Capturing, Modeling, Rendering 3D Structures

Computer Vision (for 3D Reconstruction) In a Nutshell

Computer Vision Approach
- Calculate pixel correspondences and extract geometry
  - Not robust
  - Difficult to acquire illumination effects, e.g., specular highlights

Computer and Human Vision
- **Human**
  - Lens forms image on retina, sensors (rods and cones) respond to light
- **Computer**
  - Lens system forms image, sensors (CCD) respond to light

What is Computer Vision?
- Input: images or video
- Output: description of the 3D world
  - Low-level
  - Mid-level
  - High-level

Low-Level or "Early" Vision
- Considers local properties of an image
  - "There's an edge!"

Mid-Level Vision
- Grouping and segmentation
  - "There's an object and a background!"

* Some slides courtesy of Szymon Rusinkiewicz, Princeton University, Marc Pollefeys, UNC-Chapel Hill
**High-Level Vision**

- Recognition

  "It's a chair!"

**Big Question #1: Who Cares?**

- Applications of computer vision
  - In AI: vision serves as the "input stage"
  - In medicine: understanding human vision
  - In engineering: model extraction

**Vision and Other Fields**

- Cognitive Psychology
- Artificial Intelligence
- Signal Processing
- Computer Vision
- Computer Graphics
- Pattern Analysis
- Metrology

**Big Question #2: Does It Work?**

- Situation much the same as AI:
  - Some fundamental algorithms
  - Large collection of hacks / heuristics

- Vision is hard!
  - Especially at high level, physiology unknown
  - Requires integrating many different methods
  - Requires abstract reasoning and understanding

**Computer Vision**

- Early Vision
  - Image Resolution and Filtering
  - Edges, Corners, and Textures
- Motion
  - Optical Flow
- Reconstruction
  - Feature Tracking
  - Epipolar Geometry
  - Correspondence
  - Depth Estimation from Images Pairs
- We are dealing with images (or 3D views) of limited resolution.

**Thresholding**
- Make all pixels > 128 equal to white
- Make all pixels < 128 equal to black

**Gaussian Filters**
- Original: Mandrill
- Smoothed with Gaussian kernel
### Gaussian Filters
- One-dimensional Gaussian
  \[ G_1(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \]
- Two-dimensional Gaussian
  \[ G_2(x, y) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

### Gaussian Convolution
- Convolution
  - If \( f \) is \( n \times n \), \( f \) is \( m \times m \), takes time \( O(m^2n^2) \)
  - OK for small filter kernels, bad for large ones
- Gaussian convolutions are used because:
  - Smooth
  - Decay to zero rapidly
  - Simple analytic formula

### Edges, Corners, Textures
- What are the basic image features?

...ambiguous?

- What are the basic image features?

edge

- What are the basic image features?

corner

- What are the basic image features?
Edges, Corners, Textures

- What are the basic image features?

Example Feature Detection: Edges

- How do we find edges?
- Is it hard?
- Are edges well-defined?

What is an Edge?

- Edge easy to find

What is an Edge?

- Where is edge? Single pixel wide or multiple pixels?

What is an Edge?

- Noise: have to distinguish noise from actual edge

What is an Edge?

- Is this one edge or two?
What is an Edge?

Is a texture discontinuity an edge?

Canny Edge Detector

Original: Lena

Edges

Computer Vision

- Early Vision
  - Image Resolution and Filtering
  - Edges, Corners, and Textures
- Motion
  - Optical Flow
- Reconstruction
  - Feature Tracking
  - Epipolar Geometry
  - Correspondence
  - Depth Estimation from Images Pairs

Motion in Computer Vision

- Given an initial image, how does the image change as we move the viewpoint?

Optical Flow of the Image

- Can pixels flow in any direction?
Optical Flow of the Image

- Can pixels flow at any speed?

Feature Tracking

- Follow feature A from image I to image J
- Requires "matching" features…

Feature Matching = Object Matching

- Simplest: SSD with "pixel windows"

Computer Vision

- Early Vision
  - Image Resolution and Filtering
  - Edges, Corners, and Textures
- Motion
  - Optical Flow
- Reconstruction
  - Feature Tracking
  - Epipolar Geometry
  - Correspondence
  - Depth Estimation from Images Pairs

Feature Matching = Object Matching
Comparing Windows:

\[ SSD = \sum_{(i,j) \in R} (f(i,j) - g(i,j))^2 \]

Most popular

Subpixel SSD

- When motion is a few pixels or less, motion of an integer no. of pixels can be insufficient.

Bilinear Interpolation

To compare pixels that are not at integer grid points, we resample the image.
Assume image is locally bilinear.
\[ I(x, y) = ax + by + cxy + d = 0. \]
Given the value of the image at four points: \( I(x,y), I(x+1,y), I(x,y+1), I(x+1,y+1) \) we can solve for \( a, b, c, d \) linearly. Then, for any \( u \) between \( x \) and \( x+1 \), for any \( v \) between \( y \) and \( y+1 \), we use this equation to find \( I(u,v) \).

Matching: How to Match Efficiently

- Baseline approach: try everything.
\[ \arg \min_{u,v} \sum (W(x,y) - I(x+u,y+v))^2 \]
- Could range over whole image.
- Or only over a small displacement.

Matching: Multiscale
The Gaussian Pyramid

When motion is small: Optical Flow

- Small motion: \( (u \text{ and } v \text{ are less than 1 pixel}) \)
- Brute force not possible

\[
\frac{\partial I}{\partial x} \text{ displacement } = (u, v) \\
I(x,y) \\
I(x+u,y+v)
\]

- suppose we take the Taylor series expansion of \( I \):

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

Optical flow equation

- Combining these two equations

\[
0 = I(x+u,y+v) - H(x,y) \quad \text{shorthand: } I_x = \frac{\partial I}{\partial x}
\]

\[
 \approx I(x,y) + I_x u + I_y v - H(x,y)
\]

\[
 \approx (I(x,y) - H(x,y)) + I_x u + I_y v
\]

\[
 \approx I_x + I_y \nabla f
\]

- In the limit as \( u \) and \( v \) go to zero, this becomes exact

\[
0 = I_x + \nabla f \cdot \begin{pmatrix} u \\ v \end{pmatrix}
\]

Optical flow equation

- Q: how many unknowns and equations per pixel?

\[
0 = I_x + \nabla f \cdot \begin{pmatrix} u \\ v \end{pmatrix}
\]

- Intuitively, what does this constraint mean?

  - The component of the flow in the gradient direction is determined
  - The component of the flow parallel to an edge is unknown

First Order Approximation

When we assume that:

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

We assume an image locally is:

Aperture problem: where are we?
Aperture problem: where are we?

Solving the aperture problem

- How to get more equations for a pixel?
  - Basic idea: impose additional constraints
    - most common is to assume that the flow field is smooth locally
    - one method: pretend the pixel's neighbors have the same (u,v)
      - If we use a 5x5 window, that gives us 25 equations per pixel

\[
0 = I(x_0) + \nabla I(x_0) \cdot (u, v)
\]

\[
\begin{pmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
I_x(p_{25}) & I_y(p_{25})
\end{pmatrix}
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix}
= \begin{pmatrix}
I_x(p_1) \\
I_x(p_2) \\
I_x(p_{25})
\end{pmatrix}
\]

Lucas-Kanade flow

- We have more equations than unknowns: solve least squares problem.
  This is given by:

\[
\min_{u, v} \| A \mathbf{d} - b \|^2
\]

\[
(A^T A) \hat{d} = A^T b
\]

- Summations over all pixels in the KxK window
Conditions for solvability

– Optimal \((u, v)\) satisfies Lucas-Kanade equation

\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -\begin{bmatrix}
\sum I_x I_y \\
\sum I_y I_y
\end{bmatrix}
\]

When is This Solvable?
• \(A^T A\) should be invertible
• \(A^T A\) should not be too small due to noise
  – eigenvalues \(\lambda_1\) and \(\lambda_2\) of \(A^T A\) should not be too small
• \(A^T A\) should be well-conditioned
  – \(\lambda_1/\lambda_2\) should not be too large \((\lambda_1 = \text{lager eigenvalue})\)

What does this look like?

... Formula for Finding Corners

We look at matrix:

\[
C = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix}
\]

Sum over a small region, the hypothetical corner

Matrix is symmetric

Why this?

First, consider case where:

\[
C = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix} = \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix}
\]

This means all gradients in neighborhood are:

\((k,0)\) or \((0, c)\) or \((0, 0)\) (or off-diagonals cancel).

What is region like if:
1. \(\lambda_1 = 0\)?
2. \(\lambda_2 = 0\)?
3. \(\lambda_1 = 0\) and \(\lambda_2 = 0\)?
4. \(\lambda_1 > 0\) and \(\lambda_2 > 0\)?

General Case:

From Singular Value Decomposition it follows that since \(C\) is symmetric:

\[
C = R^{-1} \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} R
\]

where \(R\) is a rotation matrix.
So every case is like one on last slide.

So, corners are the things we can track

– Corners are when \(\lambda_1, \lambda_2\) are big; this is also when Lucas-Kanade works.
– Corners are regions with two different directions of gradient (at least).
– Aperture problem disappears at corners.
– Not perfect, but works...

Edge

\[
\sum \nabla I(x, y)^4
\]

– large gradients, all the same
– large \(\lambda_1\), small \(\lambda_2\)

(Seitz)
Low texture region
- gradients have small magnitude
  - small $\lambda_1$, small $\lambda_2$

High textured region
- gradients are different, large magnitudes
  - large $\lambda_1$, large $\lambda_2$

Errors in Lucas-Kanade
- What are the potential causes of errors in this procedure?
  - Suppose $A^TA$ is easily invertible
  - Suppose there is not much noise in the image

- When our assumptions are violated
  - Brightness constancy is not satisfied
  - The motion is not small
  - A point does not move like its neighbors
    - window size is too large
    - what is the ideal window size?

Computer Vision
- Early Vision
  - Image Resolution and Filtering
  - Edges, Corners, and Textures
- Motion
  - Optical Flow
- Reconstruction
  - Feature Tracking
  - Epipolar Geometry
  - Correspondence
  - Depth Estimation from Images Pairs

Simple 3D Reconstruction from a Pair of Images
- If we do not know the corresponding features...

Epipolar Geometry
- Epipolar plane
  - Defined by the centers-of-projection $C$, $C'$ and point $X$
Epipolar Geometry

- Epipolar constraint: even if the distance from C to X is unknown, we know its projection on C' must move along the epipolar line
  - This means 1-degree of freedom…

Example epipolar lines: converging cameras

Example epipolar lines: lateral motion

Example epipolar lines: forward motion

Correspondence

- Find a feature in the other image of the pair
  - Epipolar constraint reduces the problem to a 1D search
  - But still a fundamentally difficult problem!!!
Depth Estimation

- How far a projected feature moves along the epipolar line is inversely proportional to the depth (distance) to the feature
- How "big" is a pixel?
- What is the precision of reconstruction from two images?

Depth Estimation

Image
Intensity = 1 / Depth
"Perfect example"

Depth Estimation

- More realistic examples of what you can expect...

Normalized correlation
Simulated annealing

Computer Vision Approach

- Calculate pixel correspondences and extract geometry
  - Not robust
  - Difficult to acquire illumination effects, e.g. specular highlights

[Pollefeys99]