Stereo II

Computer Vision I
CSE252A
Lecture 16
Announcements

• HW3 assigned
Epipolar Constraint: Calibrated Case

\[ \overrightarrow{O_p} \cdot [\overrightarrow{OO'} \times \overrightarrow{O'p'}] = 0 \]

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

with

\[
\begin{align*}
\mathbf{p} &= (u, v, 1)^T \\
\mathbf{p'} &= (u', v', 1)^T \\
\mathbf{M} &= (\mathbf{I} \: \mathbf{0}) \\
\mathbf{M'} &= (\mathbf{R}^T, -\mathbf{R}^T \mathbf{t})
\end{align*}
\]

Essential Matrix
(Longuet-Higgins, 1981)

\[ \mathbf{p}^T \mathbf{E} \mathbf{p'} = 0 \] with
\[ \mathbf{E} = [\mathbf{t} \times] \mathbf{R} \]
The Eight-Point Algorithm (Longuet-Higgins, 1981)
Much more on multi-view in CSE252B!!

\[
\begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} = 0 \quad \text{Minimize:} \quad \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} = 0
\]

Set \( F_{33} \) to 1

\[
\begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1u'_1 & v_1v'_1 & v_1u'_1 & v'_1 & v_1 & u'_1 & v'_1 \\ u_2u'_2 & u_2v'_2 & u_2u'_2 & v_2v'_2 & v_2u'_2 & v'_2 & v_2 & u'_2 & v'_2 \\ u_3u'_3 & u_3v'_3 & u_3u'_3 & v_3v'_3 & v_3u'_3 & v'_3 & v_3 & u'_3 & v'_3 \\ u_4u'_4 & u_4v'_4 & u_4u'_4 & v_4v'_4 & v_4u'_4 & v'_4 & v_4 & u'_4 & v'_4 \\ u_5u'_5 & u_5v'_5 & u_5u'_5 & v_5v'_5 & v_5u'_5 & v'_5 & v_5 & u'_5 & v'_5 \\ u_6u'_6 & u_6v'_6 & u_6u'_6 & v_6v'_6 & v_6u'_6 & v'_6 & v_6 & u'_6 & v'_6 \\ u_7u'_7 & u_7v'_7 & u_7u'_7 & v_7v'_7 & v_7u'_7 & v'_7 & v_7 & u'_7 & v'_7 \\ u_8u'_8 & u_8v'_8 & u_8u'_8 & v_8v'_8 & v_8u'_8 & v'_8 & v_8 & u'_8 & v'_8 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{pmatrix} = - \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}
\]

Minimize:

\[
\sum_{i=1}^{n} (p_i^T \mathcal{F} p'_i)^2
\]

under the constraint

\[
|\mathcal{F}|^2 = 1.
\]
Epipolar geometry example
Rectification

Given a pair of images, transform both images so that epipolar lines are scan lines.

Input Images
Rectification

Given a pair of images, transform both images so that epipolar lines are scan lines.

Rectified Images

See Section 7.3.7 for specific method
Image pair rectification

simplify stereo matching by warping the images

Apply projective transformation so that epipolar lines correspond to horizontal scanlines

map epipole \( e \) to \((1,0,0)\)

try to minimize image distortion

Note that rectified images usually not rectangular
Epipolar correspondence
Correspondence Search Algorithm

For  for j=1:ncols
    best(i,j) = -1
    for k = mindisparity:maxdisparity
        c = Match_Metric(I\(_1\)(i,j),I\(_2\)(i,j+k),winsize)
        if (c > best(i,j))
            best(i,j) = c
            disparities(i,j) = k
        end
    end
end

O(nrows * ncols * disparities * winx * winy)
Finding Correspondences

\[ W(p_1) \quad W(p_{1'}) \]
# Match Metric Summary

<table>
<thead>
<tr>
<th>MATCH METRIC</th>
<th>DEFINITION</th>
</tr>
</thead>
</table>
| Normalized Cross-Correlation (NCC)  | \[
\sum_{u,v} \left( I_1(u,v) - \bar{I}_1 \right) \cdot \left( I_2(u+d,v) - \bar{I}_2 \right) \over \sqrt{\sum_{u,v} (I_1(u,v) - \bar{I}_1)^2} \cdot \sqrt{\sum_{u,v} (I_2(u+d,v) - \bar{I}_2)^2} \right] 
\] |
| Sum of Squared Differences (SSD)    | \[
\sum_{u,v} (I_1(u,v) - I_2(u+d,v))^2 \]                                      |
| Normalized SSD                      | \[
\sum_{u,v} \left( \left( I_1(u,v) - \bar{I}_1 \right) \over \sqrt{\sum_{u,v} (I_1(u,v) - \bar{I}_1)^2} \right)^2 - \left( I_2(u+d,v) - \bar{I}_2 \right) \over \sqrt{\sum_{u,v} (I_2(u+d,v) - \bar{I}_2)^2} \right)^2 \] |
| Sum of Absolute Differences (SAD)   | \[
\sum_{u,v} |I_1(u,v) - I_2(u+d,v)| \]                                                |
| Zero Mean SAD                       | \[
\sum_{u,v} |(I_1(u,v) - \bar{I}_1) - (I_2(u+d,v) - \bar{I}_2)| \]                       |
| Rank                                | \[
I'_k(u,v) = \sum_{m,n} I_k(m,n) < I_k(u,v) \\
\sum_{u,v} (I'_1(u,v) - I'_2(u+d,v)) \]                                    |
| Census                              | \[
I'_k(u,v) = \text{BITSTRING}_{m,n} (I_k(m,n) < I_k(u,v)) \\
\sum_{u,v} \text{HAMMING}(I'_1(u,v), I'_2(u+d,v)) \]                          |
Stereo results

– Data from University of Tsukuba

Scene

Ground truth

(Seitz)
Results with window correlation

Window-based matching
(best window size)

Ground truth
(Seitz)
Results with better method

State of the art method

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts,
International Conference on Computer Vision, September 1999.

(Seitz)

Ground truth
Some Issues

- Ambiguity
- Window size
- Window shape
- Lighting
- Half occluded regions
Ambiguity

Many to one matches possible
Window size

- Effect of window size

Better results with *adaptive window*


(Seitz)
Window Shape and Forshortening
Window Shape: Fronto-parallel Configuration
Lighting Conditions (Photometric Variations)

\[ W(P_1) \quad W(P_r) \]
Problem of Occlusion

- Occluded in Right Image
- Occluded in Left Image

Left Image

Left Center of Projection

Right Image

Right Center of Projection
## Stereo Constraints

<table>
<thead>
<tr>
<th>CONSTRAINT</th>
<th>BRIEF DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-D Epipolar Search</td>
<td>Arbitrary images of the same scene may be rectified based on epipolar geometry such that stereo matches lie along one-dimensional scanlines. This reduces the computational complexity and also reduces the likelihood of false matches.</td>
</tr>
<tr>
<td>Monotonic Ordering</td>
<td>Points along an epipolar scanline appear in the same order in both stereo images, assuming that all objects in the scene are approximately the same distance from the cameras.</td>
</tr>
<tr>
<td>Image Brightness Constancy</td>
<td>Assuming Lambertian surfaces, the brightness of corresponding points in stereo images are the same.</td>
</tr>
<tr>
<td>Match Uniqueness</td>
<td>For every point in one stereo image, there is at most one corresponding point in the other image.</td>
</tr>
<tr>
<td>Disparity Continuity</td>
<td>Disparities vary smoothly (i.e. disparity gradient is small) over most of the image. This assumption is violated at object boundaries.</td>
</tr>
<tr>
<td>Disparity Limit</td>
<td>The search space may be reduced significantly by limiting the disparity range, reducing both computational complexity and the likelihood of false matches.</td>
</tr>
<tr>
<td>Fronto-Parallel Surfaces</td>
<td>The implicit assumption made by area-based matching is that objects have fronto-parallel surfaces (i.e. depth is constant within the region of local support). This assumption is violated by sloping and creased surfaces.</td>
</tr>
<tr>
<td>Feature Similarity</td>
<td>Corresponding features must be similar (e.g. edges must have roughly the same length and orientation).</td>
</tr>
<tr>
<td>Structural Grouping</td>
<td>Corresponding feature groupings and their connectivity must be consistent.</td>
</tr>
</tbody>
</table>

(From G. Hager)
Stereo matching

- Similarity measure (SSD or NCC)
- Optimal path (dynamic programming)

Constraints
- epipolar
- ordering
- uniqueness
- disparity limit
- disparity gradient limit

Trade-off
- Matching cost (data)
- Discontinuities (prior)

(From Pollefeys)

(Cox et al. CVGIP’96; Koch’96; Falkenhagen´97; Van Meerbergen, Vergauwen, Pollefeys, VanGool IJCV’02)
Stereo Matching with Dynamic Programming

(Slides adapted from Jim Rehg at GA Tech)

Dynamic programming yields the optimal path through grid. This is the best set of matches that satisfy the ordering constraint.

Every pixel on each scanline will be labeled as matching, or occluded.
Dynamic Programming

Used with Hidden Markov Models, Viterbi Algorithm

Consider a graph with $MT$ nodes, where the graph represents $T$ different times ($t=1..T$) and $M$ states ($i=1,…M$) at each time. There are directed arcs, only between successive times and for each time $t$, $\Pi_{ij}$ denotes the cost of transitioning from state $i$ to state $j$.

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

- Efficient algorithm for solving sequential decision (optimal path) problems. Cost associated with each arc.

What is cost of path?
How many paths through this trellis?
Using Dynamic Programming, can find optimal path in $O(M \cdot T)$ time (here $M=3$)

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming for Stereo

- Efficient algorithm for solving sequential decision (optimal path) problems.

\[
\begin{align*}
  i = 1 & \quad \begin{array}{c} 1 \end{array} \\
  i = 2 & \quad \begin{array}{c} 2 \end{array} \\
  i = 3 & \quad \begin{array}{c} 3 \end{array} \\
  t = 1 & \quad \begin{array}{c} 1 \end{array} \\
  t = 2 & \quad \begin{array}{c} 2 \end{array} \\
  t = 3 & \quad \begin{array}{c} 3 \end{array} \\
  t = T & \quad \begin{array}{c} 1 \end{array}
\end{align*}
\]

For Stereo,
\( t \) can denote pixel coordinates across an epipolar line in one image
\( i \) can denote the disparity to the other epipolar line

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

Used with Hidden Markov Models, Viterbi Algorithm

Suppose cost can be decomposed into stages:

\[ \Pi_{ij} = \text{Cost of going from state } i \text{ to state } j \]
Dynamic Programming
Minimum Cost Path

What is minimum cost of reaching node $j$ at time $t$? $C_t(j)$

$$C_t(j) = \min_i (\Pi_{ij} + C_{t-1}(i))$$

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

Used with Hidden Markov Models, Viterbi Algorithm

What is minimum cost of reaching node j at time t?

\[ C_t(j) = \min_i \left( \Pi_{ij} + C_{t-1}(i) \right) \]

Minimum cost of path from \( t=0 \) to reach state j at time t.

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

Used with Hidden Markov Models, Viterbi Algorithm

\[ i = 1 \]
\[ \Pi_{12} = 2 \]
\[ i = 2 \]
\[ \Pi_{22} = 1.3 \]
\[ \Pi_{32} = 6.1 \]
\[ i = 3 \]
\[ \Pi_{33} = 3 \]

\[
C_t(j) = \min_i \left( \Pi_{ij} + C_{t-1}(i) \right)
\]

\[
b_t(j) = \arg \min_i \left( \Pi_{ij} + C_{t-1}(i) \right)
\]

\[ b_t(j) \text{ gives previous state along minimum cost path} \]

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

\[ C_t(j) = \min_i \left( \Pi_{ij} + C_{t-1}(i) \right) \]

\[ b_t(j) = \arg \min_i \left( \Pi_{ij} + C_{t-1}(i) \right) \]

\[ b_t(j) \] gives previous state along minimum cost path

e.g. \[ b_t(3) = 3 \]
\[ b_t(2) = 2 \]
\[ b_t(1) = 2 \]

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

\[ C_t(j) = \min_i \left( \prod_{ij} + C_{t-1}(i) \right) \]

\[ b_t(j) = \arg \min_i \left( \prod_{ij} + C_{t-1}(i) \right) \]

\[ b_t(j) \text{ gives previous state along minimum cost path} \]

So, \( b_t(3) = 3 \)
\[ b_t(2) = 2 \]
\[ b_t(1) = 2 \]

(Slides adapted from Jim Rehg at GA Tech)
Dynamic Programming

1. Iteratively, compute minimum cost to reach all nodes
   \[ C_t(j) = \min_i \Pi_{ij} + C_{t-1}(i) \]

2. Recursively, starting with the node at time t-max, select lowest cost terminal node, and backtrack along path
   \[ b_t(j) = \arg \min_i \Pi_{ij} + C_{t-1}(i) \]

Min cost path

1. 
   - \( i = 1 \)
   - \( i = 2 \)
   - \( i = 3 \)
Dynamic Programming

Note that arc cost $\Pi_{ij}(t)$ may include both a transition cost, plus a node cost. In stereo, the node cost would be the result the match cost.

(Slides adapted from Jim Rehg at GA Tech)
Compute Optimal Path Costs

```
\[
\text{Occlusion} = \left[ \ln \left( \frac{P_D}{1-P_D} \frac{\phi}{(2\pi)^{d/2}S_{\phi}^{1/2}} \right) \right]
\]

\text{for } (i=1; i \leq N; i++) \{ \text{C}(i,0) = i*\text{Occlusion} \} \text{ for } (i=1; i \leq M; i++) \{ \text{C}(0,i) = i*\text{Occlusion}\}

\text{for}(i=1; i \leq N; i++)\{
\text{for}(j=1; j \leq M; j++)\{
\text{min1} = \text{C}(i-1, j-1) + \text{C}(z_{1,i}, z_{2,j}); \text{min2} = \text{C}(i-1, j) + \text{Occlusion}; \text{min3} = \text{C}(i, j-1) + \text{Occlusion}; \text{C}(i,j) = \text{cmin} = \min(\text{min1, min2, min3}); \\
\text{if}(\text{min1==cmin}) \text{ M}(i,j) = 1; \text{if}(\text{min2==cmin}) \text{ M}(i,j) = 2; \text{if}(\text{min3==cmin}) \text{ M}(i,j) = 3; 
\}
\}
```

\(\text{C}(i,j)\): Cost of optimal path to match of pixels \(i\) and \(j\)

\(\text{M}(i,j)\): ‘Pointer’ to previous node along optimal path

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
Back tracking to get optimal path

```c
p=N;
q=M;
while(p!=0 && q!=0){
    switch(M(p,q)){
        case 1:
            p matches q
            p--;q--;
            break;
        case 2:
            p is unmatched
            p--;
            break;
        case 3:
            q is unmatched
            q--;
            break;
    }
}
```
Stereo Matching with Dynamic Programming

C(i,j) is minimum of
1. C(i-1,j-1) + match-cost of pixel L(i) & R(i)
2. C(i-1,j) + occlusion-penalty
3. C(i,j-1) + occlusion-penalty

Occluded Pixels

Left scanline

Dis-occluded Pixels

Right scanline

Terminal
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors.
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors.
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors.
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors. Backtrack from the terminal to get the optimal path.
Stereo Matching with Dynamic Programming

Once $C(i,j)$ is completely calculated:

Backtrack from the terminal to get the optimal path.