CS 532: 3D Computer Vision
4th Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philipppos.Mordohai@stevens.edu
Office: Lieb 215
Lecture Outline

• Binocular Stereo
  – Continued
• Confidence for stereo
• Stereo beyond the Winner-Take-All algorithm

Partially based on slides by M. Bleyer, R. Szeliski, S. Seitz and R. Zabih
Naïve Stereo Algorithm

• For each pixel \( p \) of the left image:
  – Compare color of \( p \) against the color of each pixel on the same horizontal scanline in the right image.
  – Select the pixel of most similar color as matching point
Window-Based Matching

• Instead of matching single pixels, center a small window on a pixel and match the whole window in the right image.
Window-Based Matching

- the disparity \( d_p \) of a pixel \( p \) in the left image is computed as

\[
d_p = \arg \min_{0 \leq d \leq d_{\text{max}}} \sum_{q \in W_p} c(q, q - d)
\]

- where
  - \( \arg \min \) returns the value at which the function takes a minimum
  - \( d_{\text{max}} \) is a parameter defining the maximum disparity (search range)
  - \( W_p \) is the set of all pixels inside the window centered on \( p \)
  - \( c(p, q) \) is a function that computes the color difference between a pixel \( p \) of the left and a pixel \( q \) of the right image
Results

• The window size is a crucial parameter

Window size = 3x3 pixels  Window size = 21x21 pixels
Untextured Regions

(a) Left image

(b) Right image

Multiple points fit equally well. What is the correct disparity?
Aperture Problem

- There needs to be a certain amount of texture with vertical orientation.

(a) Left image

Texture with only horizontal orientation

(b) Right image

Multiple points fit equally well. What is the correct disparity?
Repetitive Patterns

(a) Left image

(b) Right image

Multiple points fit equally well. What is the correct disparity?
Effects of these Problems

Window size = 3x3 pixels

Low Texture
Aperture Problem
Repetitive Pattern
Foreground Fattening

• By using a window as matching primitive, we have made an implicit smoothness assumption:
  – All pixels within the window are assumed to have the same disparity
• This leads to a systematic error in regions close to disparity discontinuities
Foreground Fattening

- Background regions close to disparity discontinuities tend to be erroneously assigned to the foreground disparity.
Foreground Fattening

Ground Truth Disparities

Window size = 21x21 pixels
Large vs. Small Windows

• Large windows are better for:
  – Untextured Regions
  – Aperture Problem
  – Repetitive Patterns

• Small windows reduce:
  – Foreground Fattening Effect

• Problem:
  – There is no ‘optimal’ window size that can handle all these problems at once

Why?
Matching Costs

• See slides 41-46 from previous set of notes
Implement SAD
Occlusion

• There are pixels that are only visible in one of the two views (red pixels in the images)

• For occluded, pixels there exists no correspondence => We cannot estimate disparity
Effects of Occlusion

Ground Truth with Occlusions in Black

Core Algorithm of [Hosni,ICIP09]
Left-Right Consistency

• Compute 2 disparity maps
  – Using the left image as reference frame
  – Using the right image as reference frame

• Left-right consistency check:
  – For each pixel $p_l$ of the left view:
  – Lookup $p_l$’s matching point $m_r$ in the right view using the left disparity map
  – For the pixel $m_r$, lookup its matching point $q_l$ in the left view using the right disparity map
  – If $p_l = q_l$ the disparity is assumed to be correct
  – Otherwise, the disparity is invalidated

• Check typically fails for
  – Occluded pixels
  – Mismatched pixels
Left-Right Consistency

Disparity Map (Left Reference)  Disparity Map (Right Reference)
Left-Right Consistency

$p = \langle x, y \rangle$

disparity = 50 pixels

$p' = \langle x-50, y \rangle$

disparity = 50 pixels
Left-Right Consistency

\[ p = (x, y) \]
\[ \text{disparity} = \text{50 pixels} \]

\[ p' = (x-50, y) \]
\[ \text{disparity} = \text{50 pixels} \]

Disparity Map (Left Reference)  Disparity Map (Right Reference)
Left-Right Consistency

\[ p = \langle x, y \rangle \]
\[ \text{disparity} = 60 \text{ pixels} \]

Disparity Map (Left Reference)  Disparity Map (Right Reference)
Left-Right Consistency

$p = (x, y)$

disparity = 60 pixels

$p' = (x-60, y)$

disparity = 25 pixels
Left-Right Consistency

$p = \langle x, y \rangle$

Disparity $= 60$ pixels

Test Failed

$p' = \langle x-60, y \rangle$

Disparity $= 25$ pixels
Confidence Measures for Stereo Matching
Cost Functions

\[ SAD(x, y, d) = \sum_{i \in W} |I_L(x_i, y_i) - I_R(x_i - d, y_i)| \]

\[ NCC(x, y, d) = \frac{\sum_{i \in W} (I_L(x_i, y_i) - \mu_L)(I_R(x_i - d, y_i) - \mu_R)}{\sigma_L\sigma_R} \]

- Functions of disparity \( d \)
  - \( d = x_L - x_R \)
Ideal Cost Curve
Non-Ideal Cost Curves
Correspondence Uncertainty Measures

1. Matching Cost
2. Local Properties of the Cost Curve
3. Local Minima of the Cost Curve
4. The Entire Cost Curve
6. Consistency Between the Left and Right Disparity Maps
7. More...
c₁ global minimum of the cost curve

\( c_2 \) second smallest value of the cost curve

\( c_{2m} \) second smallest value of the cost curve that is also a local minimum
Simply using matching cost:
Low cost values correspond to high confidence
high cost values correspond to low confidence
Local Properties of the Cost Curve

\[ C_{CUR} = -2c(d_1) + c(d_1 - 1) + c(d_1 + 1) \]

Low curvature indicates flat region around minimum cost
Local Minima of the Cost Curve

\[ C_{PKR} = \frac{c_{2m}}{c_1} \quad \quad C_{PKRN} = \frac{c_2}{c_1} \]

Match is ambiguous if multiple strong candidates exist
The Entire Cost Curve

Tests for both flat regions and multiple strong candidates by converting cost curve to probability function and measuring the probability of the best match.
Left-Right Consistency

\[ C_{LRC}(x, y) = -|d_1 - D_R(x - d_1, y)| \]

LRC is not binary as before, but equal to difference of corresponding disparities
Distinctiveness-based Confidence Measures

- Distinctiveness: Perceptual distance to the most similar other point in the search window in the reference image
Stereo
Beyond Winner-Take-All
Stereo with Non-Linear Diffusion

- Problem with traditional approach:
  - gets confused near discontinuities
- Non-Linear Diffusion:
  - use iterative (non-linear) aggregation to obtain better estimate
Diffusion

- Average energy with neighbors + starting value

\[ E(x, y, d) \leftarrow (1-4\lambda)E(x, y, d) + \lambda \sum_{(k,l) \in \mathcal{N}_4} E(x+k, y+l, d) + \beta(E_0(x, y, d) - E(x, y, d)) \]

- window diffusion

Requires appropriate stopping criteria to prevent excessive smoothing
Feature-based Stereo

- Match “corner” (interest) points
- Interpolate complete solution
Dynamic Programming

• 1-D cost function

\[ E(d) = \sum_{x,y} \rho_P(d_{x+1,y} - d_{x,y}) + \sum_{x,y} E_0(x, y; d) \]

\[ \tilde{E}(x, y, d) = E_0(x, y; d) + \min_{d'} \left( \tilde{E}(x - 1, y, d') + \rho_P(d_{x,y} - d'_{x-1,y}) \right) \]
Dynamic Programming

• Disparity space image and min. cost path
Dynamic Programming

• Sample result (note horizontal streaks)
Dynamic Programming

• Can we apply this trick in 2D as well?

No: \(d_{x,y-1}\) and \(d_{x-1,y}\) may depend on different values of \(d_{x-1,y-1}\)
Graph Cuts

- Solution technique for general 2D problem

\[
E_{\text{total}}(d) = E_{\text{data}}(d) + \lambda E_{\text{smoothness}}(d)
\]

\[
E_{\text{data}}(d) = \sum_{x,y} f_{x,y}(d_{x,y})
\]

\[
E_{\text{smoothness}}(d) = \sum_{x,y} \rho(d_{x,y} - d_{x-1,y}) + \sum_{x,y} \rho(d_{x,y} - d_{x,y-1})
\]

(a) original image  (b) observed image  (c) local min w.r.t. standard moves  (d) local min w.r.t. \(\alpha\)-expansion moves
In each $a$-expansion a given label “$a$” grabs space from other labels. For each move choose the expansion that gives the largest decrease in the energy: *binary optimization problem*.
Feature Extraction
Features

• So far, we have assumed that ideal points can be detected in the images and matches across images
  – E.g. for homography and fundamental matrix estimation
What makes a Good Feature?

Uniqueness – at least Distinctiveness
Invariance
Availability - Efficiency
Local Measures of Uniqueness

Suppose we only consider a small window of pixels

– What defines whether a feature is a good or bad candidate?
Feature Detection

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.
Consider shifting the window $W$ by $(u,v)$

- how do the pixels in $W$ change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” of $E(u,v)$:

$$E(u,v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$
Small Motion Assumption

Taylor Series expansion of $I$:

$$I(x + u, y + v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms}$$

If the motion $(u, v)$ is small, then first order approx is good

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

$$\approx I(x, y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…
Feature Detection: the Math

\[ E(u, v) = \sum_{(x, y) \in W} [I(x + u, y + v) - I(x, y)]^2 \]

\[ \approx \sum_{(x, y) \in W} [I(x, y) + [I_x I_y] \begin{bmatrix} u \\ v \end{bmatrix}]^2 - I(x, y)] \]

\[ \approx \sum_{(x, y) \in W} [I_x I_y] \begin{bmatrix} u \\ v \end{bmatrix}]^2 \]
Feature Detection: the Math

This can be rewritten:

\[ E(u, v) = \sum_{(x,y) \in W} \begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

For the example above:

- You can move the center of the window to anywhere on the blue unit circle.
- Which directions will result in the largest and smallest \( E \) values?
- We can find these directions by looking at the eigenvectors of \( H \).
Quick Eigenvalue/Eigenvector Review

The eigenvectors of a matrix $A$ are the vectors $x$ that satisfy:

$$Ax = \lambda x$$

The scalar $\lambda$ is the eigenvalue corresponding to $x$

- The eigenvalues are found by solving:

$$\det(A - \lambda I) = 0$$

- In our case, $A = H$ is a 2x2 matrix, so we have

$$\det\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

Once you know $\lambda$, you find $x$ by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$
Feature Detection: the Math

This can be rewritten:

\[ E(u, v) = \sum_{(x, y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- \( x_+ \) = direction of **largest** increase in \( E \).
- \( \lambda_+ \) = amount of increase in direction \( x_+ \)
- \( x_- \) = direction of **smallest** increase in \( E \).
- \( \lambda_- \) = amount of increase in direction \( x_- \)
Feature Detection: the Math

How are $\lambda_+$, $x_+$, $\lambda_-$, and $x_-$ relevant for feature detection?
  - What’s our feature scoring function?

Want $E(u,v)$ to be **large** for small shifts in **all** directions
  - the *minimum* of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
  - this minimum is given by the smaller eigenvalue ($\lambda_-$) of $H$
Feature Detection Summary

- Compute the gradient at each point in the image
- Create the $H$ matrix from the entries in the gradient
- Compute the eigenvalues
- Find points with large response ($\lambda_+ >$ threshold)
- Choose those points where $\lambda_+$ is a local maximum as features
The Harris Operator

$\lambda_\circ$ is a variant of the “Harris operator” for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)}$$

- The trace is the sum of the diagonals, i.e., $\text{trace}(H) = h_{11} + h_{22}$
- Very similar to $\lambda_\circ$ but less expensive (no square root)
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular
The Harris Operator

Harris operator

\( \lambda_\)
Harris Detector Example
f value (red high, blue low)
Threshold \((f > \text{value})\)
Harris Features (in red)
Invariance

Suppose you rotate the image by some angle
– Will you still pick up the same features?

What if you change the brightness?

Scale?
Feature Matching

Given a feature in $I_1$, how to find the best match in $I_2$?

1. Define distance function that compares two descriptors
2. Test all the features in $I_2$, find the one with min distance
Feature Distance

How to define the difference between two features $f_1, f_2$?

- Simple approach is $SSD(f_1, f_2)$
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches
Feature Distance

– Better approach:
  ratio distance = \( \frac{\text{SSD}(f_1, f_2)}{\text{SSD}(f_1, f_2')} \)
  
  \( f_2 \) is best SSD match to \( f_1 \) in \( I_2 \)
  
  \( f_2' \) is 2\(^{\text{nd}}\) best SSD match to \( f_1 \) in \( I_2 \)
  
  gives small values for ambiguous matches
Evaluating the Results

How can we measure the performance of a feature matcher?
The distance threshold affects performance

- True positives = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?