Beginner’s Guide to Image Processing

or

“How to make vision-guided killbots, step 1”

viral infection edition

8 April 2014

University of Queensland
The Blorch

- Paul has contracted ‘the Blorch’
  - Bad case of insides-fall-out syndrome
  - Surya will guest lecture today
  - Paul’s notes to Surya will take the form of Blorchnotes on the slide
But first…

Some house keeping
House keeping

• Rubbish and drink bottles in the lab
  – Warning 1. If you see someone making/leaving a mess in the lab, politely excoriariate them.
  – Lab is No Food Zone. Eat outside plz kthxbye. *Note: Make sure they know we aren’t kidding about this*

• Progress Reviews are done
  – (Almost) everyone passed; some barely
  – Wake-up call for a lot of people
  – We are seeing lots of hacking, very little math
House keeping

• Group 10 is no more; Group 9 is now bigger
  – Should have no advantage or disadvantage

• On PCBs:
  – Submit to Steve Wright for BEC batch runs
  – If you want to use a specific fab (e.g. for fast turnaround) talk to ETSG – they will allow you to place the order yourself and get reimbursed
  – Ask John Kolbach first.

Blorchnote: If students are uncertain, have them ask John
House keeping

Reminder:

• You must submit your designs to Doug for machining parts no later than week 7
  – No parts will be machined for you after then
  – You can machine your own parts, but you won’t be able to go through the workshop
# Calendar at a glance

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Lecture</th>
<th>Reviews</th>
<th>Demos</th>
<th>Assessment submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3/3 – 7/3</td>
<td>Introduction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10/3 – 15/3</td>
<td>Principles of Mechatronic Systems design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>17/3 – 21/3</td>
<td>Professional Engineering Topics</td>
<td></td>
<td></td>
<td>Design brief</td>
</tr>
<tr>
<td>4</td>
<td>24/3 – 28/3</td>
<td>Your soldering is (probably) terrible</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>31/3 – 4/3</td>
<td>No lecture</td>
<td>Progress review 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7/4 – 11/4</td>
<td>Image segmentation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14/4 – 18/4</td>
<td>By request</td>
<td>Progress seminar</td>
<td>25% demo</td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td>21/4 – 25/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>28/4 – 3/5</td>
<td>By request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5/5 – 9/5</td>
<td>By request</td>
<td>Progress review</td>
<td>50% demo</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12/5 – 16/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>19/5 – 23/5</td>
<td></td>
<td>75% demo</td>
<td>Preliminary report</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>26/5 – 30/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2/6 – 6/6</td>
<td>Closing lecture</td>
<td>Final testing</td>
<td></td>
<td>Final report and addendum</td>
</tr>
</tbody>
</table>
Progress seminars

• Progress seminars are next week!
  – First group-based assessment
  – Gives you presenting experience and brings us up to date with your team’s progress

• Sign up for session slots via Doodle poll
  – Link to poll will be sent out via Blackboard announcement after the lecture (closes Friday)

**Blochnote: Same procedure as last time... nothing scary**
Progress Seminar

• Group presentation – 10 minutes per team
  – Stand up and talk about your progress
  – Each person talks for roughly equal time

• Focus on progress, not the requirements!
  – We already know what the project goal is.
  – We know (roughly) what your approach is.
  – Don’t waste valuable time repeating them.
  – Above all, show evidence of your work.

*Blorchnote: The Blorch just loves evidence and justification*
Progress seminars

• How to sign up:
  – Have **one and only one** member of your team nominate a time for your team on the
  – When they sign up, they must include their **full name and team number**. If they don’t have both, the slot will be cleared.

• If you absolutely can’t get a slot that works for all of your group, email me ASAP
  – But this should never happen
FAQ Roundup

• Can we use off the shelf tires, or do we have to make our own?
  – Off the shelf tires are fine, so long as you aren’t using a rubber wheel that is 100% tire.

• Can we use Lego shafts?
  – Sure, but why would you want to?

• Can we use Arduino/RaspPi/Raspduino?
  – Yes… but I won’t respect you in the morning.

• Wasn’t there supposed to be a doodle poll for lecture topics?
  – Yeah, about that…
Lecture nominations

• Only one person nominated a topic
  – This presentation is therefore on that topic.

• Future topic polls will be conducted in class
  – Hey, let’s do that now!

Blorchnote: Please take an informal poll of the students as to what topics they would like to hear about – if no consensus, let them know that coffee+Q&A is often very helpful, too... nice if the topics are easy to teach!
Demos

• The first preliminary demonstration round runs next week
  – Mark cap at 25 per cent.
  – One demo attempt per team (in this round)

• If you are ready to test your car, contact me now so I can arrange things.
  – Bookings must be made by 5pm Friday.
Back to business…

And now a special guest…
Perception / Computer Vision

“What’s an object?”

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

  ➣ Gamma Corrected Luma (Y) + Chrominance

  ➣ BW ➔ Colour TVs: Just add the Chrominance

  ➣ $\gamma$ Correction: CRTs $\gamma=2.2-2.5$

\[
Y' = 16+ (65.481 \cdot R' + 128.553 \cdot G' + 24.966 \cdot B')
\]
\[
C_B = 128+ (-37.797 \cdot R' + 74.203 \cdot G' + 112.0 \cdot B')
\]
\[
C_R = 128+ (112.0 \cdot R' - 93.786 \cdot G' - 18.214 \cdot B')
\]

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

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

\[ y = mx + b \]

Adopted from Williams, Fitch, and Singh, MTRX 4700
Line Estimation

Least squares minimization of the line:

- **Line Equation:** \( y - mx - b = 0 \)

- **Error in Fit:** \( \sum_i (y_i - mx_i - b)^2 \)

- **Solution:**

\[
\begin{pmatrix}
\bar{x}
\bar{y}
\end{pmatrix}
= \begin{pmatrix}
\bar{x}^2 & \bar{x} & 1
\end{pmatrix}
\begin{pmatrix}
m \\
b
\end{pmatrix}
\]

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
Let \( \perp \) 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

\[ y = ax + b \]

\[ a = -\frac{1}{x}b + \frac{y}{x} \]

- 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)
• 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: \( r, \theta \)
    \( \Rightarrow \) 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 \((r, \theta)\) 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

![Graph of r vs θ]

• Repeating for all points on the image

![Graph of r vs θ]

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]
Rectification

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

(c) Image pair transformed by the similarity $H_\theta$ and $H'_\theta$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

(d) Final image rectification after shearing transform $H_\phi$ and $H'_\phi$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!
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)

Slide from Szeliski, *Computer Vision: Algorithms and Applications*
Image registration (revisited)

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

\[ d \text{ is the disparity (horizontal motion)} \]

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

• 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
Stereo: certainty modeling

- Compute certainty map from correlations
Plane Sweep Stereo

- Sweep family of planes through volume

\[
\text{input image} \quad \Rightarrow \text{virtual camera} \quad \Rightarrow \text{projective re-sampling of } (X,Y,Z) \quad \Rightarrow \text{composite homography}
\]

- 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, 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

\[ E(x, y; d) = \sum_{(x', y') \in N(x, y)} E_0(x', y', d) \]

(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
D. Wedge, *The Fundamental Matrix Song*
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 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


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 $[u,v]$:

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

Adopted from S. Lazebnik, Gang Hua (CS 558)
Corner Detection: Mathematics

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

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

Adopted from S. Lazebnik, Gang Hua (CS 558)
Corner Detection: Mathematics

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

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

Window function

Shifted intensity

Intensity

Window function \( w(x,y) = \)

1 in window, 0 outside

or

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 $[u,v]$:

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

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} [u \ v]$$

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

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

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

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

\( E_u (u, v) = \sum_{x, y} 2w(x, y) I_x (x + u, y + v) - I(x, y) \bar{I}_x(x + u, y + v) \)

\( E_{uu} (u, v) = \sum_{x, y} 2w(x, y) I_x(x + u, y + v)I_x(x + u, y + v) \)

\[
+ \sum_{x, y} 2w(x, y) I(x + u, y + v) - I(x, y) \bar{I}_{xx}(x + u, y + v)
\]

\( E_{uv} (u, v) = \sum_{x, y} 2w(x, y) I_y(x + u, y + v)I_x(x + u, y + v) \)

\[
+ \sum_{x, y} 2w(x, y) I(x + u, y + v) - I(x, y) \bar{I}_{xy}(x + u, y + v)
\]

Adopted from S. Lazebnik, Gang Hua (CS 558)
Corner Detection: Mathematics

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

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

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

\[ E(0,0) = 0 \]
\[ E_{u}(0,0) = 0 \]
\[ E_{v}(0,0) = 0 \]
\[ E_{uu}(0,0) = \sum_{x,y} 2w(x, y) I_x(x, y) I_x(x, y) \]
\[ E_{vv}(0,0) = \sum_{x,y} 2w(x, y) I_y(x, y) I_y(x, y) \]
\[ E_{uv}(0,0) = \sum_{x,y} 2w(x, y) I_x(x, y) I_y(x, y) \]

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

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


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: $R > \text{threshold}$

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

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

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

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

From Szeliski, *Computer Vision: Algorithms and Applications*
Feature distance

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

From Szeliski, *Computer Vision: Algorithms and Applications*
Feature distance

• How to define the difference between two features $f_1$, $f_2$?
  – Better approach: ratio distance = $\frac{\text{SSD}(f_1, f_2)}{\text{SSD}(f_1, f_2')}$
  
  • $f_2$ is best SSD match to $f_1$ in $I_2$
  • $f_2'$ is 2$^{nd}$ best SSD match to $f_1$ in $I_2$
  • gives small values for ambiguous matches

From Szeliski, *Computer Vision: Algorithms and Applications*
Evaluating the results

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

From Szeliski, *Computer Vision: Algorithms and Applications*
True/false positives

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

From Szeliski, *Computer Vision: Algorithms and Applications*
Camera calibration

• Determine camera parameters from known 3D points or calibration object(s)

• internal or intrinsic parameters such as focal length, optical center, aspect ratio: what kind of camera?

• external or extrinsic (pose) parameters: where is the camera?

• How can we do this?

From Szeliski, *Computer Vision: Algorithms and Applications*
Camera calibration – approaches

• Possible approaches:
  – linear regression (least squares)
  – non-linear optimization
  – vanishing points
  – multiple planar patterns
  – panoramas (rotational motion)

From Szeliski, *Computer Vision: Algorithms and Applications*
Image formation equations

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
\end{bmatrix}
= \begin{bmatrix}
R
\end{bmatrix}_{3\times3}
\begin{bmatrix}
X \\
Y \\
Z \\
\end{bmatrix}
+ t
\]

\[
\begin{bmatrix}
u \\
v \\
1 \\
\end{bmatrix}
\sim
\begin{bmatrix}
U \\
V \\
W \\
\end{bmatrix}
= \begin{bmatrix}
f & 0 & u_c \\
0 & f & v_c \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
\end{bmatrix}
\]

From Szeliski, *Computer Vision: Algorithms and Applications*
Calibration matrix

- Is this form of K good enough?
- non-square pixels (digital video)
- skew
- radial distortion

\[
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix} \sim \begin{bmatrix}
f & 0 & u_c \\
0 & f & v_c \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix} = K \begin{bmatrix}
X_c
\end{bmatrix}
\]

\[
\begin{bmatrix}
f a & s & u_c \\
0 & f & v_c \\
0 & 0 & 1
\end{bmatrix} = K
\]

From Szeliski, *Computer Vision: Algorithms and Applications*
Levenberg-Marquardt

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

\[
\hat{u}_i = f(m, x_i) + \frac{\partial f}{\partial m} \Delta m
\]

\[
\hat{v}_i = g(m, x_i) + \frac{\partial g}{\partial m} \Delta m
\]

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

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

From Szeliski, *Computer Vision: Algorithms and Applications*
Levenberg-Marquardt

- What if it doesn’t converge?
  - Multiply diagonal by \((1 + l)\), increase \(l\) until it does
  - Halve the step size \(D_m\) (my favorite)
  - Use line search
  - Other ideas?

- Uncertainty analysis: covariance \(S = A^{-1}\)

- Is maximum likelihood the best idea?

- How to start in vicinity of global minimum?

From Szeliski, *Computer Vision: Algorithms and Applications*
Camera matrix calibration

• Advantages:
  – very simple to formulate and solve
  – can recover $K [R \mid t]$ from $M$ using QR decomposition [Golub & VanLoan 96]

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

From Szeliski, *Computer Vision: Algorithms and Applications*
Multi-plane calibration

- Use several images of planar target held at unknown orientations [Zhang 99]
  - Compute plane homographies
    \[
    \begin{bmatrix}
    u_i \\
    v_i \\
    1
    \end{bmatrix}
    \sim
    \begin{bmatrix}
    r_1 & r_2 & t
    \end{bmatrix}
    \begin{bmatrix}
    x_i \\
    y_i \\
    1
    \end{bmatrix}
    \sim
    HX
    \]
  - Solve for K-TK-1 from Hk’s
    - 1 plane if only f unknown
    - 2 planes if (f,uc,vc) unknown
    - 3+ planes for full K
  - Code available from Zhang and OpenCV

From Szeliski, *Computer Vision: Algorithms and Applications*
Rotational motion

• Use pure rotation (large scene) to estimate f
  – estimate f from pairwise homographies
  – re-estimate f from 360° “gap”
  – optimize over all \( \{K, R_j\} \) parameters
    [Stein 95; Hartley ’97; Shum & Szeliski ’00; Kang & Weiss ’99]

  ![f=510](image1) ![f=468](image2)

• Most accurate way to get f, short of surveying distant points

From Szeliski, *Computer Vision: Algorithms and Applications*
SFM: Structure from Motion
(& Cool Robotics Share (this week))
Structure [from] Motion

• Given a set of feature tracks, estimate the 3D structure and 3D (camera) motion.

• Assumption: orthographic projection

• Tracks: \((u_{fp}, v_{fp}), f: \text{frame}, p: \text{point}\)

• Subtract out mean 2D position...

\[ u_{fp} = \dot{i}_f^T s_p, \quad v_{fp} = \dot{j}_f^T s_p \]

From Szeliski, *Computer Vision: Algorithms and Applications*
Structure from motion

• How many points do we need to match?

• 2 frames:
  – \((R,t)\): 5 dof + 3n point locations \leq 4n point measurements \Rightarrow n \geq 5
  – n \geq 5

• k frames:
  – 6(k–1)-1 + 3n \leq 2kn

• always want to use many more

From Szeliski, *Computer Vision: Algorithms and Applications*
Measurement equations

- Measurement equations
  \[ u_{fp} = i_f^T s_p \quad i_f: \text{rotation}, \ s_p: \text{position} \]
  \[ v_{fp} = j_f^T s_p \]

- Stack them up…
  \[ W = R S \]
  \[ R = (i_1, \ldots, i_F, j_1, \ldots, j_F)^T \]
  \[ S = (s_1, \ldots, s_P) \]

From Szeliski, *Computer Vision: Algorithms and Applications*
Factorization

\[ W = R_{2F \times 3} S_{3 \times P} \]

**SVD**

\[ W = U \Lambda V \quad \Lambda \text{ must be rank 3} \]

\[ W' = (U \Lambda^{1/2})(\Lambda^{1/2} V) = U' V' \]

Make \( R \) orthogonal

\[ R = QU', \quad S = Q^{-1}V' \]

\[ i_f^T Q^T Q i_f = 1 \ldots \]

From Szeliski, *Computer Vision: Algorithms and Applications*
Results

- Look at paper figures...

Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, *Computer Vision: Algorithms and Applications*
Questions
Tune-in next time for...

<Class voted topic here>!

_or_

“When we switch to Q&A, we can get coffee.”

Fun fact: Mathematical models predict the Blorch will overwhelm all of humanity some time in 2017