Photometric Stereo
& motion

Introduction to Computer Vision
CSE 152
Lecture 13

- Midterm will be graded before Monday
- HW3 will be out by tomorrow
Multi-view stereo vs. Photometric Stereo: Assumptions

- Multi-view (binocular) Stereo
  - Multiple images
  - Dynamic scene
  - Multiple viewpoints
  - Fixed lighting

- Photometric Stereo
  - Multiple images
  - Static scene
  - Fixed viewpoint
  - Multiple lighting conditions

Photometric Stereo Process

1. From a single viewpoint of a static scene, acquire $k$ images under different lighting.
2. Independently, estimate the surface normal for each pixel location from $k$ measurements.
3. Integrate the normals to estimate depth across the surface.
Assumptions

We generally make the following simplifying assumptions:
1. Light source is distant
2. BRDF is known and has a few parameters

Coordinate system

Surface: \( \mathbf{s}(x,y) = (x,y, f(x,y)) \)

Tangent vectors:
\[
\frac{\partial \mathbf{s}(x,y)}{\partial x} = \begin{pmatrix} 1, 0, \frac{\partial f}{\partial x} \end{pmatrix} \\
\frac{\partial \mathbf{s}(x,y)}{\partial y} = \begin{pmatrix} 0, 1, \frac{\partial f}{\partial y} \end{pmatrix}
\]

Normal vector
\[
\mathbf{n} = \frac{\partial \mathbf{s}}{\partial x} \times \frac{\partial \mathbf{s}}{\partial y} = \begin{pmatrix}
-\frac{\partial f}{\partial x} & -\frac{\partial f}{\partial y} & 1
\end{pmatrix}
\]
Not unit length
Photometric Stereo for Lambertian Surface with Distant Known Lighting

- When light source is “distant”, like the sun
  1. The direction to the light source is constant over the surface.
  2. The distance from points on the surface to light location is constant, and so brightness of light doesn’t vary.
- So, we can model as a having constant direction $s$ and strength $s_0$
Lambertian Surface

At image location \((u,v)\), the intensity of a pixel \(x(u,v)\) is:

\[
e(u,v) = [a(u,v) \hat{n}(u,v)] \cdot [s_0 \hat{s}]
\]

where
- \(a(u,v)\) is the albedo of the surface projecting to \((u,v)\)
- \(n(u,v)\) is the direction of the surface normal
- \(s_0\) is the light source intensity
- \(s\) is the direction to the light source

Let
\[
b(u,v) = a(u,v)\hat{n}(u,v)
\]
\[
s = s_0 \hat{s}
\]

Lambertian Photometric stereo

- If the light sources \(s_1\), \(s_2\), and \(s_3\) are known, then we can recover \(b\) from as few as three images. (Photometric Stereo: Silver 80, Woodham81).
- For a single pixel location in 3 gray scale images, we have

\[
[e_1 \ e_2 \ e_3] = b^T [s_1 \ s_2 \ s_3]
\]

where \(e_1\), \(e_2\), and \(e_3\) are measured pixel intensities and we know \(s_1\), \(s_2\), and \(s_3\). We can then compute \(b\) at the pixel location by solving a linear system.

\[
b^T = [e_1 \ e_2 \ e_3][s_1 \ s_2 \ s_3]^{-1}
\]

- Normal is: \(\hat{n} = b/|b|\), albedo is: \(a = |b|\)
Computing $\mathbf{b}$ for every pixel

- Slow way: Loop over every pixel and apply equations in previous slide.
- Better: For an image with $n$ pixels,
  - Let
    \[
    \mathbf{E} = \begin{bmatrix}
    e_{1,1} & e_{1,2} & e_{1,3} \\
    e_{2,1} & e_{2,2} & e_{2,3} \\
    \vdots & \vdots & \vdots
    \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}
    b_{1,1} & b_{1,2} & b_{1,3} \\
    b_{2,1} & b_{2,2} & b_{2,3} \\
    \vdots & \vdots & \vdots
    \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix}
    s_1 \\
    s_2 \\
    s_3
    \end{bmatrix}
    \]
  - where $e_{ij}$ is the measured image intensity at pixel $j$ in image $i$, and row $i$ of $\mathbf{B}$ is the surface normal scaled by the albedo $\mathbf{b}$ for pixel $i$.
  - So image formation equation is $\mathbf{E} = \mathbf{BS}$
  - Solving for $\mathbf{B}$, we have: $\mathbf{B} = \mathbf{ES}^{-1}$

Normal Field – Normals at every location
What if we have more than 3 Images?
Linear Least Squares

\[
\begin{bmatrix} e_1 & e_2 & e_3 & \ldots & e_n \end{bmatrix} = \mathbf{b}^T \begin{bmatrix} s_1 & s_2 & s_3 & \ldots & s_n \end{bmatrix}
\]

Let the residual be
\[ r = e - S\mathbf{b} \]

Squaring this:
\[ r^2 = r^T r = (e - S\mathbf{b})^T (e - S\mathbf{b}) = e^T e - 2\mathbf{b}^T S^T e + \mathbf{b}^T S^T S \mathbf{b} \]

Zero derivative is a necessary condition for a minimum
\[
\frac{\partial r^2}{\partial \mathbf{b}} = 0 = -2S^T e + 2S^T S \mathbf{b}
\]

Solving for \( \mathbf{b} \) gives
\[
\mathbf{b} = (S^T S)^{-1} S^T e
\]

Plastic Baby Doll: Normal Field
Next step:
Going from normal field to surface

Depth recovery as solution to partial differential equation (pde)

- From estimated \( \mathbf{n} = (n_x, n_y, n_z) \) at each pixel \((x, y)\), compute
  \[ p = -\frac{n_x}{n_z} \]
  \[ q = -\frac{n_y}{n_z} \]
- But \( p \) and \( q \) are just just the partial derivatives of the depth function \( f(x, y) \) with respect to \( x \) and \( y \).
- System of two first order partial differential equations (pde)
  \[ \frac{\partial f(x, y)}{\partial x} = p(x, y) \]
  \[ \frac{\partial f(x, y)}{\partial y} = q(x, y) \]
- Assume surface and derivatives are continuous and then solve pde for \( f(x, y) \)
Step 2: Recovering the surface $f(x,y)$

Many methods: Simplest approach

1. Integrate $\frac{\partial f}{\partial x} = p(x,y)$ along the top row $(x,0)$ to get $f(x,0)$

$$f(x,0) = \int_{x'=0}^{x'} p(x',0) \, dx'$$

or for a discrete image $f(i,0) = \sum_{j=1}^{i} p(j,0) = f(i-1,0) + p(i,0)$

2. Then integrate $\frac{\partial f}{\partial y} = q(x,y)$ along each column starting with $f(x,0)$ of the first row

What might go wrong?

• Height $z(x,y)$ is obtained by integration along a curve from $(x_0, y_0)$.

$$z(x, y) = z(x_0, y_0) + \int_{(x_0,y_0)}^{(x,y)} (p \, dx + q \, dy)$$

• If one integrates the derivative field along any closed curve, one expects to get back to the starting value.

• Might not happen because of noisy estimates of $(p,q)$
What might go wrong?

Integrability. If \( f(x,y) \) is the height function, we expect that

\[
\frac{\partial}{\partial y} \frac{\partial f}{\partial x} = \frac{\partial}{\partial x} \frac{\partial f}{\partial y}
\]

In terms of estimated gradient \((p,q)\), this means:

\[
\frac{\partial p}{\partial y} = \frac{\partial q}{\partial x}
\]

where \( p = -n_x/n_z \), \( q = -n_y/n_z \) with \( n = [n_x \ n_y \ n_z] \)

But \( n \) (and in turn \( p, q \)) are estimated independently at each pixel, equality is not going to exactly hold

Horn’s Method

[“Robot Vision, B.K.P. Horn, 1986”]

- Formulate estimation of surface height \( z(x,y) \) from gradient field by minimizing cost functional:

\[
\iint_{\text{Image}} (z_x - p)^2 + (z_y - q)^2 \, dx \, dy
\]

where \((p,q)\) are estimated components of the gradient while \( z_x \) and \( z_y \) are partial derivatives of best fit surface
- Solved using calculus of variations – iterative updating
- \( z(x,y) \) can be discrete or represented in terms of basis functions.
- Integrability is naturally satisfied.
Input Images

Recovered albedo
Recovered normal field
(shown on recovered 3D structure)

Surface recovered by integration
Lambertian Photometric Stereo

Constant albedo map

Reconstruction with Albedo Map
Another person

Constant Albedo map
**Generalizations**

1. **Non-Lambertian BRDF**
   - For Lambertian surface, we estimated 3 parameters at each pixel (albedo, normal) from 3 images
   - For general BRDF $\rho(\omega_i, \omega_o)$ that has $k$ parameters, we have $k+2$ parameters and need at least $k+2$ images.

2. **Nearby light sources**
   - equations become nonlinear and so harder to solve

3. **Unknown lighting**
   - For Lambertian surface, what if $S$ matrix is unknown
   - Can recover surface up to a Generalized Bas Relief Transformation

---

**Generalized Bas-Relief Ambiguity**

From 3 or more images of a Lambertian surface where the lighting is unknown, the shape and lights can only be recovered up to a Generalized Bas-Relief transformation

$$\tilde{f}(x, y) = \lambda f(x, y) + \mu x + \nu y$$

where $f(x, y)$ is the depth of the true surface.

Shading and shadows are the same for any $\lambda$, $\mu$, $\nu$

[Belhumeur, Kriegman, Yuille 1999]
Bas-Relief Sculpture

Motion

- https://www.youtube.com/watch?v=TKsVVmOGV9I
Estimate 3D structure from images
Structure-from-Motion (SFM)

Given two or more images or video w/o any information on camera position/motion as input, estimate camera motion and 3-D structure of a scene.

Two Approaches
1. Discrete motion (wide baseline)
2. Continuous (Infinitesimal) motion usually from video

Random Dot Kinnetogram

- Motion reveals structure

- Ego motion
  - https://www.youtube.com/watch?v=y4JRamQvKrM

- 3D structure from motion + ambiguity
  - https://www.youtube.com/watch?v=DLBkwig3M2U
Small Motion

Motion

“When objects move at equal speed, those more remote seem to move more slowly.”
- Euclid, 300 BC
The Motion Field
Where in the image did a point move?

Down and left
Motion field

- The motion field is the projection of the 3D scene motion into the image
Motion blur.
Usually in direction of motion field

What causes a motion field?

1. Camera moves (translates, rotates)
2. Object’s in scene move rigidly
3. Objects articulate (pliers, humans, animals)
4. Objects bend and deform (fish)
5. Blowing smoke, clouds
6. Multiple movements
An example motion field:
Camera moving straight along optical axis

The “instantaneous” velocity of all points in an image

LOOMING

The Focus of Expansion (FOE)

Intersection of velocity vector with image plane

With just this information it is possible to calculate:

1. Direction of motion
2. Time to collision

Rigid Motion: General Case

Position and orientation of a rigid body
Rotation Matrix & Translation vector

Rigid Motion:

Velocity Vector: \( T \)
Angular Velocity Vector: \( \omega \)

\[ \dot{p} = T + \omega \times p \]
General Motion

\[
\begin{bmatrix}
    u \\
    v
\end{bmatrix} = \frac{f}{z} \begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]

Let \((x,y,z)\) be functions of time \((x(t), y(t), z(t))\):

\[
\begin{bmatrix}
    \dot{u} \\
    \dot{v}
\end{bmatrix} = \frac{f}{z} \begin{bmatrix}
    \dot{x} \\
    \dot{y}
\end{bmatrix} - \frac{f\dot{z}}{z^2} \begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]

\[
= \frac{f}{z} \begin{bmatrix}
    \dot{x} \\
    \dot{y}
\end{bmatrix} - \frac{\dot{z}}{z} \begin{bmatrix}
    u \\
    v
\end{bmatrix}
\]

Substitute \(\dot{p} = T + \omega \times p\) where \(p=(x,y,z)^T\)

---

Motion Field Equation

\[
\begin{align*}
\dot{u} &= \frac{z}{z} \frac{T u - T f}{z} - \omega_y f + \omega_z v + \frac{\omega_y uv}{f} - \frac{\omega_y u^2}{f} \\
\dot{v} &= \frac{z}{z} \frac{T v - T f}{z} + \omega_x f - \omega_z u - \frac{\omega_y uv}{f} - \frac{\omega_x v^2}{f}
\end{align*}
\]

- **Image**
  - \((u,v)\): Image point coordinates
  - \((\dot{u},\dot{v})\): Image point velocity

- **Camera**
  - \(T\): Components of 3-D linear motion
  - \(\omega\): Angular velocity vector
  - \(f\): focal length

- **Scene**
  - \(z\): depth
**Pure Translation**

If camera is just translating with velocity \((T_x, T_y, T_z)\), there’s no rotation:

\[
\omega = 0
\]

\[
\dot{u} = \frac{T_x u - T_y f}{Z} - \omega_y f + \omega_z v + \frac{\omega_z u v}{f} - \frac{\omega_z v^2}{f}
\]

\[
\dot{v} = \frac{T_y v - T_x f}{Z} + \omega_x f - \omega_z u - \frac{\omega_z u v}{f} - \frac{\omega_z v^2}{f}
\]

- \(\dot{u}, \dot{v}\) is inversely proportional to \(Z\) (remember Euclid)
- Focus of expansion is located at \((u, v)\) where \(\dot{u} = \dot{v} = 0\)

---

**Forward Translation & Focus of Expansion**

[Gibson, 1950]
Focus of Expansion (FOE)

\[ \hat{u} = \frac{T_z u - T_x f}{Z} \]
\[ \hat{v} = \frac{T_z v - T_y f}{Z} \]

- Focus of expansion is located at \((u,v)\) where \( \hat{u} = \hat{v} = 0 \)

\[ T_z u - T_x f = 0 \]
\[ T_z v - T_y f = 0 \]

- Solve for \(u,v\)

\[ u = f \frac{T_x}{T_z} \]
\[ v = f \frac{T_y}{T_z} \]

Insight: The FOE is the perspective projection of the linear velocity vector \((T_x, T_y, T_z)\).

Pure Translation

Radial about FOE

Parallel (FOE point at infinity)

\[ T_Z = 0 \]

Motion parallel to image plane
Sideways Translation

[Gibson, 1950]

Parallel
(FOE point at infinity)

\[ T_z = 0 \]

Motion parallel to image plane

Pure Rotation: \( T=0 \)

\[
\begin{align*}
\dot{u} &= \frac{f}{Z} (u - T_x f) - \omega_y f + \omega_z v + \frac{\omega_z uv}{f} - \frac{\omega_y^2}{f} \\
\dot{v} &= \frac{f}{Z} (v - T_y f) + \omega_x f - \omega_z u - \frac{\omega_x uv}{f} - \frac{\omega_x v^2}{f}
\end{align*}
\]

• Independent of \( T_x \), \( T_y \), \( T_z \)
• Independent of \( Z \)
• Only function of image plane position \((u,v)\), \( f \) and \( \omega \)
Rotational MOTION FIELD

The “instantaneous” velocity of points in an image

PURE ROTATION

\[ \omega = (0,0,1)^T \]

Motion Field Equation: Depth Estimation

\[
\begin{align*}
\dot{u} &= \frac{T_z u - T_x f}{Z} - \omega_y f + \omega_z v + \frac{\omega_x u v}{f} - \frac{\omega_y u^2}{f} \\
\dot{v} &= \frac{T_z v - T_y f}{Z} + \omega_x f - \omega_z u - \frac{\omega_y u v}{f} - \frac{\omega_x v^2}{f}
\end{align*}
\]

If \( T, \omega, \) and \( f \) are known or measured, then for each image point \((u,v)\), one can solve for the depth \( Z \) given measured motion \( \dot{u}, \dot{v} \) at \((u,v)\).

\[
Z = \frac{T_z u - T_x f}{\dot{u} + \omega_y f - \omega_z v - \frac{\omega_x u v}{f} + \frac{\omega_y u^2}{f}}
\]

Inversely proportional to image velocity.