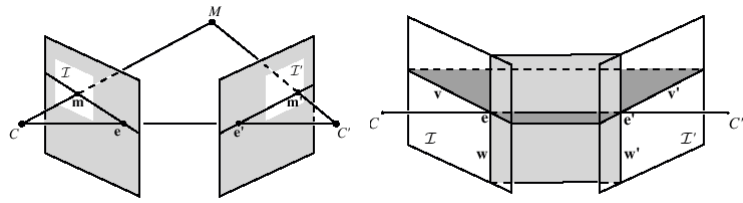
- for two images (or images with collinear camera centers), can find epipolar lines
- epipolar lines are the projection of the pencil of planes passing through the centers

- Rectification: warping the input images (perspective transformation) so that epipolar lines are horizontal

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Rectification

- Project each image onto same plane, which is parallel to the epipole
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion



- [Zhang and Loop, MSR-TR-99-21]

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Rectification

(a) Original image pair overlayed with several epipolar lines.

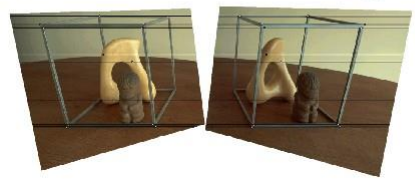
(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Rectification

(c) Image pair transformed by the similarity $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).



(d) Final image rectification after shearing transform $\mathbf{H}_a$ and $\mathbf{H}'_a$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Matching criteria

- Raw pixel values (correlation)
- Band-pass filtered images [Jones & Malik 92]
- "Corner" like features [Zhang, …]
- Edges [many people…]
- Gradients [Seitz 89; Scharstein 94]
- Rank statistics [Zabih & Woodfill 94]

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Finding correspondences

- Apply feature matching criterion (e.g., correlation or Lucas-Kanade) at all pixels simultaneously
- Search only over epipolar lines (many fewer candidate positions)

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

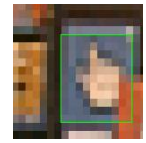
Image registration (revisited)

- How do we determine correspondences?
  - block matching or SSD (sum squared differences)

d is the disparity (horizontal motion)

$$E(x,y;d) = \sum_{(x',y')\in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$

- How big should the neighborhood be?

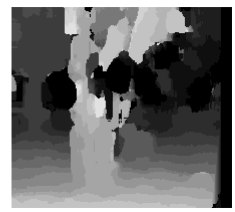
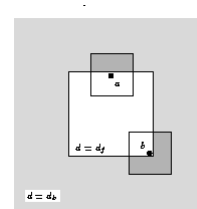



Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Neighborhood size

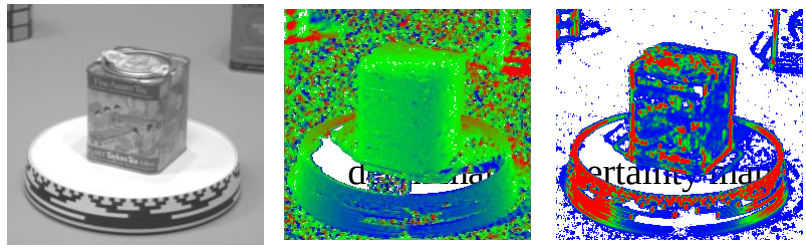
- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes

W

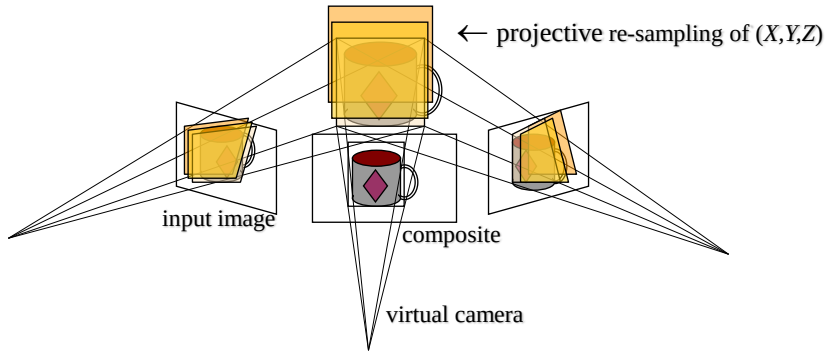
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Stereo: certainty modeling

- Compute certainty map from correlations
- 

Plane Sweep Stereo

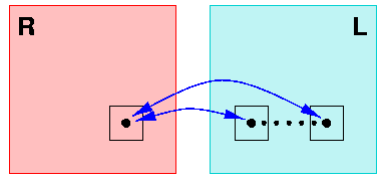
- Sweep family of planes through volume

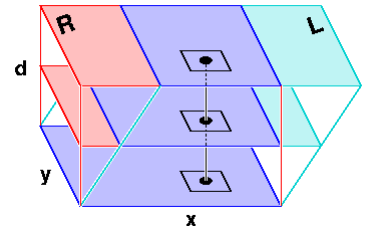


- each plane defines an image ⇒ composite homography

Plane sweep stereo

- Re-order (pixel / disparity) evaluation loops





for every pixel,
for every disparity
compute cost

for every disparity
for every pixel
compute cost

Stereo matching framework

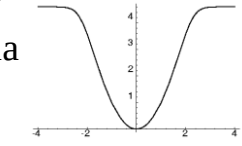
- For every disparity, compute raw matching costs

Why use a robust function?

– occlusions, other outliers

$$E_0(x, y; d) = \rho(I_L(x' + d, y') - I_R(x', y'))$$

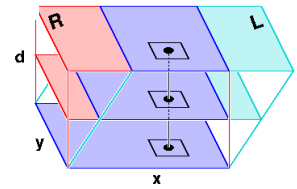
- Can also use alternative match criteria



Stereo matching framework

- Aggregate costs spatially

- Here, $E(x, y; d) = \sum_{(x',y') \in N(x,y)} E_0(x', y', d)$ (efficient moving average implementation)
- Can also use weighted average, [non-linear] diffusion…

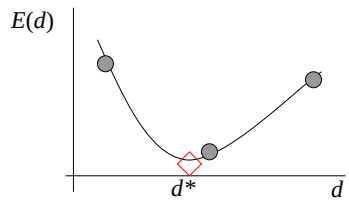


Stereo matching framework

- Choose winning disparity at each pixel

$$d(x, y) = \arg\min_{d} E(x, y; d)$$

- Interpolate to sub-pixel accuracy



Traditional Stereo Matching

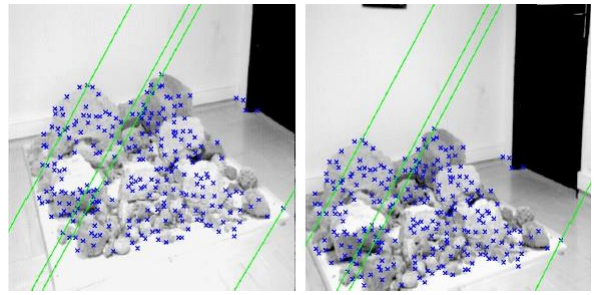
- Advantages:
  - gives detailed surface estimates
  - fast algorithms based on moving averages
  - sub-pixel disparity estimates and confidence
- Limitations:
  - narrow baseline $\Rightarrow$ noisy estimates
  - fails in textureless areas
  - gets confused near occlusion boundaries

Stereo with Non-Linear Diffusion

- Problem with traditional approach:
  - gets confused near discontinuities
- New approach:
  - use iterative (non-linear) aggregation to obtain better estimate
  - provably equivalent to mean-field estimate of Markov Random Field

Feature-based stereo

- Match “corner” (interest) points



- Interpolate complete solution

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Cool Robotics Share

D. Wedge, *The Fundamental Matrix Song*

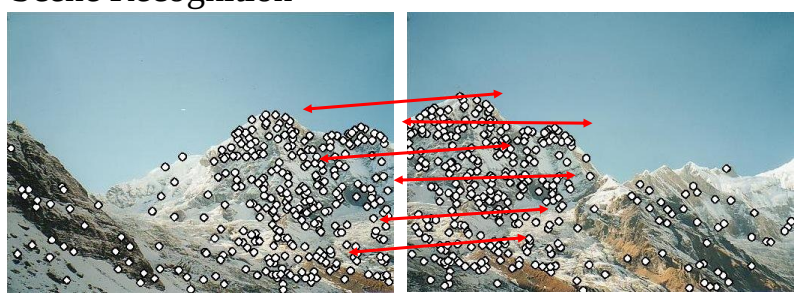
Edge Detection

- Canny edge detector:
  - Pepsi Sequence:

Image Data: http://www.cs.brown.edu/~black/mixtureOF.html and Szeliski, CS223B-L9
See also: Use of Temporal information to aid segmentation:
http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html

Why extract features?

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



- Steps:
  - Extract Features
  - Match Features

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

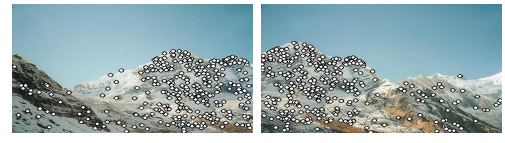
Why extract features? [2]

- Panorama stitching…
  → Step 3: Align images
  (see: Hartley & Zisserman,
  Multiple View Geometry)

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

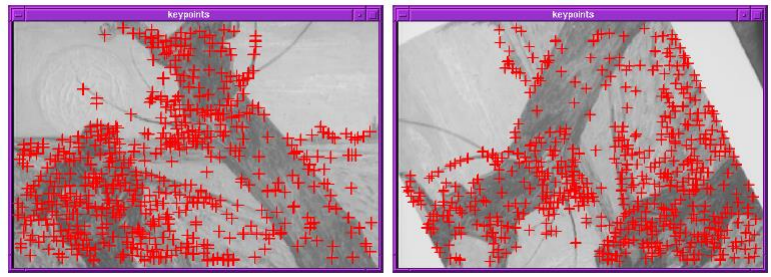
Characteristics of good features

- 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



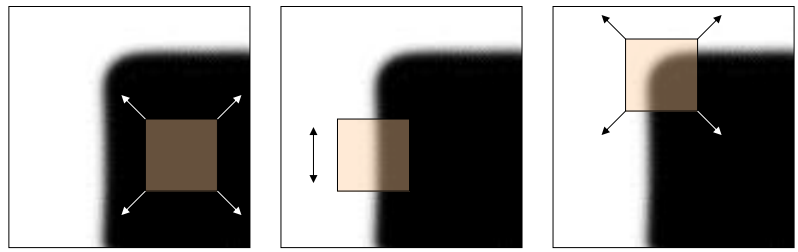
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

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

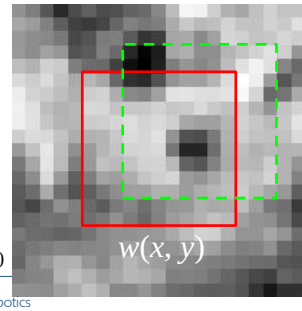
Source: A. Efros

Corner Detection: Mathematics

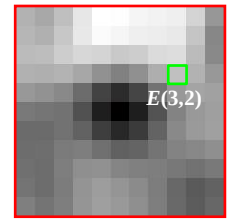
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$



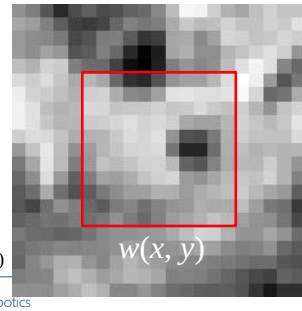
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Corner Detection: Mathematics

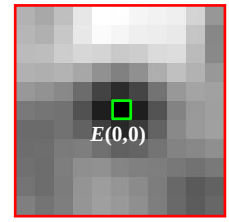
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$



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

Corner Detection: Mathematics

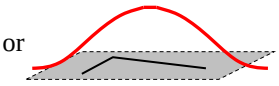
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

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)

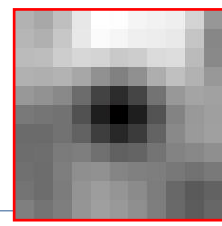
Corner Detection: Mathematics

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):

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

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

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

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

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

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

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

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.

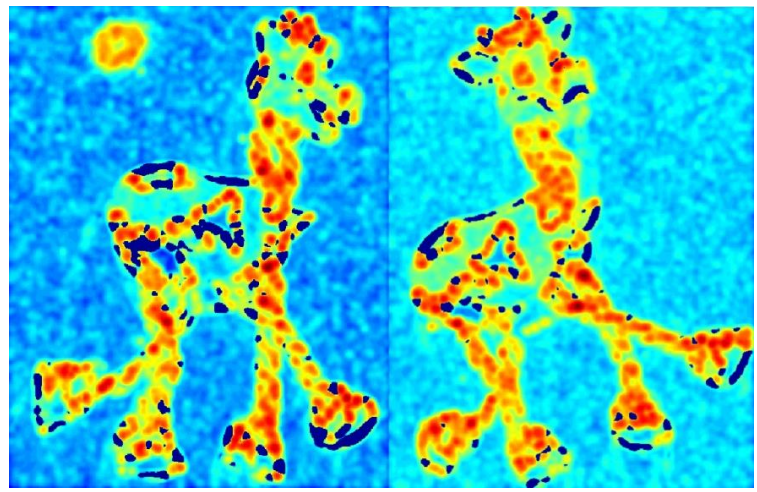
Adopted from
S. Lazebnik,
Gang Hua (CS 558)

Harris Detector: Steps

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

Harris Detector: Steps

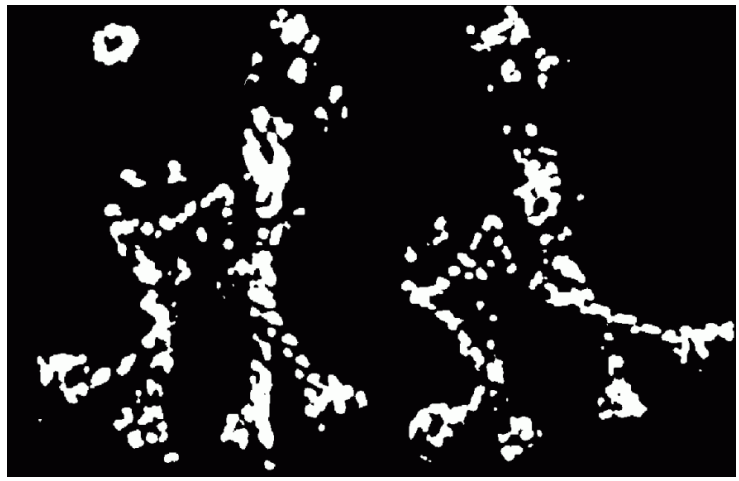
Compute corner response *R*



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

Harris Detector: Steps

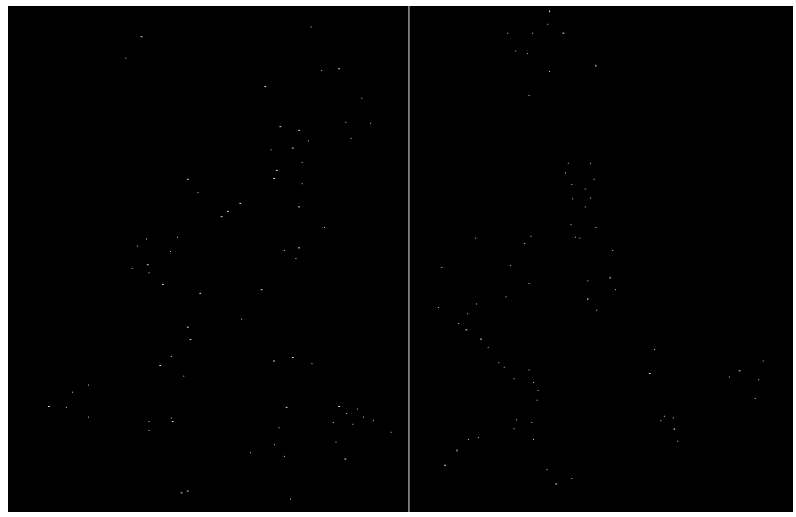
Find points with large corner response: R>threshold



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)

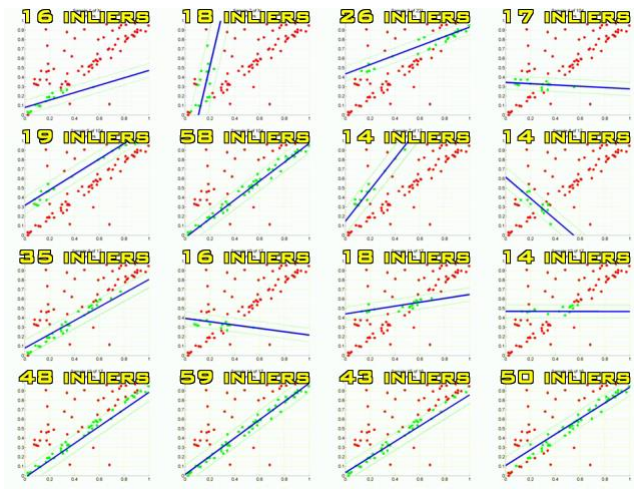
RANdom SAmple Consensus

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*

Cool Robotics Share (this week)

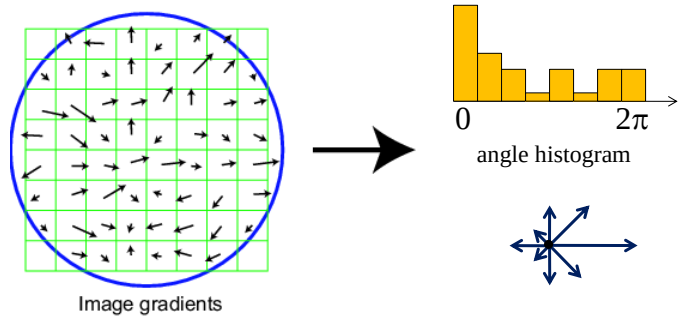


D. Wedge, *The RANSAC Song*

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

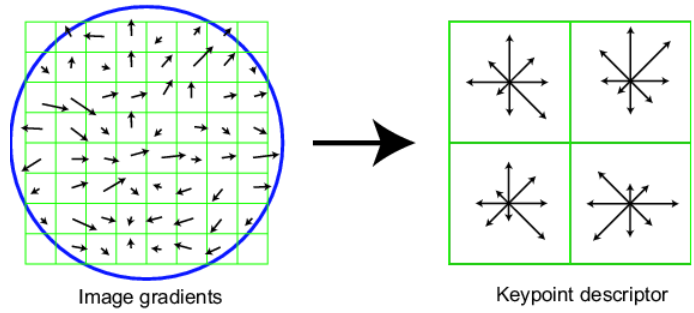


Adapted from slide by David Lowe

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*

Cool Robotics Share

Source: Youtube: Wired, How the Tesla Model S is Made