Image Processing

Cosimo Distante
Cosimo.distante@cnr.it

Lecture: Stereo
Inferring 3D from 2D

- Model based pose estimation
  - Single (calibrated) camera
    - Can determine the pose of the model
  - Known model

- Stereo vision
  - Two (calibrated) cameras
    - Can determine the positions of points in the scene
  - Arbitrary scene
    - Relative pose between cameras is also known
Stereo Process

- Extract features from the left and right images
- Match the left and right image features, to get their disparity in position (the “correspondence problem”)
- Use stereo disparity to compute depth (the reconstruction problem)

http://vision.middlebury.edu/stereo/data/scenes2003/

- The correspondence problem is the most difficult
Characteristics of Human Stereo Vision

- Matching features must appear similar in the left and right images

For example, we can’t fuse a left stereo image with a negative of the right image...
Example

Left image

Right image

Reconstructed surface with image texture
Characteristics of Human Stereo Vision

- Can only “fuse” objects within a limited range of depth around the fixation distance
- Vergence eye movements are needed to fuse objects over larger range of depths
Panum’s Fusional Area

• Panum's fusional area is the range of depths for which binocular fusion can occur (without changing vergence angles)

• It’s actually quite small ... we are able to perceive a wide range of depths because we are changing vergence angles

Figure 7. Haplopic method of determining the horopter involves locating the region of single binocular vision at a distance of 40cm. Panum's fusional area lies between the outer and inner limits of the region of single binocular vision.
Fixation, convergence

From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

Human eyes **fixate** on point in space – rotate so that corresponding images form in centers of fovea.
**Human stereopsis: disparity**

**Disparity** occurs when eyes fixate on one object; others appear at different visual angles.

Disparity is distance from $b_1$ to $b_2$ along retina.

From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

Adapted from David Forsyth, UC Berkeley
Stereo photography and stereo viewers

Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.

Invented by Sir Charles Wheatstone, 1838

Image from fisher-price.com
3D movies were popular in the 1950's. The left and right images were displayed as red and blue.
Stereo Displays

• Current technology for 3D movies and computer displays is to use polarized glasses
• The viewer wears eyeglasses which contain circular polarizers of opposite handedness

http://www.3dsgamenews.com/2011/01/3ds-to-feature-3d-movies/
Stereo vision

Two cameras, simultaneous views

Single moving camera and static scene
Why multiple views?

Structure and depth can be ambiguous from single views...

Images from Lana Lazebnik
Why multiple views?

Points at different depths along a line project to a single point.
Multiple views

Stereo vision
Structure from motion
Optical flow
Multi-view geometry problems

- **Stereo correspondence**: Given a point in one of the images, where could its corresponding points be in the other images?

Camera 1: \( R_1, t_1 \)

Camera 2: \( R_2, t_2 \)

Camera 3: \( R_3, t_3 \)

Slide credit: Noah Snavely
Multi-view geometry problems

- **Structure:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point.

Camera 1 \( R_1, t_1 \)

Camera 2 \( R_2, t_2 \)

Camera 3 \( R_3, t_3 \)
Multi-view geometry problems

- **Motion:** Given a set of corresponding points in two or more images, compute the camera parameters.
Multi-view geometry problems

- **Optical flow**: Given two images, find the location of a world point in a second close-by image with no camera info.
Estimating depth with stereo

- **Stereo**: shape from “motion” between two views
- We’ll need to consider:
  - Info on camera pose ("calibration")
  - Image point correspondences
Depth with stereo: basic idea
Depth with stereo: basic idea

Basic Principle: Triangulation
- Gives reconstruction as intersection of two rays
- Requires
  - camera pose (calibration)
  - *point correspondence*

Source: Steve Seitz
Camera calibration

Extrinsic parameters:
- Camera frame \( \leftrightarrow \) Reference frame

Intrinsic parameters:
- Image coordinates relative to camera \( \leftrightarrow \) Pixel coordinates

- **Extrinsic** params: rotation matrix and translation vector
- **Intrinsic** params: focal length, pixel sizes (mm), image center point, radial distortion parameters

We’ll assume for now that these parameters are given and fixed.
Example Calibration Patterns

- Geometry of target is known
- Pose of target is not known
Lens Distortion

• Lens distortion - projected points do not follow the simple pinhole camera formula

• Most common is barrel distortion and pin-cushion distortion
  – Points are displaced radially inward (barrel) or outward (pincushion) from correct position
  – Tangential distortions are also possible

• There are other types of lens aberrations that we won’t consider
  – Spherical, coma, astigmatism (these blur the point)
  – Chromatic aberrations (color affects focal length)
Radial Distortion Examples

wideangle (barrel)

telephoto (pincushion)
\[
\begin{pmatrix}
    x_d \\
    y_d
\end{pmatrix} =
\begin{pmatrix}
    x \\
    y
\end{pmatrix} (1 + k_1 r^2 + k_2 r^4 + k_5 r^6)
\]

where \( r = \text{distance from center}, \quad r^2 = x^2 + y^2 \)
\[ dx = \begin{bmatrix}
2k_3xy + k_4\left(r^2 + 2x^2\right) \\
k_3\left(r^2 + 2y^2\right) + 2k_4xy
\end{bmatrix} \]

Model of Tangential Distortion
Figure 11-5. Tangential distortion results when the lens is not fully parallel to the image plane; in cheap cameras, this can happen when the imager is glued to the back of the camera (image courtesy of Sebastian Thrun)
Geometry for a simple stereo system

• First, assuming parallel optical axes, known camera parameters (i.e., calibrated cameras):
Stereo Geometry – Simple Case

- Assume image planes are coplanar
- There is only a translation in the $X$ direction between the two coordinate frames
- $b$ is the baseline distance between the cameras

\[
x_L = f \frac{X_L}{Z_L}, \quad x_R = f \frac{X_R}{Z_R}
\]

\[\Rightarrow \quad x_L = f \frac{X_R + b}{Z}
\]

\[d = x_L - x_R = f \frac{(X_R + b) - X_R}{Z} = f \frac{b}{Z}
\]

\[\Rightarrow \quad Z = f \frac{b}{d}
\]

Disparity $d = x_L - x_R$
image point (left)

Focal length

optical center (left)

image point (right)

optical center (right)

World point

Depth of \( p \)

baseline

Focal length \( f \)

image point \( p_l \)

image point \( p_r \)
Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). We can triangulate via:

  Similar triangles \((p_l, P, p_r)\) and \((O_l, P, O_r)\):

  \[
  \frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}
  \]

  \[
  Z = f \frac{T}{x_r - x_l}
  \]

  disparity
Disparity example

\[
(x', y') = (x + D(x, y), y)
\]
What do we need to know?

1. Calibration for the two cameras.
   1. Intrinsic matrices for both cameras (e.g., f)
   2. Baseline distance T in parallel camera case
   3. R, t in non-parallel case

2. Correspondence for every pixel.
1. Calibration for the two cameras.

\[ x = K[R \ t]X \]

- \( x \): Image Coordinates: \((u, v, 1)\)
- \( K \): Intrinsic Matrix (3x3)
- \( R \): Rotation (3x3)
- \( t \): Translation (3x1)
- \( X \): World Coordinates: \((X, Y, Z, 1)\)

Think of \([R \ t]\) as the transformation from camera 1 to camera 2 in stereo system.
How to calibrate the camera?
(also called “camera resectioning”)

$$x = K[R \quad t]X$$

$$\begin{bmatrix}
su \\
sv \\
s
\end{bmatrix} = \begin{bmatrix}
* & * & * & * \\
* & * & * & * \\
* & * & * & * \\
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}$$
Least squares line fitting

• Data: \((x_1, y_1), \ldots, (x_n, y_n)\)
• Line equation: \(y_i = mx_i + b\)
• Find \((m, b)\) to minimize

\[
E = \sum_{i=1}^{n} (y_i - mx_i - b)^2
\]

\[
E = \sum_{i=1}^{n} \left( \begin{bmatrix} x_i \\ 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2
= \left\| \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2
= \left\| Ap - y \right\|^2
\]

\[
\]

\[
\frac{dE}{dp} = 2A^T Ap - 2A^T y = 0
\]

Matlab: \(p = A \backslash y\);

\[
A^T Ap = A^T y \implies p = (A^T A)^{-1} A^T y \quad \text{(Closed form solution)}
\]

Modified from S. Lazebnik
Example: solving for translation

\[
\begin{bmatrix}
  x_i^B \\
  y_i^B
\end{bmatrix} = \begin{bmatrix}
  x_i^A \\
  y_i^A
\end{bmatrix} + \begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]

Least squares setup

\[
\begin{bmatrix}
  1 & 0 \\
  0 & 1 \\
  \vdots & \vdots \\
  1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix} =
\begin{bmatrix}
  x_1^B - x_1^A \\
  y_1^B - y_1^A \\
  \vdots \\
  x_n^B - x_n^A \\
  y_n^B - y_n^A
\end{bmatrix}
\]
World vs Camera coordinates
Calibrating the Camera

Use an scene with **known** geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)
How do we calibrate a camera?

**Known 2d image coords**

| 880 | 214 |
| 43  | 203 |
| 270 | 197 |
| 886 | 347 |
| 745 | 302 |
| 943 | 128 |
| 476 | 590 |
| 419 | 214 |
| 317 | 335 |
| 783 | 521 |
| 235 | 427 |
| 665 | 429 |
| 655 | 362 |
| 427 | 333 |
| 412 | 415 |
| 746 | 351 |
| 434 | 415 |
| 525 | 234 |
| 716 | 308 |
| 602 | 187 |

**Known 3d world locations**

| 312.747 | 309.140 | 30.086 |
| 305.796 | 311.649 | 30.356 |
| 307.694 | 312.358 | 30.418 |
| 310.149 | 307.186 | 29.298 |
| 311.937 | 310.105 | 29.216 |
| 311.202 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
| 309.317 | 312.490 | 30.230 |
| 307.435 | 310.151 | 29.318 |
| 308.253 | 306.300 | 28.881 |
| 306.650 | 309.301 | 28.905 |
| 308.069 | 306.876 | 28.660 |
What is least squares doing?

• Given 3D point evidence, find best $M$ which minimizes error between estimate ($p'$) and known corresponding 2D points ($p$).

\[ p = \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ p = \text{distance from image center} \]
What is least squares doing?

• Best $M$ occurs when $p' = p$, or when $p' - p = 0$
• Form these equations from all point evidence
• Solve for model via closed-form regression

$p' \text{ under } M$

Error between $M$ estimate and known projection point

$p = \begin{bmatrix} u \\ v \end{bmatrix}$

$p = \text{distance from image center}$

$p' = \text{distance from known projection point}$

$p' - p = \text{error between model estimate and known projection point}$

$M = \begin{bmatrix} u & 0 \\ 0 & v \end{bmatrix}$

Camera center

$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
Known 3d locations

**Unknown Camera Parameters**

\[
\begin{bmatrix}
\begin{array}{cccc}
  s & u \\
  sv & \\
  s & \\
\end{array}
\end{bmatrix} = \begin{bmatrix}
  m_{11} & m_{12} & m_{13} & m_{14} \\
  m_{21} & m_{22} & m_{23} & m_{24} \\
  m_{31} & m_{32} & m_{33} & m_{34}
\end{bmatrix} \begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

- **Known 2d image coords**
- **Known 3d locations**

First, work out where \(X, Y, Z\) projects to under candidate \(M\).

\[
\begin{align*}
  su &= m_{11}X + m_{12}Y + m_{13}Z + m_{14} \\
  sv &= m_{21}X + m_{22}Y + m_{23}Z + m_{24} \\
  s &= m_{31}X + m_{32}Y + m_{33}Z + m_{34}
\end{align*}
\]

Two equations per 3D point correspondence

\[
\begin{align*}
  u &= \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}} \\
  v &= \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}
\end{align*}
\]
Unknown Camera Parameters

\[
\begin{bmatrix}
u \\
v \end{bmatrix} = \begin{bmatrix}
m_{11} & m_{12} & m_{13} & m_{14} \\
m_{21} & m_{22} & m_{23} & m_{24} \\
m_{31} & m_{32} & m_{33} & m_{34}
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

Next, rearrange into form where all \( M \) coefficients are individually stated in terms of \( X,Y,Z,u,v \).

\[
u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}
\]

\[
v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}
\]

\[
(m_{31}X + m_{32}Y + m_{33}Z + m_{34})u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}
\]

\[
(m_{31}X + m_{32}Y + m_{33}Z + m_{34})v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}
\]

\[
m_{31}uX + m_{32}uY + m_{33}uZ + m_{34}u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}
\]

\[
m_{31}vX + m_{32}vY + m_{33}vZ + m_{34}v = m_{21}X + m_{22}Y + m_{23}Z + m_{24}
\]
Known Camera Parameters

\[
\begin{bmatrix}
  su \\
  sv \\
  s
\end{bmatrix} =
\begin{bmatrix}
  m_{11} & m_{12} & m_{13} & m_{14} \\
  m_{21} & m_{22} & m_{23} & m_{24} \\
  m_{31} & m_{32} & m_{33} & m_{34}
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

Next, rearrange into form where all \( M \) coefficients are individually stated in terms of \( X, Y, Z, u, v \).

\[
m_{31} uX + m_{32} uY + m_{33} uZ + m_{34} u = m_{11} X + m_{12} Y + m_{13} Z + m_{14}
\]
\[
m_{31} vX + m_{32} vY + m_{33} vZ + m_{34} v = m_{21} X + m_{22} Y + m_{23} Z + m_{24}
\]
\[
0 = m_{11} X + m_{12} Y + m_{13} Z + m_{14} - m_{31} uX - m_{32} uY - m_{33} uZ - m_{34} u
\]
\[
0 = m_{21} X + m_{22} Y + m_{23} Z + m_{24} - m_{31} vX - m_{32} vY - m_{33} vZ - m_{34} v
\]
• Finally, solve for m’s entries using linear least squares
• Method 1 – \( \mathbf{Ax} = \mathbf{b} \) form

\[
\begin{bmatrix}
    s \mathbf{u} \\
    s \mathbf{v} \\
    s
\end{bmatrix}
= 
\begin{bmatrix}
    m_{11} & m_{12} & m_{13} & m_{14} \\
    m_{21} & m_{22} & m_{23} & m_{24} \\
    m_{31} & m_{32} & m_{33} & m_{34}
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

\[
\mathbf{M} = \mathbf{A} \backslash \mathbf{b}; \\
\mathbf{M} = [\mathbf{M}; 1]; \\
\mathbf{M} = \text{reshape}(\mathbf{M}, [], 3)';
\]
Unknown Camera Parameters

$$\begin{bmatrix} s & u \\ s & v \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Or, solve for m’s entries using total linear least-squares.
- Method 2 – $Ax=0$ form

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$[U, S, V] = \text{svd}(A);$$
$$M = V(:, \text{end});$$
$$M = \text{reshape}(M, [], 3)';$$
How do we calibrate a camera?

Known 2d
image coords

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>880</td>
<td>214</td>
</tr>
<tr>
<td>43</td>
<td>203</td>
</tr>
<tr>
<td>270</td>
<td>197</td>
</tr>
<tr>
<td>886</td>
<td>347</td>
</tr>
<tr>
<td>745</td>
<td>302</td>
</tr>
<tr>
<td>943</td>
<td>128</td>
</tr>
<tr>
<td>476</td>
<td>590</td>
</tr>
<tr>
<td>419</td>
<td>214</td>
</tr>
<tr>
<td>317</td>
<td>335</td>
</tr>
<tr>
<td>783</td>
<td>521</td>
</tr>
<tr>
<td>235</td>
<td>427</td>
</tr>
<tr>
<td>665</td>
<td>429</td>
</tr>
<tr>
<td>655</td>
<td>362</td>
</tr>
<tr>
<td>427</td>
<td>333</td>
</tr>
<tr>
<td>412</td>
<td>415</td>
</tr>
<tr>
<td>746</td>
<td>351</td>
</tr>
<tr>
<td>434</td>
<td>415</td>
</tr>
<tr>
<td>525</td>
<td>234</td>
</tr>
<tr>
<td>716</td>
<td>308</td>
</tr>
<tr>
<td>602</td>
<td>187</td>
</tr>
</tbody>
</table>

Known 3d
world locations

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>312.747</td>
<td>309.140</td>
<td>30.086</td>
</tr>
<tr>
<td>305.796</td>
<td>311.649</td>
<td>30.356</td>
</tr>
<tr>
<td>307.694</td>
<td>312.358</td>
<td>30.418</td>
</tr>
<tr>
<td>310.149</td>
<td>307.186</td>
<td>29.298</td>
</tr>
<tr>
<td>311.937</td>
<td>310.105</td>
<td>29.216</td>
</tr>
<tr>
<td>311.202</td>
<td>307.572</td>
<td>30.682</td>
</tr>
<tr>
<td>307.106</td>
<td>306.876</td>
<td>28.660</td>
</tr>
<tr>
<td>309.317</td>
<td>312.490</td>
<td>30.230</td>
</tr>
<tr>
<td>307.435</td>
<td>310.151</td>
<td>29.318</td>
</tr>
<tr>
<td>308.253</td>
<td>306.300</td>
<td>28.881</td>
</tr>
<tr>
<td>306.650</td>
<td>309.301</td>
<td>28.905</td>
</tr>
<tr>
<td>308.069</td>
<td>306.831</td>
<td>29.189</td>
</tr>
<tr>
<td>309.671</td>
<td>308.834</td>
<td>29.029</td>
</tr>
<tr>
<td>308.255</td>
<td>309.955</td>
<td>29.267</td>
</tr>
<tr>
<td>307.546</td>
<td>308.613</td>
<td>28.963</td>
</tr>
<tr>
<td>311.036</td>
<td>309.206</td>
<td>28.913</td>
</tr>
<tr>
<td>307.518</td>
<td>308.175</td>
<td>29.069</td>
</tr>
<tr>
<td>309.950</td>
<td>311.262</td>
<td>29.990</td>
</tr>
<tr>
<td>312.160</td>
<td>310.772</td>
<td>29.080</td>
</tr>
<tr>
<td>311.988</td>
<td>312.709</td>
<td>30.514</td>
</tr>
</tbody>
</table>
Projection error defined by two equations – one for $u$ and one for $v$

$$
\begin{bmatrix}
312.747 & 309.140 & 30.086 & 1 & 0 & 0 & 0 & 0 & -880 \times 312.747 & -880 \times 309.140 & -880 \times 30.086 & -880
\end{bmatrix}
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 312.747 & 309.140 & 30.086 & 1 & -214 \times 312.747 & -214 \times 309.140 & -214 \times 30.086 & -214
\end{bmatrix}
$$

...
Known 2d image coords

<table>
<thead>
<tr>
<th>Known 3d world locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
</tr>
<tr>
<td>312.747</td>
</tr>
<tr>
<td>305.796</td>
</tr>
<tr>
<td>307.694</td>
</tr>
<tr>
<td>310.149</td>
</tr>
<tr>
<td>311.937</td>
</tr>
<tr>
<td>311.202</td>
</tr>
<tr>
<td>307.106</td>
</tr>
<tr>
<td>309.317</td>
</tr>
<tr>
<td>307.435</td>
</tr>
</tbody>
</table>

Projection error defined by two equations – one for u and one for v

\[
\begin{bmatrix}
X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_nV \\
0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_nV \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
m_{11} \\
m_{12} \\
m_{13} \\
m_{14} \\
m_{21} \\
m_{22} \\
m_{23} \\
m_{24} \\
m_{31} \\
m_{32} \\
m_{33} \\
m_{34} \\
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \end{bmatrix}
\]
How many points do I need to fit the model?

\[ \mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}] \mathbf{X} \]

Degrees of freedom?

\[
\begin{bmatrix}
\mathbf{u} \\
\mathbf{v} \\
1
\end{bmatrix} =
\begin{bmatrix}
\alpha & s & u_0 \\
0 & \beta & v_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
r_{11} & r_{12} & r_{13} & t_x \\
r_{21} & r_{22} & r_{23} & t_y \\
r_{31} & r_{32} & r_{33} & t_z
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

Think 3:
- Rotation around \( x \)
- Rotation around \( y \)
- Rotation around \( z \)
How many points do I need to fit the model?

\[ x = K[R \ t]X \]

Degrees of freedom?

\[
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix}
= \begin{bmatrix}
\alpha & s & u_0 \\
0 & \beta & v_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
r_{11} & r_{12} & r_{13} & t_x \\
r_{21} & r_{22} & r_{23} & t_y \\
r_{31} & r_{32} & r_{33} & t_z
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

\( M \) is 3x4, so 12 unknowns, but projective scale ambiguity – 11 deg. freedom. One equation per unknown -> 5 1/2 point correspondences determines a solution (e.g., either \( u \) or \( v \)).

More than 5 1/2 point correspondences -> overdetermined, many solutions to \( M \). Least squares is finding the solution that best satisfies the overdetermined system.

Why use more than 6? Robustness to error in feature points.
Calibration with linear method

• Advantages
  – Easy to formulate and solve
  – Provides initialization for non-linear methods

• Disadvantages
  – Doesn’t directly give you camera parameters
  – Doesn’t model radial distortion
  – Can’t impose constraints, such as known focal length

• Non-linear methods are preferred
  – Define error as difference between projected points and measured points
  – Minimize error using Newton’s method or other non-linear optimization
Can we factorize $M$ back to $K \{R \mid T\}$?

• Yes!

• We can directly solve for the individual entries of $K \{R \mid T\}$.
Extracting camera parameters

\[
M = \frac{\begin{pmatrix}
\alpha r_1^T - \alpha \cot \theta r_2^T + u_0 r_3^T \\
\frac{\beta}{\sin \theta} r_2^T + v_0 r_3^T \\
r_3^T \\
\end{pmatrix}}{\rho} = K \begin{bmatrix} R & T \end{bmatrix}
\]

Box 1

\[
A = \begin{bmatrix}
    a_1^T \\
    a_2^T \\
    a_3^T 
\end{bmatrix} \quad b = \begin{bmatrix}
    b_1 \\
    b_2 \\
    b_3 
\end{bmatrix}
\]

Intrinsic

\[
\rho = \frac{\pm 1}{|a_3|} \quad u_o = \rho^2 (a_1 \cdot a_3) \\
v_o = \rho^2 (a_2 \cdot a_3)
\]

\[
\cos \theta = \frac{(a_1 \times a_3) \cdot (a_2 \times a_3)}{|a_1 \times a_3| \cdot |a_2 \times a_3|}
\]
Extracting camera parameters

\[
\begin{align*}
\mathbf{M} &= \frac{\begin{pmatrix}
\alpha \mathbf{r}_1^T - \alpha \cot \theta \mathbf{r}_2^T + u_0 \mathbf{r}_3^T \\
\beta \frac{\mathbf{r}_2^T}{\sin \theta} + v_0 \mathbf{r}_3^T \\
\mathbf{r}_3^T
\end{pmatrix}}{\rho} \\
&= \begin{pmatrix}
\alpha t_x - \alpha \cot \theta t_y + u_0 t_z \\
\beta \frac{t_y}{\sin \theta} + v_0 t_z \\
t_z
\end{pmatrix} = K \begin{pmatrix} R & T \end{pmatrix}
\end{align*}
\]

\[
\begin{align*}
\mathbf{A} &= \begin{bmatrix}
\mathbf{a}_1^T \\
\mathbf{a}_2^T \\
\mathbf{a}_3^T
\end{bmatrix} \\
\mathbf{b} &= \begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}
\end{align*}
\]

Intrinsic values

\[
\alpha = \rho^2 |\mathbf{a}_1 \times \mathbf{a}_3| \sin \theta
\]
\[
\beta = \rho^2 |\mathbf{a}_2 \times \mathbf{a}_3| \sin \theta
\]

Estimated values
Extracting camera parameters

\[
\mathbf{M} = \begin{pmatrix}
\alpha \mathbf{r}_1^T - \alpha \cot \theta \mathbf{r}_2^T + u_0 \mathbf{r}_3^T \\
\frac{\beta}{\sin \theta} \mathbf{r}_2^T + v_0 \mathbf{r}_3^T \\
\mathbf{r}_3^T
\end{pmatrix}
\begin{pmatrix}
\alpha t_x - \alpha \cot \theta t_y + u_0 t_z \\
\frac{\beta}{\sin \theta} t_y + v_0 t_z \\
t_z
\end{pmatrix} = K \begin{bmatrix} R & T \end{bmatrix}
\]

\[
A = \begin{bmatrix}
\mathbf{a}_1^T \\
\mathbf{a}_2^T \\
\mathbf{a}_3^T
\end{bmatrix}
\quad b = \begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}
\]

Extrinsic

\[
r_1 = \frac{(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_2 \times \mathbf{a}_3|}
\quad r_3 = \pm \frac{\mathbf{a}_3}{|\mathbf{a}_3|}
\]

\[
r_2 = r_3 \times r_1 
\quad T = \rho \ K^{-1} b
\]

Estimated values
Can we factorize $M$ back to $K [R \mid T]$?

• Yes!

• We can also use $RQ$ factorization (not $QR$)
  – $R$ in $RQ$ is not rotation matrix $R$; crossed names!

• $R$ (right diagonal) is $K$

• $Q$ (orthogonal basis) is $R$.

• $T$, the last column of $[R \mid T]$, is $\text{inv}(K) \times$ last column of $M$.
  – But you need to do a bit of post-processing to make sure that the matrices are valid. See http://ksimek.github.io/2012/08/14/decompose/
General case, with calibrated cameras

- The two cameras need not have parallel optical axes.
Stereo correspondence constraints

- Given $p$ in left image, where can corresponding point $p'$ be?
Given p in left image, where can corresponding point p’ be?
Stereo correspondence constraints
Stereo correspondence constraints

- Geometry of two views allows us to constrain where the corresponding pixel for some image point in the first view must occur in the second view.

**Epipolar constraint**: Why is this useful?
- Reduces correspondence problem to 1D search along *conjugate* epipolar lines
Epipolar geometry

- Epipolar Plane
- Epipoles
- Baseline
- Epipolar Lines

Adapted from M. Pollefeys, UNC
Epipolar geometry: terms

- **Baseline**: line joining the camera centers
- **Epipole**: point of intersection of baseline with the image plane
- **Epipolar plane**: plane containing baseline and world point
- **Epipolar line**: intersection of epipolar plane with the image plane

- All epipolar lines intersect at the epipole
- An epipolar plane intersects the left and right image planes in epipolar lines
• Potential matches for $p$ have to lie on the corresponding epipolar line $l'$. 

• Potential matches for $p'$ have to lie on the corresponding epipolar line $l$.

http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

Source: M. Pollefeys
Example
Example: converging cameras

As position of 3d point varies, epipolar lines “rotate” about the baseline

Figure from Hartley & Zisserman
Example: motion parallel with image plane

Figure from Hartley & Zisserman
Example: forward motion

Epipole has same coordinates in both images. Points move along lines radiating from e: “Focus of expansion”
Uncertainty
Reconstruction Error

- Given the uncertainty in pixel projection of the point, what is the error in depth?
Reconstruction Error

Error in depth ($\Delta Z$) depends on
- $Z$, $T$ (baseline) and $f$ (focal length)
- $\Delta x_L$ and $\Delta x_R$
Reconstruction Error

What is the expected value and variance of the error?

First source of error is in locating the corresponding pixel, then the disparity measure $d$

$$d = x_L - x_R$$

Taking total derivative of each side

$$D(d) = Dx_L - Dx_R$$

$$\Delta d = \Delta x_L - \Delta x_R$$

To avoid confusion with disparity
We denote total derivative as “$D$”
Reconstruction Error

Assuming $\Delta x_L$ and $\Delta x_R$ are independent and zero mean

$$
\mu = E[\Delta d] = E[\Delta x_L] - E[\Delta x_R] = 0
$$

$$
Var[\Delta d] = E[(\Delta d - \mu)^2] = E[(\Delta d)^2]
$$

and

$$
Var[\Delta d] = E[(\Delta x_L - \Delta x_R)^2] = E[\Delta x_L^2 - 2\Delta x_L \Delta x_R + \Delta x_R^2]
$$

$$
= E[\Delta x_L^2] - 2E[\Delta x_L \Delta x_R] + E[\Delta x_R^2]
$$

$$
= E[\Delta x_L^2] + E[\Delta x_R^2]
$$

$$
\sigma_d^2 = \sigma_L^2 + \sigma_R^2
$$
Reconstruction Error

Let us take the total derivative of \( Z = f \frac{T}{d} \)

\[
\Delta Z = f \frac{T}{d^2} (-\Delta d)
\]

The mean error is

\[
\mu_Z = E[\Delta Z] \quad \Rightarrow \quad \mu_Z = 0
\]

The variance of the error is

\[
\sigma_Z^2 = E[(\Delta Z - \mu_Z)^2] = E[\Delta Z^2] = \left( f \frac{T}{d^2} \right)^2 E[(-\Delta d)^2]
\]

\[
\sigma_Z^2 = \left( f \frac{T}{d^2} \right)^2 \sigma_d^2 \quad \Rightarrow \quad \sigma_Z = \left( f \frac{T}{d^2} \right) \sigma_d = Z \frac{\sigma_d}{d}
\]
Reconstruction Error - Example

A stereo vision system estimates the disparity of a point as $d=10$ pixels.

What is the depth ($Z$) of the point, if $f = 500$ pixels and $b = 10$ cm?

$$Z = f \frac{T}{d} = (500 \text{ pixels}) \frac{10 \text{ cm}}{10 \text{ pixel}} = 500 \text{ cm}$$

What is the uncertainty (standard deviation) of the depth, if the standard deviation of locating a feature in each image = 1 pixel?

$$\sigma_d = \frac{Z \sigma_d}{d} = (500 \text{ cm}) \frac{\sqrt{2} \text{ pixel}}{10 \text{ pixel}} \approx 70 \text{ cm}$$

$$\sigma_d^2 = \sigma_{x_L}^2 + \sigma_{x_R}^2 = 2 \Rightarrow \sigma_d = \sqrt{2}$$

How to handle uncertainty in both disparity and focal length?

$$Z = f \frac{T}{d} \quad \Delta Z = \Delta f \frac{T}{d} + f \frac{T}{d^2} (-\Delta d)$$
Geometry – general case

- Cameras not aligned, but we still know relative pose
- Assuming $f=1$, we have
  
  \[
  \mathbf{p}_L = \begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix}, \quad \mathbf{p}_R = \begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix}
  \]

- In principle, you can find $P$ by intersecting the rays $O_Lp_L$ and $O_Rp_R$
- However, they may not intersect
- Instead, find the midpoint of the segment perpendicular to the two rays
Triangulation cont’d

- The projection of P onto the left image is
  \[ Z_L \mathbf{p}_L = \mathbf{M}_L \mathbf{P} \]
- The projection of P onto the right image is
  \[ Z_R \mathbf{p}_R = \mathbf{M}_R \mathbf{P} \]
- where
  \[
  \mathbf{M}_L = \begin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
  \end{pmatrix}
  \]
  \[
  \mathbf{M}_R = \begin{pmatrix}
    r_{11} & r_{12} & r_{13} & t_x \\
    r_{21} & r_{22} & r_{23} & t_y \\
    r_{31} & r_{32} & r_{33} & t_z \\
  \end{pmatrix}
  \]

\[
  = \begin{pmatrix}
    R^R_L \\
    t^R_{t_{Lorg}}
  \end{pmatrix}
  \]
Triangulation cont’d

- Note that $\mathbf{p}_L$ and $\mathbf{M}_L \mathbf{P}$ are parallel, so their cross product should be zero.
- Similarly for $\mathbf{p}_R$ and $\mathbf{M}_R \mathbf{P}$.
- Point $\mathbf{P}$ should satisfy both:
  \[
  p_L \times \mathbf{M}_L \mathbf{P} = 0 \]
  \[
  p_R \times \mathbf{M}_R \mathbf{P} = 0
  \]
- This is a system of four equations; can solve for the three unknowns $(X_L, Y_L, Z_L)$ using least squares.
- Method also works for more than two cameras.
• For a given stereo rig, how do we express the epipolar constraints algebraically?
Stereo geometry, with calibrated cameras

If the rig is calibrated, we know:

how to rotate and translate camera reference frame 1 to get to camera reference frame 2.

Rotation: 3 x 3 matrix; translation: 3 vector.
Rotation matrix

Express 3d rotation as series of rotations around coordinate axes by angles $\alpha, \beta, \gamma$

Overall rotation is product of these elementary rotations:

$$R = R_x R_y R_z$$
3d rigid transformation

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix} = 
\begin{bmatrix}
r_{11} & r_{12} & r_{13} \\
r_{21} & r_{22} & r_{23} \\
r_{31} & r_{32} & r_{33}
\end{bmatrix} 
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + 
\begin{bmatrix}
T_x \\
T_y \\
T_z
\end{bmatrix}
\]

\[X' = RX + T\]
Stereo geometry, with calibrated cameras

Camera-centered coordinate systems are related by known rotation $\mathbf{R}$ and translation $\mathbf{T}$:

$$\mathbf{X}' = \mathbf{RX} + \mathbf{T}$$
Cross product

\[ \vec{a} \times \vec{b} = \vec{c} \]
\[ \vec{a} \cdot \vec{c} = 0 \]
\[ \vec{b} \cdot \vec{c} = 0 \]

Vector cross product takes two vectors and returns a third vector that’s perpendicular to both inputs.

So here, c is perpendicular to both a and b, which means the dot product = 0.
From geometry to algebra

Because these vectors are all coplanar
From geometry to algebra

Instead of treating $p$ as a point, let’s consider it as a 3D direction vector, no matter where it is

$$\begin{align*}
f=1 \\
p &= \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \\
p' &= \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}
\end{align*}$$

The direction of $p'$ in camera coordinates $C$ is $\begin{pmatrix} C \end{pmatrix} R p'$
Instead of treating \( p \) as a point, let’s consider it as a 3D direction vector, no matter where it is

\[
f=1 \quad p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad p' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}
\]

The direction of \( p' \) in camera coordinates \( C \) is \( C R p' \)
From geometry to algebra

\[ \overrightarrow{O_C p} \cdot \left( \overrightarrow{O_C p'} \times \overrightarrow{O_C O_C'} \right) = 0 \]

\[ \mathbf{p} \cdot (\mathbf{t} \times \mathbf{Rp'}) = 0 \]

Coplanar constraint where \( \mathbf{R} \) is the rotation from \( C' \) to \( C \)

And \( \mathbf{t} \) is translation of the camera \( C' \) to camera \( C \)

\[ \begin{bmatrix} c \mathbf{t}_{C'} \end{bmatrix} \]

Remember that pose of camera \( C' \) to camera \( C \) is

\[ \begin{bmatrix} c \mathbf{R} & c \mathbf{t}_{C'} \\ 0 & 1 \end{bmatrix} \]
From geometry to algebra

\[ X' = RX + T \]

\[ T \times X' = \text{Normal to the plane} \]

\[ = T \times RX \]

\[ X' \cdot (T \times X') = X' \cdot (T \times RX) = 0 \]
Matrix form of cross product

$\vec{a} \times \vec{b} = \vec{c}$

$\vec{a} \cdot \vec{c} = 0$

$\vec{b} \cdot \vec{c} = 0$

Can be expressed as a matrix multiplication.

$[a_x] = \begin{bmatrix}
0 & -a_z & a_y \\
a_z & 0 & -a_x \\
-a_y & a_x & 0
\end{bmatrix}$

$\vec{a} \times \vec{b} = [a_x] \vec{b}$

Grauman
Matrix form of cross product - Example

\[ \vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \]

\[ \vec{a} \times \vec{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x) \]

\[ [\vec{a}] \cdot \vec{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x) \]

\[ [a_x] = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \]
From geometry to algebra

\[ \mathbf{X}' = \mathbf{R} \mathbf{X} + \mathbf{T} \]

\[ \mathbf{T} \times \mathbf{X}' = \mathbf{T} \times \mathbf{R} \mathbf{X} + \mathbf{T} \times \mathbf{T} \]

Normal to the plane

\[ = \mathbf{T} \times \mathbf{R} \mathbf{X} \]

\[ \mathbf{X}' \cdot (\mathbf{T} \times \mathbf{X}') = \mathbf{X}' \cdot (\mathbf{T} \times \mathbf{R} \mathbf{X}) = 0 \]
Essential matrix

\[
X' \cdot (T \times RX) = 0
\]
\[
X' \cdot (T_x \cdot RX) = 0
\]

Let \( E = T_x \cdot R \)

\[
X'^T E X = 0
\]

This holds for the rays \( p \) and \( p' \) that are parallel to the camera-centered position vectors \( X \) and \( X' \), so we have:

\[
p'^T E p = 0
\]

\( E \) is called the **essential matrix**, which relates corresponding image points [Longuet-Higgins 1981]
Essential matrix and epipolar lines

\[ p'^T E p = 0 \]

Epipolar constraint: if we observe point \( p \) in one image, then its position \( p' \) in second image must satisfy this equation.

\( E p \) is the coordinate vector representing the epipolar line associated with point \( p \)

\( E^T p' \) is the coordinate vector representing the epipolar line associated with point \( p' \)
Essential matrix: properties

- Relates image of corresponding points in both cameras, given rotation and translation
- Assuming intrinsic parameters are known

$$E = T_x \cdot R$$
Essential matrix example: parallel cameras

$$R = \begin{bmatrix} -d, 0, 0 \\ T \end{bmatrix}$$

$$E = [T]_x R =$$

$$p^T E p = 0$$

For the parallel cameras, image of any point must lie on same horizontal line in each image plane.
What about when cameras’ optical axes are not parallel?
Stereo image rectification

In practice, it is convenient if image scanlines are the epipolar lines.

reproject image planes onto a common plane parallel to the line between optical centers
pixel motion is horizontal after this transformation

two homographies (3x3 transforms), one for each input image reprojection

Adapted from Li Zhang
Stereo image rectification: example

Source: Alyosha Efros
Essential Matrix estimation example
Create an image pair with known rotation and translation between the views, and corresponding image points.

clear all
close all
L = 300; % size of image in pixels
I1 = zeros(L,L);
% Define f, u0, v0
f = L;
u0 = L/2;
v0 = L/2;
% Create the matrix of intrinsic camera parameters
K = [ f 0 u0;
     0 f v0;
     0 0 1 ];
DEG_TO_RAD = pi/180;
% Create some points on the face of a cube
P_M = [0 0 0 0 0 0 0 0 0 1 2 1 2 1 2 1 2;
       2 1 0 2 1 0 2 1 0 0 0 0 0 0 0;
       0 0 0 -1 -1 -1 -2 -2 -2 0 -1 -1 -2 -2;
       1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
NPTS = length(P_M);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define pose of model with respect to camera 1
ax = 120 * DEG_TO_RAD;
ay = 0 * DEG_TO_RAD;
az = 60 * DEG_TO_RAD;
Rx = [ 1 0 0;
      0 cos(ax) -sin(ax);
      0 sin(ax) cos(ax) ];
Ry = [ cos(ay) 0 sin(ay);
      0 1 0;
      -sin(ay) 0 cos(ay) ];
Rz = [ cos(az) -sin(az) 0;
      sin(az) cos(az) 0;
      0 0 1 ];
R_m_c1 = Rx * Ry * Rz;
Pmorg_c1 = [0; 0; 5]; % translation of model
wrt camera
M = [ R_m_c1 Pmorg_c1 ; 0 0 0 1 ];
% Extrinsic camera parameter matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Render image 1
p1 = M * P_M;
p1(1,:) = p1(1,:) ./ p1(3,:);
p1(2,:) = p1(2,:) ./ p1(3,:);
p1(3,:) = p1(3,:) ./ p1(3,:);
u1 = K * p1;
% Convert image points from normalized to unnormalized
for i=1:length(u1)
    x = round(u1(1,i));
    y = round(u1(2,i));
    I1(y-2:y+2, x-2:x+2) = 255;
end
figure(1), imshow(I1, []), title('View 1'); pause

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up second view.
% Define rotation of camera 2 with respect to camera 1
ax = 0 * DEG_TO_RAD;
ay = -25 * DEG_TO_RAD;
az = 0;
Rx = [ 1 0 0;
     0 cos(ax) -sin(ax);
     0 sin(ax) cos(ax) ];
Ry = [ cos(ay) 0 sin(ay);
     0 1 0;
     -sin(ay) 0 cos(ay) ];
Rz = [ cos(az) -sin(az) 0;
      sin(az) cos(az) 0;
      0 0 1 ];
R_c2_c1 = Rx * Ry * Rz;
% Define translation of camera 2 with respect to camera 1
Pc2org_c1 = [3; 0; 1];
% Figure out pose of model wrt camera 2.
H_m_c1 = [ R_m_c1 Pmorg_c1 ; 0 0 0 1 ];
H_c2_c1 = [ R_c2_c1 Pc2org_c1 ; 0 0 0 1 ];
H_c1_c2 = inv(H_c2_c1);
H_m_c2 = H_c1_c2 * H_m_c1;
R_m_c2 = H_m_c2(1:3,1:3);
Pmorg_c2 = H_m_c2(1:3,4);
% Extrinsic camera parameter matrix
M = [ R_m_c2 Pmorg_c2 ];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Render image 2
I2 = zeros(L,L);
p2 = M * P_M;
p2(1,:) = p2(1,:) ./ p2(3,:);
p2(2,:) = p2(2,:) ./ p2(3,:);
p2(3,:) = p2(3,:) ./ p2(3,:);
u2 = K * p2;
for i=1:length(u2)
    x = round(u2(1,i));
    y = round(u2(2,i));
    I2(y-2:y+2, x-2:x+2) = 255;
end
figure(2), imshow(I2, []), title('View 2');disp(['Points in image 1:']); disp(u1);
disp(['Points in image 2:']); disp(u2);
imwrite(I1, 'I1.tif');
imwrite(I2, 'I2.tif');

% This is the "true" essential matrix between the views
t = Pc2org_c1;
Etrue = [ 0 -t(3) t(2); t(3) 0 -t(1); -t(2) t(1) 0 ] * R_c2_c1;
disp('True essential matrix:'); disp(Etrue);
Createpoints.m
Calculating the Essential Matrix

- We have
  \[ p_0^T E p_1 = 0 \]
  \[ \begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0 \]

- Write out as equation
  \[ \begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} E_{11}x_1 + E_{12}y_1 + E_{13} \\ E_{21}x_1 + E_{22}y_1 + E_{13} \\ E_{31}x_1 + E_{32}y_1 + E_{33} \end{pmatrix} = 0 \]
  \[ E_{11}x_0x_1 + E_{12}x_0y_1 + E_{13}x_0 + E_{21}y_0x_1 + E_{22}y_0y_1 + E_{13}y_0 + E_{31}x_1 + E_{32}y_1 + E_{33} = 0 \]

- Write as \[ A \mathbf{x} = 0 \], where \[ \mathbf{x} = (E_{11}, E_{12}, E_{13}, \ldots, E_{33}) \]
  \[ \begin{pmatrix} x_0x_1 & x_0y_1 & x_0 & y_0x_1 & y_0y_1 & y_0 & x_1 & y_1 & 1 \end{pmatrix} \begin{pmatrix} E_{11} \\ E_{12} \\ E_{13} \\ \vdots \\ E_{33} \end{pmatrix} = 0 \]

Actually, \( A \) will have one row for each point correspondence \((x_0,y_0) - (x_1,y_1)\)
Solving for $E$

- We have $A \mathbf{x} = 0$
- This is a system of homogeneous equations
- Ignoring the trivial solution $\mathbf{x} = 0$, you can find a unique solution for $\mathbf{x}$ that gives the least squares solution for $\mathbf{x}$; i.e., the solution that minimizes $\sum (\mathbf{p}_0^T E \mathbf{p}_1)^2$

- It is proportional to the only zero eigenvalue of $A^T A$

- You can use Singular Value Decomposition to find it: $A = U D V^T$
- The solution $\mathbf{x}$ is the column of $V$ corresponding to the only null singular value of $A$
- This is the rightmost column of $V$
Finding E using 8-point linear algorithm

- Solve $A \mathbf{x} = 0$ using Singular Value Decomposition (SVD):
  \[
  A = U D V^T 
  \]
- The solution $\mathbf{x}$ is the rightmost column of $V$

```matlab
% Compute essential matrix E from point correspondences.
% We know that \( p_1' E p_2 = 0 \), where \( p_1, p_2 \) are the normalized image coords.
% We write out the equations in the unknowns \( E(i,j) \)
% \( A \mathbf{x} = 0 \)
A = [p1s(1,:)'.*p2s(1,:)' p1s(1,:)'.*p2s(2,:)' p1s(1,:)'; ...
    p1s(2,:)'.*p2s(1,:)' p1s(2,:)'.*p2s(2,:)' p1s(2,:)]'; ...
    p2s(1,:)'; p2s(2,:)'; ones(length(p1s),1)];

% The solution to \( Ax=0 \) is the singular vector of \( A \) corresponding to the
% smallest singular value; that is, the last column of \( V \) in \( A = UDV' \)
[U,D,V] = svd(A);
\( x = V(:,\text{size}(V,2)) \); \% get last column of \( V \)

% Put unknowns into a 3x3 matrix. Transpose because Matlab's "reshape"
% uses the order \( E_{11} E_{21} E_{31} E_{12} \ldots \)
Escale = reshape(x,3,3)';
```
Observations

• Results can be unstable, due to poor numerical conditioning

• We can improve results by:
  – Preconditioning: We will first translate and scale the data points so they are centered at the origin and the average distance to the origin is $\sqrt{2}$
  – Postconditioning: The values of $E$ are not independent. There are only five independent parameters. $E$ must have rank=2 ... we will enforce this
Preconditioning

% Scale and translate image points so that the centroid of the points is at the origin, and the average distance of the points to the origin is equal to sqrt(2).

```matlab
% xn is a 2xN matrix
N = size(xn,2);

t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is 1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is sqrt(2)
T1 = [s*eye(2), -s*t ; 0 0 1];
p1s = T1 * p1;

% xn is a 2xN matrix
N = size(xn,2);

t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is 1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is sqrt(2)
T2 = [s*eye(2), -s*t ; 0 0 1];
p2s = T2 * p2;
```
Postconditioning

• Enforce the property that the essential matrix has only two non-zero eigenvalues, and that they are equal

• You can force this by taking the SVD of $E$

  $$E = U S V^T$$

• then reconstruct $E$ with only its first two eigenvalues

  $$E' = U \text{diag}(1,1,0) V^T$$

• In Matlab

  ```matlab
  [U,D,V] = svd(Escale);
  Escale = U*diag([1 1 0])*V';
  ```
Undoing Preconditioning

• After computing the essential matrix $\text{Escale}$, you then have to adjust the result to undo the effect of point scaling. This can be done by $E = T_1^T \text{Escale} \ T_2$.

$$E = T_1' \ * \ \text{Escale} \ * \ T_2; \quad \% \ \text{Undo scaling}$$
Example

• Run program “drawepipolar.m”
  – This inputs a pair of images, a set of corresponding points, and an essential matrix
  – It draws epipolar lines in the images
Structure from motion
Recovering Motion Parameters (advanced)

• Once you have essential matrix $E$, you can recover the relative motion between cameras

• Recall that the essential matrix is made up of the translation and rotation matrices; ie., $E = [t]_x R$

• We can extract the translation and rotation by taking SVD of $E$ again, $E = U D V^T$

• Then form the following combinations:
  – $t$ is either $u_3$ or $-u_3$, where $u_3$ is the third (last) column of $U$
  – $R$ is either $U W V^T$ or $U W^T V^T$

• where

$$W = \begin{pmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}$$
Recovering Motion (continued)

- There are actually 4 possible solutions, but 3 of them are nonsensical, meaning that they represent situations where the scene is behind one or both of the cameras.

- Only one of the four solutions corresponds to the case where the scene points are in front of both cameras.
  - To find the correct one, we will need to reconstruct a scene point and see if it is front of both cameras.

- Remember that we can only determine motion up to a scale factor.
  - The translation $t$ will have an arbitrary magnitude; we can only know the direction of $t$.
  - The rotation $R$ is correct, though.
% Extract motion parameters from essential matrix.
% We know that E = [tx] R, where
%  [tx] = [ 0 -t3 t2; t3 0 -t1; -t2 t1 0]
% If we take SVD of E, we get E = U diag(1,1,0) V'
% t is the last column of U
[U,D,V] = svd(E);

W = [0 -1 0; 1 0 0; 0 0 1];
Hresult_c2_c1(:,:,1) = [ U*W*V'  U(:,3) ; 0 0 0 1];
Hresult_c2_c1(:,:,2) = [ U*W*V'  -U(:,3) ; 0 0 0 1];
Hresult_c2_c1(:,:,3) = [ U*W'*V'  U(:,3) ; 0 0 0 1];
Hresult_c2_c1(:,:,4) = [ U*W'*V'  -U(:,3) ; 0 0 0 1];

% make sure each rotation component is a legal rotation matrix
for k=1:4
    if det(Hresult_c2_c1(1:3,1:3,k)) < 0
        Hresult_c2_c1(1:3,1:3,k) = -Hresult_c2_c1(1:3,1:3,k);
    end
end
Reconstruction

• Given a hypothesized pose between the cameras, we want to reconstruct the 3D position of a point from its image projections

• The projection of $\mathbf{P}$ onto the left image is

$$Z_1 \mathbf{p}_1 = \mathbf{M}_1 \mathbf{P}$$

• The projection of $\mathbf{P}$ onto the right image is

$$Z_2 \mathbf{p}_2 = \mathbf{M}_2 \mathbf{P}$$

• where

$$\mathbf{M}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{M}_2 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} = \begin{pmatrix} 2 \mathbf{R} & 2 \mathbf{t}_{1org} \end{pmatrix}$$
Reconstruction (continued)

• Note that $p_1$ and $M_1 P$ are parallel, so their cross product should be zero.

• Similarly for $p_2$ and $M_2 P$.

• Point $P$ should satisfy both
  
  \[ p_1 \times M_1 P = 0 \]
  
  \[ p_2 \times M_2 P = 0 \]

• This is a system of four equations; can solve for the three unknowns of $P$ ($X, Y, Z$) using least squares.
Reconstruction (continued)

- Rewrite

\[
p_1 \times M_1 P = 0 \\
p_2 \times M_2 P = 0
\]

\[
\begin{bmatrix}
(p_1) \times \\
(p_2) \times
\end{bmatrix}
\begin{bmatrix}
M_1 \\
M_2
\end{bmatrix}P = 0
\]

- Solve the system \( AP = 0 \) using SVD

```matlab
plx = [ 0    -p1(3,1)  p1(2,1);    % skew symmetric matrix
        p1(3,1)  0      -p1(1,1);  \\
       -p1(2,1)  p1(1,1)  0 ];

p2x = [ 0    -p2(3,1)  p2(2,1);    \\
        p2(3,1)  0      -p2(1,1);  \\
       -p2(2,1)  p2(1,1)  0 ];

A = [ plx * M1; p2x * M2 ];
% The solution to AP=0 is the singular vector of A corresponding to the 
% smallest singular value; that is, the last column of V in A=UDV'
[U,D,V] = svd(A);
P = V(:,4);                  % get last column of V
P1est = P/P(4);             % normalize
```
Testing the 4 possibilities

- Take a corresponding pair, p1 and p2
- For each possible pose $^{2}_1\mathbf{H}$
  - Reconstruct the 3D position of the point with respect to camera 1: $^{1}\mathbf{P}$
  - Compute 3D position of the point with respect to camera 2: $^{2}\mathbf{P} = ^{1}_2\mathbf{H}^{1}\mathbf{P}$
  - Test if the Z components of $^{1}\mathbf{P}$ and $^{2}\mathbf{P}$ are both > 0

```matlab
for i=1:4
    Hresult_c1_c2 = inv(Hresult_c2_c1(:,:,i));
    M2 = Hresult_c1_c2(1:3,1:4);
    A = [ p1x * M1; p2x * M2 ];
    % The solution to AP=0 is the singular vector of A corresponding to the
    % smallest singular value; that is, the last column of V in A=UDV'
    [U,D,V] = svd(A);
    P = V(:,4); % get last column of V
    P1est = P/P(4); % normalize
    P2est = Hresult_c1_c2 * P1est;
    if P1est(3) > 0 & P2est(3) > 0
        Hest_c2_c1 = Hresult_c2_c1(:,:,i); % We've found a good solution
        break; % break out of for loop; can stop searching
    end
end
```
Example

- Run program “twoview.m”
  - This inputs a pair of images, a set of corresponding points, and an essential matrix
  - It calculates the pose (rotation & translation) between the views

- Compare computed pose of c2 with respect to c1, to ground truth

True pose:
\[ H_{c2,c1} = \]

\[
\begin{bmatrix}
0.9063 & 0 & -0.4226 & 3.0000 \\
0 & 1.0000 & 0 & 0 \\
0.4226 & 0 & 0.9063 & 1.0000 \\
0 & 0 & 0 & 1.0000 \\
\end{bmatrix}
\]

Reconstructed pose of camera2 wrt camera1:

\[
\begin{bmatrix}
0.9063 & -0.0016 & -0.4226 & 0.9487 \\
-0.0006 & 1.0000 & -0.0051 & -0.0000 \\
0.4226 & 0.0049 & 0.9063 & 0.3162 \\
0 & 0 & 0 & 1.0000 \\
\end{bmatrix}
\]
Reconstructing rest of points

- For each pair of corresponding image points $p_1$ and $p_2$
  - Form the skew symmetric matrices $[p_{1x}]$ and $[p_{2x}]$
  - Form the matrix equation
    $$\begin{pmatrix} [p_{1x}] M_1 \\ [p_{2x}] M_2 \end{pmatrix} P = 0$$
  - where
    $$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad M_2 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} = \begin{pmatrix} I & \mathbf{R} \\ 0 & \mathbf{t}_{\text{org}} \end{pmatrix}$$
  - solve for $P$ using SVD

Recall $\mathbf{a} \times \mathbf{b} = [\mathbf{a}_x][\mathbf{b}]$, where
$$\mathbf{a}_x = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$
Hest_c1_c2 = inv(Hest_c2_c1);
M2est = Hest_c1_c2(1:3,:);

% Reconstruct point positions (these are good to the same scale factor)
fprintf('Reconstructed points wrt camera1:
');
for i=1:length(p1)
    p1x = [ 0   -p1(3,i)   p1(2,i);
           p1(3,i)  0   -p1(1,i);
           -p1(2,i) p1(1,i)  0 ];
    p2x = [ 0   -p2(3,i)   p2(2,i);
           p2(3,i)  0   -p2(1,i);
           -p2(2,i) p2(1,i)  0 ];
     A = [ p1x * M1; p2x * M2est ];
     [U,D,V] = svd(A);
     P = V(:,4);                 % get last column of V
     P1est(:,i) = P/P(4);       % normalize
     fprintf('%f %f %f
', P1est(1,i), P1est(2,i), P1est(3,i));
end

% Show the reconstruction result in 3D
figure;
plot3(P1est(1,:),P1est(2,:),P1est(3,:),'d');
axis equal;
axis vis3d;
Example

Reconstructed points wrt camera 1

<table>
<thead>
<tr>
<th>Reconstructed points wrt camera 1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.547790  -0.158133  1.855227</td>
</tr>
<tr>
<td>-0.273890  -0.079065  1.718247</td>
</tr>
<tr>
<td>0.000000  0.000000  1.581274</td>
</tr>
<tr>
<td>-0.547793  0.115762  2.013372</td>
</tr>
<tr>
<td>-0.273892  0.194825  1.876391</td>
</tr>
<tr>
<td>0.000000  0.273887  1.739416</td>
</tr>
<tr>
<td>-0.547797  0.389660  2.171521</td>
</tr>
<tr>
<td>-0.273893  0.468720  2.034536</td>
</tr>
<tr>
<td>0.000000  0.547780  1.897560</td>
</tr>
<tr>
<td>0.158129  -0.136944  1.818480</td>
</tr>
<tr>
<td>0.316261  -0.273893  2.055698</td>
</tr>
<tr>
<td>0.158130  0.136945  1.976628</td>
</tr>
<tr>
<td>0.316264  0.000000  2.213851</td>
</tr>
<tr>
<td>0.158132  0.410840  2.134777</td>
</tr>
<tr>
<td>0.316267  0.273898  2.372005</td>
</tr>
</tbody>
</table>

True 3D locations of points wrt camera 1

<table>
<thead>
<tr>
<th>True 3D locations of points wrt camera 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.7321  -0.5000  5.8660  1.0000</td>
</tr>
<tr>
<td>-0.8660  -0.2500  5.4330  1.0000</td>
</tr>
<tr>
<td>0.0000  0.0000  5.0000  1.0000</td>
</tr>
<tr>
<td>-1.7321  0.3660  6.3660  1.0000</td>
</tr>
<tr>
<td>-0.8660  0.6160  5.9330  1.0000</td>
</tr>
<tr>
<td>0.0000  0.8660  5.5000  1.0000</td>
</tr>
<tr>
<td>-1.7321  1.2321  6.8660  1.0000</td>
</tr>
<tr>
<td>-0.8660  1.4821  6.4330  1.0000</td>
</tr>
<tr>
<td>0.0000  1.7321  6.0000  1.0000</td>
</tr>
<tr>
<td>0.5000  -0.4330  5.7500  1.0000</td>
</tr>
<tr>
<td>1.0000  -0.8660  6.5000  1.0000</td>
</tr>
<tr>
<td>0.5000  0.4330  6.2500  1.0000</td>
</tr>
<tr>
<td>1.0000  0.0000  7.0000  1.0000</td>
</tr>
<tr>
<td>0.5000  1.2990  6.7500  1.0000</td>
</tr>
<tr>
<td>1.0000  0.8660  7.5000  1.0000</td>
</tr>
</tbody>
</table>
Stereo reconstruction: main steps

– Calibrate cameras
– Rectify images
– Compute disparity
– Estimate depth
Correspondence problem

Multiple match hypotheses satisfy epipolar constraint, but which is correct?

Figure from Gee & Cipolla 1999
Correspondence problem

- Beyond the hard constraint of epipolar geometry, there are “soft” constraints to help identify corresponding points
  - Similarity
  - Uniqueness
  - Ordering
  - Disparity gradient

- To find matches in the image pair, we will assume
  - Most scene points visible from both views
  - Image regions for the matches are similar in appearance
Correspondence problem

Parallel camera example – epipolar lines are corresponding rasters

Source: Andrew Zisserman
Intensity profiles

- Clear correspondence between intensities, but also noise and ambiguity

Source: Andrew Zisserman
Correspondence problem

Neighborhood of corresponding points are similar in intensity patterns.

Source: Andrew Zisserman
Normalized cross correlation

subtract mean: $A \leftarrow A - \langle A \rangle, B \leftarrow B - \langle B \rangle$

$$NCC = \frac{\sum_i \sum_j A(i, j)B(i, j)}{\sqrt{\sum_i \sum_j A(i, j)^2} \sqrt{\sum_i \sum_j B(i, j)^2}}$$

Write regions as vectors

$A \rightarrow a, \ B \rightarrow b$

$$NCC = \frac{a \cdot b}{|a||b|}$$

$-1 \leq NCC \leq 1$

Source: Andrew Zisserman
Correlation-based window matching

Source: Andrew Zisserman
Dense correspondence search

For each epipolar line

For each pixel / window in the left image

- compare with every pixel / window on same epipolar line in right image
- pick position with minimum match cost (e.g., SSD, correlation)

Adapted from Li Zhang
Textureless regions

Source: Andrew Zisserman
Effect of window size

Source: Andrew Zisserman
Effect of window size

Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

Figures from Li Zhang
Foreshortening effects

fronto-parallel surface
imaged length the same

Source: Andrew Zisserman
Sparse correspondence search

- Restrict search to sparse set of detected features
- Rather than pixel values (or lists of pixel values) use feature descriptor and an associated feature distance
- Still narrow search further by epipolar geometry
Uniqueness

- For opaque objects, up to one match in right image for every point in left image
Ordering constraint

- Points on **same surface** (opaque object) will be in same order in both views

Figure from Gee & Cipolla 1999

Grauman
Ordering constraint

- Won’t always hold, e.g. consider transparent object, or an occluding surface
Disparity gradient

- Assume piecewise continuous surface, so want disparity estimates to be locally smooth

Given matches ● and ○, point ○ in the left image must match point 1 in the right image. Point 2 would exceed the disparity gradient limit.
Scanline stereo

- Try to coherently match pixels on the entire scanline
- Different scanlines are still optimized independently
“Shortest paths” for scan-line stereo

Can be implemented with dynamic programming

Ohta & Kanade ’85, Cox et al. ‘96

Slide credit: Y. Boykov
Coherent stereo on 2D grid

• Scanline stereo generates streaking artifacts

• Can’t use dynamic programming to find spatially coherent disparities/correspondences on a 2D grid
As energy minimization...

\[ E = \alpha E_{\text{data}}(I_1, I_2, D) + \beta E_{\text{smooth}}(D) \]

\[ E_{\text{data}} = \sum_i (W_1(i) - W_2(i + D(i)))^2 \]

\[ E_{\text{smooth}} = \sum_{\text{neighbors } i,j} \rho(D(i) - D(j)) \]
As energy minimization...

Energy functions of this form can be minimized using graph cuts

Y. Boykov, O. Veksler, and R. Zabih,
Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Source: Steve Seitz
Examples...

left image

right image

range map
left image

right image

depth map
intensity = depth
Z-keying for virtual reality

- Merge synthetic and real images given depth maps

Kanade et al., CMU, 1995
Z-keying for virtual reality

Kanade et al., CMU, 1995

http://www.cs.cmu.edu/afs/cs/project/stereo-machine/www/z-key.html
Virtual viewpoint video

Figure 6: Sample results from stereo reconstruction stage: (a) input color image; (b) color-based segmentation; (c) initial disparity estimates \( \hat{d}_{ij} \); (d) refined disparity estimates; (e) smoothed disparity estimates \( d_i(x) \).

Virtual viewpoint video

Massive Arabesque

http://research.microsoft.com/IVM/VVV/
Multibaseline Stereo

• Basic Approach
  – Choose a reference view
  – Use your favorite stereo algorithm BUT
    • replace two-view SSD with SSD over all baselines

• Limitations
  – Must choose a reference view
  – Visibility: select which frames to match
    [Kang, Szeliski, Chai, CVPR’01]
Choosing the Baseline

- What’s the optimal baseline?
  - Too small: large depth error
  - Too large: difficult search problem
Effect of Baseline on Estimation

Figure 2: An example scene. The grid pattern in the background has ambiguity of matching.
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.
Epipolar-Plane Images [Bolles 87]

- \url{http://www.graphics.lcs.mit.edu/~aisaksen/projects/drlf/epi/}

Lesson: Beware of occlusions

Szeliski
Active stereo with structured light

- Project “structured” light patterns onto the object
  - simplifies the correspondence problem

Li Zhang’s one-shot stereo

Szeliski
Plane Sweep Stereo

• Sweep family of planes through volume

  • each plane defines an image ⇒ composite homography
Plane Sweep Stereo

- For each depth plane
  - compute composite (mosaic) image — mean
    - compute error image — variance
  - convert to confidence and aggregate spatially
- Select winning depth at each pixel
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
framework

1. For every disparity, compute raw matching costs

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

Why use a robust function?
- occlusions, other outliers

• Can also use alternative match criteria
2. Aggregate costs spatially

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

- Here, we are using a box filter (efficient moving average implementation)
- Can also use weighted average, [non-linear] diffusion...
framework

3. Choose winning disparity at each pixel

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

4. Interpolate to \textit{sub-pixel} accuracy
Traditional Analysis

• Advantages:
  – gives detailed surface estimates
  – fast algorithms based on moving averages
  – sub-pixel disparity estimates and confidence

• Limitations:
  – narrow baseline ⇒ noisy estimates
  – fails in textureless areas
  – gets confused near occlusion boundaries
Uncalibrated case

• What if we don’t know the camera parameters?
Review: Perspective projection

\[ (x, y, z) \rightarrow (f \frac{x}{z}, f \frac{y}{z}) \]

Scene point \rightarrow Image coordinates

Thus far, in camera’s reference frame only.
Review: Camera parameters

- **Extrinsic:** location and orientation of camera frame with respect to reference frame
- **Intrinsic:** how to map pixel coordinates to image plane coordinates
Review: Extrinsic camera parameters

$$\mathbf{P}_c = R(\mathbf{P}_w - T)$$

$$(X, Y, Z)^T$$
Review: Camera parameters

• **Extrinsic:** location and orientation of camera frame with respect to reference frame

• **Intrinsic:** how to map pixel coordinates to image plane coordinates
Projection matrix for perspective projection

\[
\begin{align*}
    x &= f \frac{X}{Z} \\
    y &= f \frac{Y}{Z}
\end{align*}
\]

From pinhole camera model

Same thing, but written in terms of homogeneous coordinates

\[
\begin{bmatrix}
    x' \\
    y' \\
    z'
\end{bmatrix}
= \begin{bmatrix}
    f & 0 & 0 & 0 \\
    0 & f & 0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

\[
\begin{align*}
    x &= \frac{x'}{z'} \\
    y &= \frac{y'}{z'} \\
    x &= \frac{fx}{Z} \\
    y &= \frac{fY}{Z}
\end{align*}
\]
Review: Intrinsic camera parameters

- Ignoring any geometric distortions from optics, we can describe them by:

\[
\begin{align*}
x &= -(x_{im} - o_x) S_x \\
y &= -(y_{im} - o_y) S_y
\end{align*}
\]

- Coordinates of projected point in camera reference frame
- Coordinates of image point in pixel units
- Coordinates of image center in pixel units
- Effective size of a pixel (mm)
Review: Camera parameters

- We know that in terms of camera reference frame:

\[
x = f \frac{X}{Z} \quad y = f \frac{Y}{Z}
\]

- Substituting previous eqns describing intrinsic and extrinsic parameters, can relate \textit{pixels coordinates} to \textit{world points}:

\[
-(x_{im} - o_x) s_x = f \frac{R_1 \cdot (P_w - T)}{R_3 \cdot (P_w - T)}
\]

\[
-(y_{im} - o_y) s_y = f \frac{R_2 \cdot (P_w - T)}{R_3 \cdot (P_w - T)}
\]

\[\text{and } P_c = R(P_w - T) = (X, Y, Z)^T\]

\[R_i = \text{Row } i \text{ of rotation matrix}\]
Review: Projection matrix

- This can be rewritten as a matrix product using homogeneous coordinates:

\[
\begin{pmatrix}
wx_{im} \\
wY_{im} \\
w
\end{pmatrix}
= \mathbf{M}_{\text{int}} \mathbf{M}_{\text{ext}}
\begin{pmatrix}
X_w \\
Y_w \\
Z_w \\
1
\end{pmatrix}
\]

\[
x_{im} = \frac{wx_{im}}{w}
\]

\[
y_{im} = \frac{wy_{im}}{w}
\]

\[
\mathbf{M}_{\text{int}} = \begin{pmatrix}
-f/s_x & 0 & o_x \\
0 & -f/s_y & o_y \\
0 & 0 & 1
\end{pmatrix}
\]

\[
\mathbf{M}_{\text{ext}} = \begin{pmatrix}
r_{11} & r_{12} & r_{13} & -R_1^T \mathbf{T} \\
r_{21} & r_{22} & r_{23} & -R_2^T \mathbf{T} \\
r_{31} & r_{32} & r_{33} & -R_3^T \mathbf{T}
\end{pmatrix}
\]

point in camera coordinates
Review: Calibrating a camera

• Compute intrinsic and extrinsic parameters using observed camera data

Main idea
• Place “calibration object” with known geometry in the scene
• Get correspondences
• Solve for mapping from scene to image: estimate \( M = M_{\text{int}} M_{\text{ext}} \)
Review: Projection matrix

- This can be rewritten as a matrix product using homogeneous coordinates:

$$\\begin{pmatrix} wx_{im} \\
wy_{im} \\
w \end{pmatrix} = \begin{pmatrix} M_{\text{int}} & M_{\text{ext}} \\
\\end{pmatrix} \begin{pmatrix} X_w \\
Y_w \\
Z_w \\
1 \end{pmatrix}$$

Product $M$ is single projection matrix encoding both extrinsic and intrinsic parameters.

Let $M_i$ be row $i$ of matrix $M$.
Review: Estimating the projection matrix

For a given feature point

\[ x_{im} = \frac{M_1 \cdot P_w}{M_3 \cdot P_w} \quad \rightarrow \quad 0 = (M_1 - x_{im} M_3) \cdot P_w \]

\[ y_{im} = \frac{M_2 \cdot P_w}{M_3 \cdot P_w} \quad \rightarrow \quad 0 = (M_2 - y_{im} M_3) \cdot P_w \]
Review: Estimating the projection matrix

\[ 0 = (M_1 - x_{im} M_3) \cdot P_w \]
\[ 0 = (M_2 - y_{im} M_3) \cdot P_w \]

Expanding this first equation, we have:

\[
\begin{bmatrix}
  m_{11} & m_{12} & m_{13} & m_{14} \\
\end{bmatrix}
- x_{im} \begin{bmatrix}
  m_{31} & m_{32} & m_{33} & m_{34} \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
  X_w \\
  Y_w \\
  Z_w \\
  1 \\
\end{bmatrix}
\]

\[
= 0
\]

\[
\begin{bmatrix}
  X_w m_{11} - X_w x_{im} m_{31} \\
  Y_w m_{12} - Y_w x_{im} m_{32} \\
  Z_w m_{13} - Z_w x_{im} m_{33} \\
  m_{14} - x_{im} m_{34} \\
\end{bmatrix}
\]

\[
= 0
\]
Review: Estimating the projection matrix

\[
0 = (M_1 - x_{im} M_3) \cdot P_w \\
0 = (M_2 - y_{im} M_3) \cdot P_w
\]
Review: Estimating the projection matrix

This is true for every feature point, so we can stack up $n$ observed image features and their associated 3D points in single equation:

$$Pm = 0$$

$$\begin{bmatrix}
X^{(1)}_w & Y^{(1)}_w & Z^{(1)}_w & 1 & 0 & 0 & 0 & 0 & 0 - x^{(1)}_w X^{(1)}_w & - x^{(1)}_w Y^{(1)}_w & - x^{(1)}_w Z^{(1)}_w & - x^{(1)}_w \\
0 & 0 & 0 & 0 & X^{(1)}_w & Y^{(1)}_w & Z^{(1)}_w & 1 - y^{(1)}_w X^{(1)}_w & - y^{(1)}_w Y^{(1)}_w & - y^{(1)}_w Z^{(1)}_w & - y^{(1)}_w \\
\end{bmatrix}
\begin{bmatrix}
m_{11} \\
m_{12} \\
m_{13} \\
m_{14} \\
m_{21} \\
m_{22} \\
m_{23} \\
m_{24} \\
m_{31} \\
m_{32} \\
m_{33} \\
m_{34}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0
\end{bmatrix}$$

Solve for $m_{ij}$’s (the calibration information) [F&P Section 3.1]
Summary: camera calibration

• Associate image points with scene points on object with known geometry
• Use together with perspective projection relationship to estimate projection matrix
• (Can also solve for explicit parameters themselves)
When would we calibrate this way?

• Makes sense when geometry of system is not going to change over time

• ...When would it change?
Weak calibration

• Want to estimate world geometry without requiring calibrated cameras
  – Archival videos
  – Photos from multiple unrelated users
  – Dynamic camera system

• Main idea:
  – Estimate epipolar geometry from a (redundant) set of point correspondences between two uncalibrated cameras
Uncalibrated case

For a given camera:

\[ \mathbf{p} = \mathbf{M}_{\text{int}} \mathbf{p} \]

Camera coordinates

So, for two cameras (left and right):

\[ \mathbf{p}_{(\text{left})} = \mathbf{M}_{\text{left, int}}^{-1} \mathbf{\bar{p}}_{(\text{left})} \]

\[ \mathbf{p}_{(\text{right})} = \mathbf{M}_{\text{right, int}}^{-1} \mathbf{\bar{p}}_{(\text{right})} \]

Camera coordinates

Image pixel coordinates

Internal calibration matrices, one per camera
Uncalibrated case: fundamental matrix

\[
p_{(\text{left})} = M^{-1}_{\text{left, int}} \bar{p}_{(\text{left})}
\]

\[
p_{(\text{right})} = M^{-1}_{\text{right, int}} \bar{p}_{(\text{right})}
\]

\[
p_{(\text{right})}^T E p_{(\text{left})} = 0
\]

From before, the essential matrix \( E \).

\[
\left(M^{-1}_{\text{right, int}} \bar{p}_{\text{right}} \right)^T E \left(M^{-1}_{\text{left, int}} \bar{p}_{\text{left}} \right) = 0
\]

\[
\bar{p}_{\text{right}}^T \left(M^{-T}_{\text{right, int}} E M^{-1}_{\text{left, int}} \right) \bar{p}_{\text{left}} = 0
\]

\[
\bar{p}_{\text{right}}^T F \bar{p}_{\text{left}} = 0
\]

Fundamental matrix
Fundamental matrix

- Relates pixel coordinates in the two views
- More general form than essential matrix: we remove need to know intrinsic parameters
- If we estimate fundamental matrix from correspondences in pixel coordinates, can reconstruct epipolar geometry without intrinsic or extrinsic parameters
Computing $F$ from correspondences

\[
F = \begin{pmatrix}
M_{right,int}^{-T} & EM_{left,int}^{-1}
\end{pmatrix}
\]

\[
\bar{p}_{right}^T F \bar{p}_{left} = 0
\]

- Cameras are uncalibrated: we don’t know $E$ or left or right $M_{int}$ matrices
- Estimate $F$ from 8+ point correspondences.
Computing F from correspondences

Each point correspondence generates one constraint on F:

$$\overline{p}_{right}^T F \overline{p}_{left} = 0$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

Collect n of these constraints:

$$\begin{bmatrix} u_1' u_1 & u_1' v_1 & u_1' u_2 & v_1' u_1 & v_1' v_1 & v_1' u_2 & u_1 & v_1 & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Solve for f, vector of parameters.
Stereo pipeline with weak calibration

So, where to start with uncalibrated cameras?

Need to find fundamental matrix $F$ and the correspondences (pairs of points $(u',v') \leftrightarrow (u,v)$).

1) Find interest points in image
2) Compute correspondences
3) Compute epipolar geometry
4) Refine

Example from Andrew Zisserman
Stereo pipeline with weak calibration

1) Find interest points
Stereo pipeline with weak calibration

2) Match points only using proximity
Putative matches based on correlation search

- Many wrong matches (10-50%), but enough to compute F
Residual Error

- For each image point $p_2$ (right image), the corresponding point $p_1$ in the other image (for ex. In left image) should ideally lie exactly on the epipolar line $l = F \cdot p_2$

- If there is noise, the residual error = distance from the actual point $p_1$ to the epipolar line

- Distance from point $p_1 = (x_1, y_1, 1)^T$ to line with parameters $l = (a, b, c)^T$ is

$$d = \frac{\text{abs}(p_1^T \cdot l)}{\sqrt{a^2 + b^2}}$$
RANSAC for robust estimation of the fundamental matrix

- Select random sample of correspondences
- Compute F using them
  - This determines epipolar constraint
- Evaluate amount of support – inliers within threshold distance of epipolar line
- Choose F with most support (inliers)

Matlab examples at:
http://www.peterkovesi.com/matlabfns/index.html
Putative matches based on correlation search

- Many wrong matches (10-50%), but enough to compute F
Pruned matches

- Correspondences consistent with epipolar geometry
• Resulting epipolar geometry
Slide Credits

• Kristen Grauman for most,
• Rick Szeliski and others as noted...