Reconstruction

16-385 Computer Vision
Carnegie Mellon University (Kris Kitani)
<table>
<thead>
<tr>
<th></th>
<th>Structure (scene geometry)</th>
<th>Motion (c)</th>
<th>Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pose Estimation</td>
<td>known</td>
<td>estimate</td>
<td>3D to 2D correspondences</td>
</tr>
<tr>
<td>Triangulation</td>
<td>estimate</td>
<td>known</td>
<td>2D to 2D correspondences</td>
</tr>
<tr>
<td>Reconstruction</td>
<td>estimate</td>
<td>estimate</td>
<td>2D to 2D correspondences</td>
</tr>
</tbody>
</table>
Reconstruction
(2 view structure from motion)

Given a set of matched points
\[ \{ x_i, x'_i \} \]

Estimate the camera matrices
\[ P, P' \]

Estimate the 3D point
\[ X \]
1. Compute the Fundamental Matrix $F$ from points correspondences
   
   8-point algorithm

   $$x'_m^T F x_m = 0$$
Procedure for Reconstruction

1. Compute the Fundamental Matrix $F$ from points correspondences
   
   **8-point algorithm**

2. Compute the camera matrices $P$ from the Fundamental matrix

   $P = [ \mathbf{I} \mid \mathbf{0}]$ and $P' = [ [\mathbf{e}_x]^T]F \mid \mathbf{e}'$
Camera matrices corresponding to the fundamental matrix $\mathbf{F}$ may be chosen as

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \quad \mathbf{P}' = \begin{bmatrix} [e \times] \mathbf{F} & \mathbf{e}' \end{bmatrix}$$

(See Hartley and Zisserman C.9 for proof)
Decomposing $F$ into $R$ and $T$

If we have calibrated cameras we have $K$ and $K'$

Essential matrix: $E = K'^\top FK$
Decomposing $\mathbf{F}$ into $\mathbf{R}$ and $\mathbf{T}$

If we have calibrated cameras we have $\mathbf{K}$ and $\mathbf{K}'$

Essential matrix: $\mathbf{E} = \mathbf{K}'^\top \mathbf{F} \mathbf{K}$

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

Let $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Decomposing $F$ into $R$ and $T$

If we have calibrated cameras we have $K$ and $K'$

Essential matrix: $E = K'^\top FK$

SVD: $E = U\Sigma V^\top$

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

We get FOUR solutions:

$E = [R|T]$

$R_1 = UWV^\top$ $R_2 = UW^\top V^\top$ $T_1 = U_3$ $T_2 = -U_3$

two possible rotations two possible translations
We get FOUR solutions:

\[
\begin{align*}
R_1 &= UWV^T \\
T_1 &= U_3 \\
R_2 &= UW^T V^T \\
T_2 &= -U_3
\end{align*}
\]

Which one do we choose?

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

Compute 3D point using triangulation, valid solution has positive \( Z \) value  
\((\text{Note: negative } Z \text{ means point is behind the camera })\)
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.
From points correspondences to camera displacement

1. Normalize the image points \( \mathbf{x}, \mathbf{x}' \) using \( \mathbf{K}, \mathbf{K}' \)

2. Use the 8-point algorithm to find an approximation of \( \mathbf{E} \) (SVD!)

3. Project \( \mathbf{E} \) to essential space (SVD!!)

4. Recover possible solutions for \( \mathbf{R} \) and \( \mathbf{T} \) (SVD!!!)

5. Use point correspondence to find the correct \( \mathbf{R}, \mathbf{T} \) pair (don’t use SVD...)
Procedure for Reconstruction

1. Compute the Fundamental Matrix $F$ from points correspondences
   8-point algorithm

2. Compute the camera matrices $P$ from the Fundamental matrix
   
   $P = \begin{bmatrix} I & 0 \end{bmatrix}$ and $P' = \begin{bmatrix} [e'x]F & e' \end{bmatrix}$

3. For each point correspondence, compute the point $X$ in 3D space (triangularization)
   
   DLT with $x = PX$ and $x' = P'X$
Projective Ambiguity

- Reconstruction is ambiguous by an arbitrary 3D projective transformation without prior knowledge of camera parameters
Calibrated cameras
(similarity projection ambiguity)

Uncalibrated cameras
(projective projection ambiguity)
<table>
<thead>
<tr>
<th></th>
<th>Structure (scene geometry)</th>
<th>Motion (c)</th>
<th>Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pose Estimation</strong></td>
<td>known</td>
<td>estimate</td>
<td>3D to 2D correspondences</td>
</tr>
<tr>
<td><strong>Triangulation</strong></td>
<td>estimate</td>
<td>known</td>
<td>2D to 2D correspondences</td>
</tr>
<tr>
<td><strong>Reconstruction</strong></td>
<td>estimate</td>
<td>estimate</td>
<td>2D to 2D correspondences</td>
</tr>
</tbody>
</table>
Stereo Vision

16-385 Computer Vision
Carnegie Mellon University (Kris Kitani)
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.

... the distance from the camera.
The diagram illustrates the concept of perspective projection in an image plane. Points $x$ and $x'$ in three-dimensional space ($X$) project to points $O$ and $O'$ on the image plane, respectively. The focal length $f$ is indicated by the dotted line connecting the image plane to the points $x$ and $x'$. The image plane is annotated with the label "image plane."
\[
\frac{X}{Z} = \frac{x}{f}
\]
\[
\frac{X}{Z} = \frac{x}{f}
\]

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

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

Disparity

$$d = x - x'$$

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

Disparity

\[
d = x - x'
\]
\[
d = \frac{bf}{f} = \frac{bf}{Z}
\]

Disparity is inversely proportional to depth.
Nomad robot searches for meteorites in Antarctica

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

Pre-collision braking
How so you compute depth from a stereo pair?
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?
It’s hard to make the image planes exactly parallel
How can you make the epipolar lines horizontal?

When this relationship holds

$$R = I \quad t = (T, 0, 0)$$
How can you make the 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

\[ x^T E x' = 0 \]
How can you make the 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

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

Write out the constraint

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

\[
\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0
\]
How can you make the 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

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

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

\[ Tv = Tv' \]

y coordinate is always the same!
What is stereo rectification?
What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers.
What is stereo rectification?

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

Two homographies (3x3 transform), one for each input image reprojection.

Stereo Rectification

1. Rotate the left camera so that the epipole is at infinity
2. Apply the same rotation to the right camera
3. Rotate the right camera by $R^{-1}$
4. Adjust the scale
Setting the epipole to infinity

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

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

\[
\mathbf{r}_1 = \mathbf{e}_1 = \frac{T}{\|T\|}
\]

Epipole coincides with translation vector

\[
\mathbf{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y \\ T_x \\ 0 \end{bmatrix}
\]

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 \( r_2 \perp r_3 \) orthogonal

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

then $R_{\text{rect}}e_1 = \begin{bmatrix} r_1^\top e_1 \\ r_2^\top e_1 \\ r_3^\top e_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Where is this point located on the image plane?
Stereo Rectification Algorithm

1. Estimate $E$ using the 8 point algorithm (SVD)

2. Estimate the epipole $e$ (SVD of $E$)

3. Build $R_{\text{rect}}$ from $e$

4. Decompose $E$ into $R$ and $T$

5. Set $R_1 = R_{\text{rect}}$ and $R_2 = RR_{\text{rect}}$

6. Rotate each left camera point (warp image)
   \[ [x' \ y' \ z'] = R_1 \ [x \ y \ z] \]

7. Rectified points as $p = f/z'[x' \ y' \ z']$

8. Repeat 6 and 7 for right camera points using $R_2$
Stereo Rectification Algorithm

1. Estimate $E$ using the 8 point algorithm
2. Estimate the epipole $e$ (solve $Ee=0$)
3. Build $R_{\text{rect}}$ from $e$
4. Decompose $E$ into $R$ and $T$
5. Set $R_1=R_{\text{rect}}$ and $R_2 = RR_{\text{rect}}$
6. Rotate each left camera point $x' \sim Hx$ where $H = KR_1$
   *You may need to alter the focal length (inside $K$) to keep points within the original image size*
7. Repeat 6 and 7 for right camera points using $R_2$
Unrectified

Rectified
Finding the best match

- 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
**Similarity Measure**

- Sum of Absolute Differences (SAD)
- Sum of Squared Differences (SSD)
- Zero-mean SAD
- Locally scaled SAD
- Normalized Cross Correlation (NCC)

**Formula**

\[
\sum_{(i,j) \in W} |I_1(i,j) - I_2(x + i, y + j)|
\]

\[
\sum_{(i,j) \in W} (I_1(i,j) - I_2(x + i, y + j))^2
\]

\[
\sum_{(i,j) \in W} |I_1(i,j) - \bar{I}_1(i,j) - I_2(x + i, y + j) + \bar{I}_2(x + i, y + j)|
\]

\[
\sum_{(i,j) \in W} |I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x + i, y + j)} I_2(x + i, y + j)|
\]

\[
\frac{\sum_{(i,j) \in W} I_1(i,j) . I_2(x + i, y + j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i,j) . \sum_{(i,j) \in W} I_2^2(x + i, y + j)}}
\]

**Images:**
- SAD
- SSD
- NCC
- Ground truth
Effect of window size

$W = 3$

$W = 20$
Effect of window size

Smaller window
+ More detail
- More noise

Larger window
+ 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
(break)
Improving Stereo Block Matching
What are some problems with the result?
How can we improve depth estimation?
Uniqueness

For any point in one image, there should be at most one matching point in the other image.
Ordering

Corresponding points should be in the same order in both views
Ordering

Corresponding points should be in the same order in both views
Smoothness

We expect disparity values to change slowly (for the most part)

Too many discontinuities

Better
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
$E(d) = E_d(d) + \lambda E_s(d)$

- **data term**
  - Want each pixel to find a good match in the other image
  - (match cost)

- **smoothness term**
  - Adjacent pixels should (usually) move about the same amount
  - (smoothness cost)
\[ E(d) = E_d(d) + \lambda E_s(d) \]

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

SSD distance between windows centered at \( I(x, y) \) and \( J(x + d(x,y), y) \)
\[ E(d) = E_d(d) + \lambda E_s(d) \]

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

SSD distance between windows centered at \( I(x, y) \) and \( J(x + d(x, y), y) \)

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

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

4-connected neighborhood

8-connected neighborhood
$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$

$V(d_p, d_q) = |d_p - d_q|$

L₁ distance

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

"Potts model"
Dynamic Programming

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

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

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

\[ D(x, y, d) = C(x, y, d) + \min_{d'} \left\{ D(x - 1, y, d') + \lambda |d - d'| \right\} \]
Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001
Stereo Vision

16-385 Computer Vision
Carnegie Mellon University (Kris Kitani)
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.
\[
\frac{X}{Z} = \frac{x}{f}
\]
\[
\frac{X}{Z} = \frac{x}{f}
\]

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

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

**Disparity**

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

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

**Disparity**

\[
d = x - x' = \frac{bf}{Z}
\]

Disparity is inversely proportional to depth.
Real-time stereo sensing

Nomad robot searches for meteorites in Antarctica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html
Subaru Eyesight system

Pre-collision braking
How do you compute depth from a stereo pair?
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?
It’s hard to make the image planes exactly parallel
How can you make the epipolar lines horizontal?

When this relationship holds

\[
R = I \quad t = (T, 0, 0)
\]
How can you make the 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

$$ x^T E x' = 0 $$
How can you make the 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

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

Write out the constraint

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

\[
\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0
\]
How can you make the 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

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

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

\[ T v = T v' \]

y coordinate is always the same!
What is stereo rectification?
What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers.
What is stereo rectification?

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

Two homographies (3x3 transform), one for each input image reprojection.

Stereo Rectification

1. Rotate the left camera so that the epipole is at infinity

2. Apply the same rotation to the right camera

3. Rotate the right camera by $R^{-1}$

4. Adjust the scale
Setting the epipole to infinity

(Building $R_{\text{rect}}$ from $e$)

Let $R_{\text{rect}} = \begin{bmatrix} r_1^\top \\ r_2^\top \\ r_3^\top \end{bmatrix}$

$r_1 = e_1 = \frac{T}{||T||}$

$epipole \ coincides \ with \ translation \ vector$

$r_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y \\ T_x \\ 0 \end{bmatrix}$

cross product of $e$ and the direction vector of the optical axis

$r_3 = r_1 \times r_2$

orthogonal vector
If \( r_1 = e_1 = \frac{T}{||T||} \) and \( r_2 \perp r_3 \) 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 \( r_2 \perp r_3 \) orthogonal \\
then \( R_{\text{rect}} e_1 = \begin{bmatrix} \bar{r}_1 e_1 \\ \bar{r}_2 e_1 \\ \bar{r}_3 e_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \)

Where is this point located on the image plane?
Stereo Rectification Algorithm

1. Estimate $E$ using the 8 point algorithm (SVD)

2. Estimate the epipole $e$ (SVD of $E$)

3. Build $R_{\text{rect}}$ from $e$

4. Decompose $E$ into $R$ and $T$

5. Set $R_1 = R_{\text{rect}}$ and $R_2 = RR_{\text{rect}}$

6. Rotate each left camera point (warp image) 
   \[ [x' \ y' \ z'] = R_1 [x \ y \ z] \]

7. Rectified points as $p = f/z' [x' \ y' \ z']$

8. Repeat 6 and 7 for right camera points using $R_2$
Stereo Rectification Algorithm

1. Estimate $E$ using the 8 point algorithm

2. Estimate the epipole $e$ (solve $Ee=0$)

3. Build $R_{\text{rect}}$ from $e$

4. Decompose $E$ into $R$ and $T$

5. Set $R_1 = R_{\text{rect}}$ and $R_2 = RR_{\text{rect}}$

6. Rotate each left camera point $x'~ Hx$ where $H = KR_1$
   *You may need to alter the focal length (inside $K$) to keep points within the original image size

7. Repeat 6 and 7 for right camera points using $R_2$
Finding the best match

- 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**

\[
\sum_{(i,j) \in W} |I_1(i,j) - I_2(x + i, y + j)|
\]

\[
\sum_{(i,j) \in W} (I_1(i,j) - I_2(x + i, y + j))^2
\]

\[
\sum_{(i,j) \in W} |I_1(i,j) - \bar{I}_1(i,j) - I_2(x + i, y + j) + \bar{I}_2(x + i, y + j)|
\]

\[
\sum_{(i,j) \in W} |I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x + i, y + j)} I_2(x + i, y + j)|
\]

\[
\frac{\sum_{(i,j) \in W} I_1(i,j) \cdot I_2(x + i, y + j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i,j) \cdot \sum_{(i,j) \in W} I_2^2(x + i, y + j)}}
\]
Effect of window size

W = 3

W = 20
Effect of window size

Smaller window
+ More detail
- More noise

Larger window
+ 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
(break)
Improving Stereo Block Matching
What are some problems with the result?
How can we improve depth estimation?
Uniqueness

For any point in one image, there should be at most one matching point in the other image.
Ordering

Corresponding points should be in the same order in both views
Ordering

Corresponding points should be in the same order in both views.
Smoothness

We expect disparity values to change slowly (for the most part)

Too many discontinuities  Better
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
$$E(d) = E_d(d) + \lambda E_s(d)$$

- **Data Term**: Want each pixel to find a good match in the other image (match cost)
- **Smoothness Term**: Adjacent pixels should (usually) move about the same amount (smoothness cost)
\[ E(d) = E_d(d) + \lambda E_s(d) \]

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

SSD distance between windows centered at \( I(x, y) \) and \( J(x + d(x, y), y) \)
\[ E(d) = E_d(d) + \lambda E_s(d) \]

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

SSD distance between windows centered at \( I(x, y) \) and \( J(x + d(x, y), y) \)

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

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

4-connected neighborhood

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

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

L₁ distance

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

“Potts model”
Dynamic Programming

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

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

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

\[ D(x, y, d) = C(x, y, d) + \min_{d'} \{ D(x - 1, y, d') + \lambda |d - d'| \} \]
Match only  Match & smoothness (graph cuts)  Ground Truth

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