# Perception / Computer Vision

“What’s an object?”

METR 4202: Advanced Control & Robotics
Dr Surya Singh
Lecture # 7
September 6, 2013

---

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

---

## Schedule

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

---

© 2013 School of Information Technology and Electrical Engineering at the University of Queensland

---

TexPoint fonts used in EMF. Read the TexPoint manual before you delete this box.: A A A A A

---

© 2013 School of Information Technology and Electrical Engineering at the University of Queensland

---

RoboticsCourseWare.org

---

© 2013 School of Information Technology and Electrical Engineering at the University of Queensland
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
- YCrCb

\[ \text{Gamma Corrected Luma (Y) + Chrominance} \]

\[ \text{BW} \rightarrow \text{Colour TVs: Just add the Chrominance} \]

\[ \gamma \text{ Correction: CRTs } \gamma = 2.2 - 2.5 \]

\[
\begin{align*}
Y' &= 16 + (65.481 \cdot N + 128.53 \cdot G' + 24.966 \cdot B') \\
C_B &= 128 + (-37.797 \cdot N - 74.203 \cdot G' + 112.0 \cdot B') \\
C_N &= 128 + (112.0 \cdot N - 93.786 \cdot G' - 18.214 \cdot B')
\end{align*}
\]

Source: Wikipedia - HSV and YCrCb

Edge Detection

- Canny edge detector:

Fig: Ch. 10, Robotics Vision and Control
Edge Detection

- Canny edge detector:

Adopted from Williams, Fitch, and Singh. MTRX 4700

Line Extraction and Segmentation
Line Formula

Adopted from Williams, Fitch, and Singh, MTRX 4700

Line Estimation

Adopted from Williams, Fitch, and Singh, MTRX 4700
Line Splitting / Segmentation

- What about corners?
  \[ \text{Split into multiple lines (via expectation maximization)} \]
  1. Expect (assume) a number of lines \( N \) (say 3)
  2. Find “breakpoints” by finding nearest neighbours up to a threshold or simply at random (RANSAC)
  3. How to know \( N \)? (Also RANSAC)

Adopted from Williams, Fitch, and Singh, MTRX 4700

\[ r = u(y_1 - y_2) + v(x_2 - x_1) + y_2x_1 - y_1x_2 \]
\[ d = \frac{r}{D} \]

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

- 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)
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
    ✓ avoids problems with vert. lines

\[ y = mx + b \]

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

• 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 \textit{(a point)} represents \textbf{a line} in which pixel points lay.

\[ \Rightarrow \text{Thus, a line can be } \textit{detected} \text{ by finding the number of Intersections between curves} \]

See also OpenCV documentation (cv::HoughLines)

Stereo: Epipolar geometry

- Match features along epipolar lines

Slide from Szeliski, \textit{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

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]
Rectification

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

(b) Image pair transformed by the specialized perspective mapping \( H_e \) and \( H_e' \). 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 \( H_s \) and \( H_s' \). Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

(d) Final image rectification after shearing transform \( H_s' \) and \( H_e' \). Note that the image pair remains rectified, but the horizontal distortion is induced.

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]

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)
Image registration (revisited)

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

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

- How big should the neighborhood be?

Neighborhood size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes
Stereo: certainty modeling

- Compute certainty map from correlations

Plane Sweep Stereo

- Sweep family of planes through volume

Each plane defines an image $\Rightarrow$ composite homography
Plane sweep stereo

- Re-order (pixel / disparity) evaluation loops

  for every pixel, for every disparity
  compute cost

for every disparity
compute cost

Stereo matching framework

- For every disparity, compute raw matching costs

  Why use a robust function?
  - occlusions, other outliers

  \[ E_0(x, y; d) = \rho(I_L(x' + d, y') - I_D(x', y')) \]

  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...

\[
d(x, y) = \arg \min_d E(x, y; d)
\]

- 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*
Edge Detection

- Canny edge detector:
  - Pepsi Sequence:

Image Data: [http://www.cs.brown.edu/~black/mixtureOF.html](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](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 from 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


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
Corner Detection: Mathematics

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

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

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 $w(x,y) = \begin{cases} 
1 \text{ in window, } 0 \text{ outside} \\
\text{Gaussian}
\end{cases}$

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

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 $E(u, v) = E(0, 0) + [u \cdot \frac{\partial E(0, 0)}{\partial u} + \frac{1}{2} [u \cdot \frac{\partial^2 E(0, 0)}{\partial u^2} + v \cdot \frac{\partial^2 E(0, 0)}{\partial u \partial v} + v \cdot \frac{\partial^2 E(0, 0)}{\partial v^2}]]$

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

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

$$E(u, v) = E(0, 0) + [u \cdot \frac{\partial E(0, 0)}{\partial u} + \frac{1}{2} [u \cdot \frac{\partial^2 E(0, 0)}{\partial u^2} + v \cdot \frac{\partial^2 E(0, 0)}{\partial u \partial v} + v \cdot \frac{\partial^2 E(0, 0)}{\partial v^2}]]$$

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

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

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

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 \((0,0)\):

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

\[
E(0,0) = 0 \\
E_u(0,0) = 0 \\
E_v(0,0) = 0 \\
E_{uv}(0,0) = \sum_{x, y} 2w(x, y) I_x(x, y) I_y(x, y) \\
E_{xx}(0,0) = \sum_{x, y} 2w(x, y) I_{xx}(x, y) \\
E_{yy}(0,0) = \sum_{x, y} 2w(x, y) I_{yy}(x, y) \\
E_{xy}(0,0) = \sum_{x, y} 2w(x, y) I_x(x, y) I_y(x, y)
\]

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.”

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: $R > \text{threshold}$

Adopted from S. Lazebnik, Gang Hua (CS 558)
Harris Detector: Steps

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
**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 $\text{med}(|e|)$
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*
**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

![Image of Scale Invariant Feature Transform](image)

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

![Image of SIFT descriptor](image)

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

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