Stereo
Overview of today’s lecture

• Revisiting triangulation.

• Disparity.

• Stereo rectification.

• Stereo matching.

• Improving stereo matching.

• Structured light.

Slide credits: Some of these slides were adapted directly from:

• Kris Kitani (16-385, Spring 2017).
• Srinivasa Narasimhan (16-823, Spring 2017).
Revisiting triangulation
How would you reconstruct 3D points?

Left image

Right image
How would you reconstruct 3D points?

1. Select point in one image (how?)
How would you reconstruct 3D points?

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
How would you reconstruct 3D points?

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)
How would you reconstruct 3D points?

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)
4. Perform triangulation (how?)
Triangulation

3D point

left image

left camera with matrix

right image

right camera with matrix

$P$

$P'$
How would you reconstruct 3D points?

1. Select point in one image
2. Form epipolar line for that point in second image
3. Find matching point along line (how?)
4. Perform triangulation

Simplify the search process
Stereo rectification
What’s different between these two images?
Objects that are close move more or less?
The amount of horizontal movement is inversely proportional to …
The amount of horizontal movement is inversely proportional to …

… the distance from the camera.

More formally…
3D point

camera center

O

camera center

O′

image plane
image plane
How is $X$ related to $x$?
\[
\frac{X}{Z} = \frac{x}{f}
\]
\[ \frac{X}{Z} = \frac{x}{f} \]

How is X related to x'?
\[
\frac{X}{Z} = \frac{x}{f}
\]

\[
\frac{b - X}{Z} = \frac{x'}{f}
\]
\[
\frac{X}{Z} = \frac{x}{f}
\]

\[
\frac{b - X}{Z} = \frac{x'}{f}
\]

Disparity

\[
d = x + x' \\
= \frac{bf}{Z} \quad \text{(wrt to camera origin of image plane)}
\]
Disparity

d = x + x' 
= \frac{bf}{Z} 

inversely proportional to depth

\frac{X}{Z} = \frac{x}{f} 

\frac{b - X}{Z} = \frac{x'}{f}
Nomad robot searches for meteorites in Antartica

http://www.frc.ri.cmu.edu/projects/meteorobot/index.html
Subaru Eyesight system

Pre-collision braking
What other vision system uses disparity for depth sensing?
Stereoscopes: A 19\textsuperscript{th} Century Pastime
HON. ABRAHAM LINCOLN, President of United States.
Mark Twain at Pool Table*, no date, UCR Museum of Photography
This is how 3D movies work
Let’s think about how to compute depth from two images of the same object
1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity
      \[ Z = \frac{bf}{d} \]
How can you make the epipolar lines horizontal?
What’s special about these two cameras?
When are epipolar lines horizontal?

When this relationship holds:

\[ R = I \quad t = (T, 0, 0) \]
When are epipolar lines horizontal?

When this relationship holds:

\[
R = I \quad t = (T, 0, 0)
\]

Let's try this out…

\[
E = t \times R = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & -T \\
0 & T & 0
\end{bmatrix}
\]

This always has to hold for rectified images

\[
x^T E x' = 0
\]
**When are epipolar lines horizontal?**

When this relationship holds:

\[
R = I \quad t = (T, 0, 0)
\]

Let's try this out...

\[
E = t \times R = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & -T \\
0 & T & 0
\end{bmatrix}
\]

This always has to hold for rectified images

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

Write out the constraint

\[
\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & -T \\
0 & T & 0
\end{bmatrix} \begin{pmatrix} u' \\
v' \\
1
\end{pmatrix} = 0
\]

\[
\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\
-T \\
Tv'
\end{pmatrix} = 0
\]
When are epipolar lines horizontal?

Write out the constraint

\[
\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0
\]

When this relationship holds:

\[
R = I \quad t = (T, 0, 0)
\]

Let's try this out…

\[
E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}
\]

This always has to hold

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

The image of a 3D point will always be on the same horizontal line.

\[
Tv = Tv'
\]

y coordinate is always the same!
It’s hard to make the image planes exactly parallel
How can you make the epipolar lines horizontal?
Use stereo rectification?
What is stereo rectification?
What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

How can you do this?
What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

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

Stereo Rectification

1. **Rotate** the right camera by $R$
   (aligns camera coordinate system orientation only)

2. Rotate (**rectify**) the left camera so that the epipole is at infinity

3. Rotate (**rectify**) the right camera so that the epipole is at infinity

4. Adjust the **scale**
Stereo Rectification:

1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
Stereo Rectification:

1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
Stereo Rectification:

1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
Stereo Rectification:

1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
Stereo Rectification:

1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
1. Compute \( E \) to get \( R \)
2. Rotate right image by \( R \)
3. Rotate both images by \( R_{\text{rect}} \)
4. Scale both images by \( H \)
Stereo Rectification:

1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
Step 1: Compute $E$ to get $R$

SVD: $ \mathbf{E} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top$

Let $ \mathbf{w} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

We get FOUR solutions:

$ \mathbf{R}_1 = \mathbf{U} \mathbf{w} \mathbf{V}^\top$

$ \mathbf{R}_2 = \mathbf{U} \mathbf{w}^\top \mathbf{V}^\top$

$ \mathbf{T}_1 = \mathbf{U}_3$

$ \mathbf{T}_2 = -\mathbf{U}_3$

Two possible rotations

Two possible translations
We get FOUR solutions:

\[ R_1 = UWV^T \]
\[ T_1 = U_3 \]

\[ R_2 = UW^T V^T \]
\[ T_2 = -U_3 \]

\[ R_1 = UWV^T \]
\[ T_2 = -U_3 \]

\[ R_2 = UW^T V^T \]
\[ T_1 = U_3 \]

Which one do we choose?

Compute determinant of R, valid solution must be equal to 1
\textit{(note: } \text{det}(R) = -1 \text{ means rotation and reflection)}

Compute 3D point using triangulation, valid solution has positive Z value
\textit{(Note: negative Z means point is behind the camera)}
Positive depth test

- $E \rightarrow R_1, T_1$

$P_1 = K_1 \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, (R = I, T = 0)$

$P_2 = K_2 \begin{pmatrix} R_1 & T_1 \end{pmatrix}, (R = R_1, T = T_1)$

Perform triangulation, with $P_1, P_2$ and $x_1, x_2$

3D coordinate $X$

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ X \\ 1 \\ \cdot \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Check if positive
Let’s visualize the four configurations…

Find the configuration where the points is in front of both cameras
Find the configuration where the points is in front of both cameras
Find the configuration where the points is in front of both cameras
1. Compute $E$ to get $R$
2. Rotate right image by $R$
3. Rotate both images by $R_{\text{rect}}$
4. Scale both images by $H$
When do epipolar lines become horizontal?
Parallel cameras

Where is the epipole?
Parallel cameras

epipole at infinity
Setting the epipole to infinity

(Building $\mathbf{R}_{\text{rect}}$ from $\mathbf{e}$)

Let $\mathbf{R}_{\text{rect}} = \begin{bmatrix} \mathbf{r}_1^\top \\ \mathbf{r}_2^\top \\ \mathbf{r}_3^\top \end{bmatrix}$

Given:
- epipole $\mathbf{e}$ (using SVD on $\mathbf{E}$)
- (translation from $\mathbf{E}$)

$\mathbf{r}_1 = \mathbf{e}_1 = \frac{T}{||T||}$

epipole coincides with translation vector

$r2 = r1 \times \{\text{previous optical axis}\}$

cross product of $\mathbf{e}$ and the direction vector of the optical axis

$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$

orthogonal vector
If \( r_1 = e_1 = \frac{T}{\|T\|} \) and \( r_2, r_3 \) orthogonal

then \( R_{\text{rect}} e_1 = \begin{bmatrix} r_1^\top e_1 \\ r_2^\top e_1 \\ r_3^\top e_1 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix} \)
If \( r_1 = e_1 = \frac{T}{\|T\|} \) and \( r_2 \quad r_3 \) orthogonal

then \( R_{\text{rect}} e_1 = \begin{bmatrix} r_1^T e_1 \\ r_2^T e_1 \\ r_3^T e_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \)

Where is this point located on the image plane?
If \( r_1 = e_1 = \frac{T}{||T||} \) and \( r_2 \quad r_3 \) orthogonal

\[
\begin{align*}
\text{then} \quad R_{\text{rect}} e_1 &= \begin{bmatrix}
  r_1^T e_1 \\
  r_2^T e_1 \\
  r_3^T e_1 
\end{bmatrix} = \begin{bmatrix}
  1 \\
  0 \\
  0 
\end{bmatrix}
\end{align*}
\]

*Where is this point located on the image plane?*

*At x-infinity*
Stereo Rectification Algorithm

1. Estimate $\mathbf{E}$ using the 8 point algorithm (SVD)
2. Estimate the epipole $\mathbf{e}$ (SVD of $\mathbf{E}$)
3. Build $\mathbf{R}_{\text{rect}}$ from $\mathbf{e}$
4. Decompose $\mathbf{E}$ into $\mathbf{R}$ and $\mathbf{T}$
5. Set $\mathbf{R}_1 = \mathbf{R}_{\text{rect}}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{\text{rect}}$
6. Rotate each left camera point (warp image)
   $$[x' \ y' \ z'] = \mathbf{R}_1 [x \ y \ z]$$
7. Rectified points as $\mathbf{p} = f/z' [x' \ y' \ z']$
8. Repeat 6 and 7 for right camera points using $\mathbf{R}_2$
What can we do after rectification?
Stereo matching
Depth Estimation via Stereo Matching
1. Rectify images  
   (make epipolar lines horizontal)  
2. For each **pixel**  
   a. Find epipolar line  
   b. Scan line for best match  
   c. Compute depth from disparity  

\[ Z = \frac{bf}{d} \]
Reminder from filtering

How do we detect an edge?
Reminder from filtering

How do we detect an edge?
• We filter with something that looks like an edge.

We can think of linear filtering as a way to evaluate how similar an image is locally to some template.
Find this template

How do we detect the template in the following image?
Find this template

How do we detect the template in the following image?

Solution 1: Filter the image using the template as filter kernel.

$$h[m, n] = \sum_{k,j} g[k, l] f[m + k, n + l]$$
Find this template

How do we detect the template in the following image?

Solution 1: Filter the image using the template as filter kernel.

What went wrong?
Find this template

How do we detect the template in the following image?

Solution 1: Filter the image using the template as filter kernel.

\[
h[m, n] = \sum_{k,j} g[k, l] f[m + k, n + l]
\]

Increases for higher local intensities.
How do we detect the template in the following image?

Solution 2: Filter the image using a zero-mean template.
Find this template

How do we detect the template in the following image?

Solution 2: Filter the image using a zero-mean template.

What went wrong?
Find this template

How do we detect the template in the following image?

Solution 2: Filter the image using a zero-mean template.

\[ h[m, n] = \sum_{k,l} (g[k, l] - \bar{g}) f[m + k, n + l] \]

Not robust to high-contrast areas
How do we detect the template in the following image?

Solution 3: Use sum of squared differences (SSD).

\[ h[m, n] = \sum_{k,l} (g[k, l] - f[m + k, n + l])^2 \]
Find this template

How do we detect the template in the following image?

Solution 3: Use sum of squared differences (SSD).

\[ h[m, n] = \sum_{k,l} (g[k, l] - f[m + k, n + l])^2 \]
Find this template

How do we detect the template in the following image?

Solution 3: Use sum of squared differences (SSD).

\[ h[m, n] = \sum_{k,l} (g[k, l] - f[m + k, n + l])^2 \]

Not robust to local intensity changes.
Find this template

How do we detect the template in the following image?

Observations so far:

• subtracting mean deals with brightness bias

• dividing by standard deviation removes contrast bias

Can we combine the two effects?
Find this template

How do we detect the template in the following image?

What will the output look like?

Solution 4: Normalized cross-correlation (NCC).
Find this template

How do we detect the template in the following image?

1-output

thresholding

Solution 4: Normalized cross-correlation (NCC).
Find this template

How do we detect the template in the following image?

Solution 4: Normalized cross-correlation (NCC).
What is the best method?

It depends on whether you care about speed or invariance.

• Zero-mean: Fastest, very sensitive to local intensity.

• Sum of squared differences: Medium speed, sensitive to intensity offsets.

• Normalized cross-correlation: Slowest, invariant to contrast and brightness.
Stereo Block Matching

- Slide a window along the epipolar line and compare contents of that window with the reference window in the left image.
- Matching cost: SSD or normalized correlation.
Normalized cross-correlation
Similarity Measure

- **Sum of Absolute Differences (SAD)**

- **Sum of Squared Differences (SSD)**

- **Zero-mean SAD**

- **Locally scaled SAD**

- **Normalized Cross Correlation (NCC)**

**Formula**

\[
\begin{align*}
\text{SAD:} & \quad \sum_{(i,j) \in W} |I_1(i,j) - I_2(x + i, y + j)| \\
\text{SSD:} & \quad \sum_{(i,j) \in W} (I_1(i,j) - I_2(x + i, y + j))^2 \\
\text{Zero-mean SAD:} & \quad \sum_{(i,j) \in W} |I_1(i,j) - \overline{I_1(i,j)} - I_2(x + i, y + j) + \overline{I_2(x + i, y + j)}| \\
\text{Locally scaled SAD:} & \quad \sum_{(i,j) \in W} |I_1(i,j) - \frac{\overline{I_1(i,j)}}{I_2(x + i, y + j)} I_2(x + i, y + j)| \\
\text{NCC:} & \quad \frac{\sum_{(i,j) \in W} I_1(i,j) \cdot I_2(x + i, y + j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i,j) \cdot \sum_{(i,j) \in W} I_2^2(x + i, y + j)}}
\end{align*}
\]
Effect of window size

W = 3

W = 20
Effect of window size

W = 3
- More detail
- More noise

W = 20
- Smoother disparity maps
- Less detail
- Fails near boundaries
When will stereo block matching fail?
When will stereo block matching fail?

- Textureless regions
- Repeated patterns
- Specularities
Improving stereo matching
What are some problems with the result?
How can we improve depth estimation?
How can we improve depth estimation?

Too many discontinuities.
We expect disparity values to change slowly.

Let’s make an assumption:
**depth should change smoothly**
Stereo matching as …

Energy Minimization

What defines a good stereo correspondence?

1. **Match quality**
   - Want each pixel to find a good match in the other image

2. **Smoothness**
   - If two pixels are adjacent, they should (usually) move about the same amount
energy function
(for one pixel)

\[ E(d) = E_d(d) + \lambda E_s(d) \]

- **data term**: Want each pixel to find a good match in the other image (block matching result)
- **smoothness term**: Adjacent pixels should (usually) move about the same amount (smoothness function)
Stereo as energy minimization

\[ I(x, y) \quad J(x, y) \]

\[ C(x, y, d); \text{the disparity space image (DSI)} \]
Stereo as energy minimization

Simple pixel / window matching: choose the minimum of each column in the DSI independently:

\[ d(x, y) = \arg \min_{d'} C(x, y, d') \]
Stereo as energy minimization

\[ y = 141 \]

\[ d \]

\[ x \]
Stereo as energy minimization

\[ y = 141 \]
Stereo as energy minimization

\[ y = 141 \]
Stereo as energy minimization

Better objective function

\[ E(d) = E_d(d) + \lambda E_s(d) \]

- **match cost**: Want each pixel to find a good match in the other image.
- **smoothness cost**: Adjacent pixels should (usually) move about the same amount.
Stereo as energy minimization

\[ E(d) = E_d(d) + \lambda E_s(d) \]

match cost:

\[ E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y)) \]
Stereo as energy minimization

\[ E(d) = E_d(d) + \lambda E_s(d) \]

match cost: \[ E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y)) \]

smoothness cost: \[ E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q) \]

\[ \mathcal{E} : \text{set of neighboring pixels} \]

Left and right
Smoothness cost

\[ E_S(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q) \]

How do we choose \( V \)?

\[ V(d_p, d_q) = |d_p - d_q| \]

\[ V(d_p, d_q) = \begin{cases} 
0 & \text{if } d_p = d_q \\
1 & \text{if } d_p \neq d_q 
\end{cases} \]

“Potts model”
Stereo as energy minimization

\[ y = 141 \]
Stereo as energy minimization

\[ y = 141 \]
Stereo as energy minimization
Dynamic programming

\[ E(d) = E_d(d) + \lambda E_s(d) \]

Can minimize this independently per scanline using dynamic programming (DP)

\[ D(x, d) : \text{minimum cost of solution such that } d(x,y) = d \]

\[ D(x, d) = C'(x, y, d) + \min_{d'} \{ D(x - 1, d') + \lambda |d - d'| \} \]
Dynamic programming

\[ y = 141 \]
Dynamic programming

Cost of optimal path from the left
How to compute for this pixel?
Dynamic programming

How to compute for this pixel?
Only 3 ways to get here...
Dynamic programming

This cost is ...
Dynamic programming

This cost is ...

\[ y = 141 \]

\[ x \]
Dynamic programming

Fill D all the way to the right

$y = 141$
Dynamic programming

$x = 141$

Pick the best one from the right column
Dynamic programming

Back track....
Dynamic programming

Back track....
Stereo as energy minimization

\[ y = 141 \]
Dynamic Programming
Dynamic programming is optimal…
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
Stereo as energy minimization

\[ E(d) = E_d(d) + \lambda E_s(d) \]

match cost:
\[ E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y)) \]

smoothness cost:
\[ E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q) \]

\[ \mathcal{E} : \text{set of neighboring pixels} \]

Left and right
Stereo as energy minimization

\[ E(d) = E_d(d) + \lambda E_s(d) \]

match cost:

\[ E_d(d) = \sum_{(x,y) \in I} C(x,y,d(x,y)) \]

smoothness cost:

\[ E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p,d_q) \]

\( \mathcal{E} \) : set of neighboring pixels

4-connected neighborhood
Pixelwise
Pixelwise -> Scanline wise

$y = 141$
Pixelwise -> Scanline wise

\[ y = 141 \]
Pixelwise -> Scanline wise

$y = 141$
Pixelwise
Stereo as energy minimization

\[ E(d) = E_d(d) + \lambda E_s(d) \]

match cost: \[ E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y)) \]

smoothness cost: \[ E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q) \]

\( \mathcal{E} \): set of neighboring pixels

\[ \mathcal{E} \]: set of neighboring pixels

4-connected neighborhood

8-connected neighborhood
Smoothness cost

\[ E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q) \]

How do we choose \( V \)?

\[ V(d_p, d_q) = |d_p - d_q| \]

\( L_1 \) distance
Stereo matching as energy minimization

- Graph-cuts can be used to minimize such energy

Same formulation with more images

• Change label from disparity to depth
• Change $E_d(d)$ by using more images
Same formulation with more images

- Change label from disparity to depth
- Change $E_d(d)$ by using more images
Same formulation with more images

- Change label from disparity to depth
- Change $E_d(d)$ by using more images
Same formulation with more images

- Change label from disparity to depth
- Change $E_d(d)$ by using more images
With more images, engineering and tricks
Structured light
Use controlled ("structured") light to make correspondences easier.

Disparity between laser points on the same scanline in the images determines the 3-D coordinates of the laser point on object.
Use controlled ("structured") light to make correspondences easier
Structured light and two cameras

laser
Structured light and one camera

Projector acts like “reverse” camera
Example: Laser scanner

Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/
The Digital Michelangelo Project, Levoy et al.
The Digital Michelangelo Project, Levoy et al.
The Digital Michelangelo Project, Levoy et al.
SIGGRAPH Talks 2011

KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction

Shahram Izadi 1, Richard Newcombe 2, David Kim 1,3, Otmar Hilliges 1,4, David Molyneaux 1,4, Pushmeet Kohli 1, Jamie Shotton 1, Steve Hodges 1, Dustin Freeman 5, Andrew Davison 2, Andrew Fitzgibbon 2

1 Microsoft Research Cambridge  2 Imperial College London  3 Newcastle University  4 Lancaster University  5 University of Toronto
References

Basic reading:
• Szeliski textbook, Section 8.1 (not 8.1.1-8.1.3), Chapter 11, Section 12.2.
• Hartley and Zisserman, Section 11.12.