3D Sensing and Reconstruction

Readings: Ch 12: 12.5-6, Ch 13: 13.1-3, 13.9.4

- Perspective Geometry
- Camera Model
- Stereo Triangulation
- 3D Reconstruction by Space Carving
3D Shape from X means getting 3D coordinates from different methods

- shading
- silhouette
- texture

- stereo
- light striping
- motion

mainly research

used in practice
Perspective Imaging Model: 1D

This is the axis of the real image plane.

O is the center of projection.

This is the axis of the front image plane, which we use.

\[
\frac{x_i}{f} = \frac{x_c}{z_c}
\]
Perspective in 2D (Simplified)

3D object point

\[ P = (x_c, y_c, z_c) = (x_w, y_w, z_w) \]

\[ \frac{x_i}{f} = \frac{x_c}{z_c} \]

\[ \frac{y_i}{f} = \frac{y_c}{z_c} \]

Here camera coordinates equal world coordinates.

\[ x_i = (f/z_c)x_c \]

\[ y_i = (f/z_c)y_c \]
3D from Stereo

3D point

d = x_{left} - x_{right}

disparity: the difference in image location of the same 3D point when projected under perspective to two different cameras.
Depth Perception from Stereo
Simple Model: Parallel Optic Axes

\[
\frac{z}{f} = \frac{x}{x_l} \quad \frac{z}{f} = \frac{x-b}{x_r} \quad \frac{z}{f} = \frac{y}{y_l} = \frac{y}{y_r}
\]

\(y\)-axis is perpendicular to the page.
Resultant Depth Calculation

For stereo cameras with parallel optical axes, focal length \( f \), baseline \( b \), corresponding image points \((x_l,y_l)\) and \((x_r,y_r)\) with disparity \( d \):

\[
\begin{align*}
    z &= \frac{f \cdot b}{(x_l - x_r)} = \frac{f \cdot b}{d} \\
    x &= x_l \cdot \frac{z}{f} \quad \text{or} \quad b + x_r \cdot \frac{z}{f} \\
    y &= y_l \cdot \frac{z}{f} \quad \text{or} \quad y_r \cdot \frac{z}{f}
\end{align*}
\]

This method of determining depth from disparity is called \textit{triangulation}. 
Finding Correspondences

• If the correspondence is correct, triangulation works VERY well.

• But correspondence finding is not perfectly solved. (What methods have we studied?)

• For some very specific applications, it can be solved for those specific kind of images, e.g. windshield of a car.
3 Main Matching Methods

1. Cross correlation using small windows.

2. Symbolic feature matching, usually using segments/corners.

3. Use the newer interest operators, i.e. SIFT.
Epipolar Geometry Constraint: 1. Normal Pair of Images

The epipolar plane cuts through the image plane(s) forming 2 epipolar lines.

The match for $P_1$ (or $P_2$) in the other image, must lie on the same epipolar line.
Epipolar Geometry: General Case
1. Epipolar Constraint: Matching points lie on corresponding epipolar lines.

2. Ordering Constraint: Usually in the same order across the lines.
Structured Light

3D data can also be derived using

• a single camera

• a light source that can produce stripe(s) on the 3D object
Structured Light
3D Computation

3D data can also be derived using

- a single camera
- a light source that can produce stripe(s) on the 3D object

\[
[b \quad x \quad y \quad z] = \frac{[x \quad y \quad f]}{f \cot \theta - x} \quad \text{(image)}
\]

\[
[3D \quad f \cot \theta - x] \quad \text{(3D point (x, y, z))}
\]
Depth from Multiple Light Stripes

What are these objects?
Our (former) System
4-camera light-striping stereo

3D object
Camera Model: Recall there are 5 Different Frames of Reference

- Object
- World
- Camera
- Real Image
- Pixel Image
The Camera Model

How do we get an image point IP from a world point P?

\[
\begin{pmatrix}
  s & IP_r \\
  s & IP_c \\
  s & 
\end{pmatrix}
= 
\begin{pmatrix}
  c_{11} & c_{12} & c_{13} & c_{14} \\
  c_{21} & c_{22} & c_{23} & c_{24} \\
  c_{31} & c_{32} & c_{33} & 1 \\
\end{pmatrix}
\begin{pmatrix}
  Px \\
  Py \\
  Pz \\
  1 \\
\end{pmatrix}
\]

image point  camera matrix C  world point

What’s in C?
The camera model handles the **rigid body** transformation from world coordinates to camera coordinates plus the **perspective** transformation to image coordinates.

1. \[ \text{CP} = \text{T} \text{R} \text{ WP} \]
2. \[ \text{FP} = \pi(f) \text{ CP} \]

Why is there not a scale factor here?
Camera Calibration

• In order work in 3D, we need to know the parameters of the particular camera setup.

• Solving for the camera parameters is called calibration.

  • intrinsic parameters are of the camera device

  • extrinsic parameters are where the camera sits in the world
Intrinsic Parameters

- principal point \((u_0, v_0)\)
- scale factors \((d_x, d_y)\)
- aspect ratio distortion factor \(\gamma\)
- focal length \(f\)
- lens distortion factor \(\kappa\)
  (models radial lens distortion)
Extrinsic Parameters

- Translation parameters
  \[ t = [t_x \ t_y \ t_z] \]

- Rotation matrix
  \[
  R = \begin{pmatrix}
  r_{11} & r_{12} & r_{13} & 0 \\
  r_{21} & r_{22} & r_{23} & 0 \\
  r_{31} & r_{32} & r_{33} & 0 \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \]

Are there really nine parameters?
The idea is to snap images at different depths and get a lot of 2D-3D point correspondences.
The Tsai Procedure

• The Tsai procedure was developed by Roger Tsai at IBM Research and is most widely used.

• Several images are taken of the calibration object yielding point correspondences at different distances.

• Tsai’s algorithm requires \( n > 5 \) correspondences

\[
\{(x_i, y_i, z_i), (u_i, v_i) \mid i = 1,\ldots,n\}
\]

between (real) image points and 3D points.

• Lots of details in Chapter 13.
We use the camera parameters of each camera for general stereo.

\[ P_1 = (r_1, c_1) \quad P_2 = (r_2, c_2) \]
For a correspondence \((r_1,c_1)\) in image 1 to \((r_2,c_2)\) in image 2:

1. Both cameras were calibrated. Both camera matrices are then known. From the two camera equations \(B\) and \(C\) we get 4 linear equations in 3 unknowns.

\[
\begin{align*}
    r_1 &= (b_{11} - b_{31} r_1)x + (b_{12} - b_{32} r_1)y + (b_{13} - b_{33} r_1)z \\
    c_1 &= (b_{21} - b_{31} c_1)x + (b_{22} - b_{32} c_1)y + (b_{23} - b_{33} c_1)z
\end{align*}
\]

\[
\begin{align*}
    r_2 &= (c_{11} - c_{31} r_2)x + (c_{12} - c_{32} r_2)y + (c_{13} - c_{33} r_2)z \\
    c_2 &= (c_{21} - c_{31} c_2)x + (c_{22} - c_{32} c_2)y + (c_{23} - c_{33} c_2)z
\end{align*}
\]

Direct solution uses 3 equations, won’t give reliable results.
Solve by computing the closest approach of the two skew rays.

If the rays intersected perfectly in 3D, the intersection would be P. Instead, we solve for the shortest line segment connecting the two rays and let P be its midpoint.

\[ V = (P_1 + a_1 u_1) - (Q_1 + a_2 u_2) \]

\[ (P_1 + a_1 u_1) - (Q_1 + a_2 u_2) \cdot u_1 = 0 \]
\[ (P_1 + a_1 u_1) - (Q_1 + a_2 u_2) \cdot u_2 = 0 \]
Surface Modeling and Display from Range and Color Data

Kari Pulli UW
Michael Cohen MSR
Tom Duchamp UW
Hugues Hoppe MSR
John McDonald UW
Linda Shapiro UW
Werner Stuetzle UW

UW = University of Washington
   Seattle, WA USA

MSR = Microsoft Research
     Redmond, WA USA
Introduction

Goal

- develop robust algorithms for constructing 3D models from range & color data

- use those models to produce realistic renderings of the scanned objects
Surface Reconstruction

Step 1: Data acquisition
Obtain range data that covers the object. Filter, remove background.

Step 2: Registration
Register the range maps into a common coordinate system.

Step 3: Integration
Integrate the registered range data into a single surface representation.

Step 4: Optimization
Fit the surface more accurately to the data, simplify the representation.
Problem

Noisy registered data
Signed distance fn & marching cubes
Hierarchical & directional space carving
Carve space in cubes

Label cubes

- Project cube to image plane (hexagon)
- Test against data in the hexagon
Several views

Processing order:
  FOR EACH cube
    FOR EACH view

Rules:
  any view thinks cube's out
    => it's out
  every view thinks cube's in
    => it's in
  else
    => it's at boundary
Hierarchical space carving

- Big cubes $\Rightarrow$ fast, poor results
- Small cubes $\Rightarrow$ slow, more accurate results
- Combination = octrees

RULES:
- cube's out $\Rightarrow$ done
- cube's in $\Rightarrow$ done
- else $\Rightarrow$ recurse
Hierarchical space carving

- Big cubes => fast, poor results
- Small cubes => slow, more accurate results
- Combination = octrees

RULES:
- cube’s out => done
- cube’s in => done
- else => recurse
The rest of the chair
Same for a husky pup
Optimizing the dog mesh

Registered points

Initial mesh

Optimized mesh
View dependent texturing