CS4501: Introduction to Computer Vision

3D Vision: Camera Calibration and Dense Stereo

Various slides from previous courses by:
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL), S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook), J. M. Frahm (UNC), V. Ordonez (UVA), Steve Seitz (UW).
Today’s Class

• Camera Calibration
• Stereo Vision – Dense Stereo / Stereo Matching
Camera Calibration

• What does it mean?
Recall the Projection matrix

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

\(\mathbf{x}\): Image Coordinates: \((u,v,1)\)

\(\mathbf{K}\): Intrinsic Matrix (3x3)

\(\mathbf{R}\): Rotation (3x3)

\(\mathbf{t}\): Translation (3x1)

\(\mathbf{X}\): World Coordinates: \((X,Y,Z,1)\)
Recall the Projection matrix

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

# Definition of the faces of the cube.
cube_pts = np.array(
    [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],  # Face 1.
     [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],  # Face 2.
     [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],  # Face 3.
     [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])  # Face 4.

# Intrinsic Camera Matrix.
f = 3.0 # focal length.
K = np.array([[f, 0, 0],
              [0, f, 0],
              [0, 0, 1]])

# Extrinsic Camera Parameters.
Rt = np.array([[1, 0, 0, 1],
               [0, 1, 0, 1],
               [0, 0, 1, 4]])

# Camera matrix.
Camera_matrix = np.dot(K, Rt)
Recall the Projection matrix

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

Goal: Find \( X \)
Camera Calibration

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

```python
# Definition of the faces of the cube.
cube_pts = np.array(
    [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],
     [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],
     [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],
     [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]])  # Face 1.
```

\[ X = \]

```
```

\[ X = \]
Camera Calibration

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

Goal: Find \( K[R \ t] \)

X =

# Definition of the faces of the cube.
cube_pts = np.array(
    [[[0,0,0], [0,0,1], [0,1,1], [0,1,0], [0,0,0]],  # Face 1.
     [[0,0,0], [0,1,0], [1,1,0], [1,0,0], [0,0,0]],  # Face 2.
     [[1,0,0], [1,0,1], [1,1,1], [1,1,0], [1,0,0]],  # Face 3.
     [[0,0,1], [0,1,1], [1,1,1], [1,0,1], [0,0,1]]])  # Face 4.
Calibrating the Camera

Use an scene with known geometry
  • Correspond image points to 3d points
  • Get least squares solution (or non-linear solution)

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

Known 2d image coords

Known 3d locations

<table>
<thead>
<tr>
<th>Known 2d image coords</th>
<th>Known 3d locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>880  214</td>
<td>312.747 309.140 30.086</td>
</tr>
<tr>
<td>43   203</td>
<td>305.796 311.649 30.356</td>
</tr>
<tr>
<td>270  197</td>
<td>307.694 312.358 30.418</td>
</tr>
<tr>
<td>886  347</td>
<td>310.149 307.186 29.298</td>
</tr>
<tr>
<td>745  302</td>
<td>311.937 310.105 29.216</td>
</tr>
<tr>
<td>943  128</td>
<td>311.202 307.572 30.682</td>
</tr>
<tr>
<td>476  590</td>
<td>307.106 306.876 28.660</td>
</tr>
<tr>
<td>419  214</td>
<td>309.317 312.490 30.230</td>
</tr>
<tr>
<td>317  335</td>
<td>307.435 310.151 29.318</td>
</tr>
<tr>
<td>783  521</td>
<td>308.253 306.300 28.881</td>
</tr>
<tr>
<td>235  427</td>
<td>306.650 309.301 28.905</td>
</tr>
<tr>
<td>665  429</td>
<td>308.069 306.831 29.189</td>
</tr>
<tr>
<td>655  362</td>
<td>309.671 308.834 29.029</td>
</tr>
<tr>
<td>427  333</td>
<td>308.255 309.955 29.267</td>
</tr>
<tr>
<td>412  415</td>
<td>307.546 308.613 28.963</td>
</tr>
<tr>
<td>746  351</td>
<td>311.036 312.709 30.514</td>
</tr>
<tr>
<td>434  415</td>
<td>311.988 312.709 30.514</td>
</tr>
<tr>
<td>525  234</td>
<td>312.160 310.772 29.080</td>
</tr>
<tr>
<td>716  308</td>
<td>311.988 312.709 30.514</td>
</tr>
<tr>
<td>602  187</td>
<td>311.988 312.709 30.514</td>
</tr>
</tbody>
</table>
Unknown Camera Parameters

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

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

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

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$
Unknown Camera Parameters

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

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

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

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

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

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

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

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

\[
m_{31} uX + m_{32} uY + m_{33} uZ + m_{34} u = m_{11} X + m_{12} Y + m_{13} Z + m_{14}
\]
\[
m_{31} vX + m_{32} vY + m_{33} vZ + m_{34} v = m_{21} X + m_{22} Y + m_{23} Z + m_{24}
\]

\[
0 = m_{11} X + m_{12} Y + m_{13} Z + m_{14} - m_{31} uX - m_{32} uY - m_{33} uZ - m_{34} u
\]
\[
0 = m_{21} X + m_{22} Y + m_{23} Z + m_{24} - m_{31} vX - m_{32} vY - m_{33} vZ - m_{34} v
\]
Unknown Camera Parameters

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

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

\[
0 = m_{21}X + m_{22}Y + m_{23}Z + m_{24}vX - m_{31}vY - m_{33}vZ - m_{34}v
\]

• Method 1 – homogeneous linear system. Solve for m’s entries using linear least squares

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

\[
\begin{bmatrix}
    m_{11} \\
    m_{12} \\
    m_{13} \\
    m_{14} \\
    m_{21} \\
    m_{22} \\
    m_{23} \\
    m_{24} \\
    m_{31} \\
    m_{32} \\
    m_{33} \\
    m_{34}
\end{bmatrix} = \begin{bmatrix}
    0 \\
    0 \\
    \vdots \\
    0 \\
    0
\end{bmatrix}
\]

\[
[U, S, V] = \text{svd}(A);
\]

\[
M = V(:,\text{end})
\]

\[
M = \text{reshape}(M,[],3)'
\]
Method 2 – nonhomogeneous linear system. Solve for m’s entries using linear least squares.

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

Ax=b form

\[
\begin{bmatrix}
    X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\
    0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \\
    \vdots
    \end{bmatrix}
\begin{bmatrix}
    m_{11} \\
    m_{12} \\
    m_{13} \\
    m_{14} \\
    m_{21} \\
    m_{22} \\
    m_{23} \\
    m_{24} \\
    m_{31} \\
    m_{32} \\
    m_{33}
\end{bmatrix}
= \begin{bmatrix}
    u_i \\
    v_i \\
    \vdots
\end{bmatrix}
\]

\[
M = A\backslash Y;
\]

\[
M = [M; 1];
\]

\[
M = \text{reshape}(M, [], 3)';
\]
Can we factorize $M$ back to $K \ [R \ | \ T]$?

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\
m_{21} & m_{22} & m_{23} & m_{24} \\
m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = K[R \ t]$$

• Yes!

• You can use $RQ$ factorization (note – not the more familiar $QR$ factorization). $R$ (right diagonal) is $K$, and $Q$ (orthogonal basis) is $R$. $T$, the last column of $[R \ | \ T]$, is $\text{inv}(K) \ast$ 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/

Credit: James Hays
Stereo:
Epipolar geometry

Vicente Ordonez
University of Virginia
Multiple views

Multi-view geometry, matching, invariant features, stereo vision

Hartley and Zisserman

Lowe
Why multiple views?

• Structure and depth are inherently ambiguous from single views.

Images from Lana Lazebnik
Why multiple views?

- Structure and depth are inherently ambiguous from single views.
• What cues help us to perceive 3d shape and depth?
Shading

[Figure from Prados & Faugeras 2006]
Focus/defocus

Images from same point of view, different camera parameters

3d shape / depth estimates

[figs from H. Jin and P. Favaro, 2002]
Texture

Perspective effects
Motion

Figures from L. Zhang

http://www.brainconnection.com/teasers/?main=illusion/motion-shape
Estimating scene shape

• “Shape from X”: Shading, Texture, Focus, Motion...

• Stereo:
  • shape from “motion” between two views
  • infer 3d shape of scene from two (multiple) images from different viewpoints

Main idea:
Human eye

Rough analogy with human visual system:

- Pupil/Iris – control amount of light passing through lens
- Retina - contains sensor cells, where image is formed
- Fovea – highest concentration of cones

Fig from Shapiro and Stockman
Human stereopsis: disparity

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.
Disparity occurs when eyes fixate on one object; others appear at different visual angles.
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
Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923
Exploit disparity as depth cue using single image.

(Single image random dot stereogram, Single image stereogram)
Estimating depth with stereo

- **Stereo**: shape from “motion” between two views
- We’ll need to consider:
  - Info on camera pose (“calibration”)
  - Image point correspondences
Key idea: Epipolar constraint

Potential matches for $x$ have to lie on the corresponding line $l'$. 

Potential matches for $x'$ have to lie on the corresponding line $l$. 
Epipolar geometry: notation

- **Baseline** – line connecting the two camera centers
- **Epipoles**
  - intersections of baseline with image planes
  - projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)
Epipolar geometry: notation

- **Baseline** – line connecting the two camera centers
- **Epipoles**
  = intersections of baseline with image planes
  = projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)
Example: Converging cameras
Geometry for a simple stereo system

• First, assuming parallel optical axes, known camera parameters (i.e., calibrated cameras):
Simplest Case: Parallel images

- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images
Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). **What is expression for \( Z \)?**

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
Depth from disparity

If we could find the corresponding points in two images, we could estimate relative depth...
Correspondence search

- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation
Correspondence search

Left

Right

scanline

SSD
Correspondence search

Left

Right

scanline

Norm. corr
Basic stereo matching algorithm

• If necessary, rectify the two stereo images to transform epipolar lines into scanlines

• For each pixel $x$ in the first image
  • Find corresponding epipolar scanline in the right image
  • Examine all pixels on the scanline and pick the best match $x'$
  • Compute disparity $x-x'$ and set $\text{depth}(x) = B*f/(x-x')$
Failures of correspondence search

Textureless surfaces

Occlusions, repetition

Non-Lambertian surfaces, specularities
Effect of window size

- Smaller window
  - More detail
  - More noise

- Larger window
  - Smoother disparity maps
  - Less detail
Results with window search

Data

Window-based matching

Ground truth
Better methods exist...


For the latest and greatest: [http://www.middlebury.edu/stereo/](http://www.middlebury.edu/stereo/)
When cameras are not aligned: 
Stereo image rectification

- Reproject image planes onto a common plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each input image reprojection

Rectification example
Questions?