CSE 152 Section 7
HW3: Photometric Stereo and Optical Flow

May 20, 2019
Owen Jow
[1a] Lambertian Photometric Stereo

Input:
images and associated lighting information

Output:
normals, albedo, depth

Photometric Stereo
[1a] Assumptions

- **Orthographic camera model:**
  
  $(x, y, z)$ projects to $(x, y)$ → our goal will be to recover the height/depth map $z = f(x, y)$

- **Distant lighting:**
  
  treat every pixel in one image as sharing the same lighting direction/intensity

- **Static scene/viewpoint:**
  
  $(x, y)$ in one image corresponds to $(x, y)$ in all of the other images
[1a] Assumptions, cont. (Lambertian)

- **Lambertian surface:**
  
  Assume that the surface being imaged is Lambertian, i.e., at any point on the surface, there is equal reflectance in all directions.

  \[ e(x, y) = [a(x, y) \mathbf{n}(x, y)] \cdot \mathbf{s} \]
  
  \[ = b(x, y) \cdot \mathbf{s} \]

<table>
<thead>
<tr>
<th>e(x, y)</th>
<th>intensity at pixel (x, y)</th>
<th>(known)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(x, y)</td>
<td>albedo at the point on the surface corresponding to (x, y)</td>
<td>(unknown)</td>
</tr>
<tr>
<td>n(x, y)</td>
<td>unit normal at the point on the surface corresponding to (x, y)</td>
<td>(unknown)</td>
</tr>
<tr>
<td>s</td>
<td>unit direction to the light, scaled by the intensity of the light</td>
<td>(known)</td>
</tr>
</tbody>
</table>
[1a] Solving for $\mathbf{b}$ at Each Pixel

$b$ is a 3-vector, so we need at least three equations/images. Let’s say we have $n$. 

$$
\begin{bmatrix}
e_1 \\
\vdots \\
e_n
\end{bmatrix} = 
\begin{bmatrix}
s_1 \cdot \mathbf{b} \\
\vdots \\
s_n \cdot \mathbf{b}
\end{bmatrix}
$$

$$
\begin{bmatrix}
e_1 \\
\vdots \\
e_n
\end{bmatrix} = 
\begin{bmatrix}
- & s_1 & - \\
- & \vdots & - \\
- & s_n & -
\end{bmatrix} 
\mathbf{b}
$$

$\mathbf{e} = \mathbf{S} \mathbf{b}$

Line: $\mathbf{e} = \mathbf{s} \cdot \mathbf{b}$

Linear least squares (see lecture)

$$
\mathbf{b} = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{e}
$$

Albedo $a = \|\mathbf{b}\|$

Normal $\mathbf{n} = \frac{\mathbf{b}}{\|\mathbf{b}\|}$
[1a] $\n$ Encodes the Partial Derivatives of Depth

Note: this uses a left-handed coordinate system, whereas lecture uses a right-handed one.

\[
\n = \text{normalized} \left( \begin{bmatrix} 0 \\ 1 \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ \frac{\partial f(x,y)}{\partial x} \end{bmatrix} \right)
\]

\[
= \frac{1}{\sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2 + 1}} \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \\ -1 \end{bmatrix}
\]

\[
\frac{\partial f(x,y)}{\partial x} = -\frac{n_1}{n_3}
\]

\[
\frac{\partial f(x,y)}{\partial y} = -\frac{n_2}{n_3}
\]
[1a] Simple Scanline Integration 1

Once we have the partial derivatives, we can integrate to get depth.

Initialize the top-left corner of the height map to 0
Once we have the partial derivatives, we can integrate to get depth.

Initialize the top-left corner of the height map to 0
For each pixel in the leftmost column [except (0, 0)]:
height = height of above pixel - \( \partial f / \partial y \)
Once we have the partial derivatives, we can integrate to get depth.

Initialize the top-left corner of the height map to 0.
For each pixel in the leftmost column [except (0, 0)]:
  height = height of above pixel - \( \frac{\partial f}{\partial y} \)
For each pixel (except the leftmost) in each row:
  height = height of pixel to the left - \( \frac{\partial f}{\partial x} \)
[1a] The Mask Parameter

- 1s for locations to use during integration
- 0s for locations to ignore during integration
- Unnecessary for 1a, but can use to filter out the background in 1b

\[(\text{img} - \text{img.min}) / (\text{img.max} - \text{img.min})\]
Applicability of the Lambertian Equation

The Lambertian equation does not apply for

- **shadowed regions**, where the view of the light is blocked
- **specularities**, which arise as a result of a different relationship

If we perform photometric stereo on such locations, we'll end up with noisy/pointy/bumpy artifacts.
[1b] Simple Shadow/Specularity Removal

Identify shadowed/specular locations based on brightness. Then, either...

1. ...clamp each associated brightness to a threshold value.
2. ...set each associated brightness to the median of a sizable surrounding window.

Do this before solving for \( b \). You should be able to mitigate the artifact(s) to some degree.
Estimate the apparent motion of each pixel from frame A to frame B.
[2a] Optical Flow 2

Estimate the apparent motion of each pixel from frame A to frame B.
[2a] The Brightness Constancy Equation

Assumption 1: the brightness/color of each pixel remains constant as it moves.

\[ I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) \]

Assumption 2: pixels don’t move too far between frames. Linearizing via Taylor expansion:

\[ I(x, y, t) + \Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t} = I(x, y, t) \]

\[ \Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t} = 0 \]

\[ \frac{\Delta x}{\Delta t} \frac{\partial I}{\partial x} + \frac{\Delta y}{\Delta t} \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0 \]

\[ uI_x + vI_y + I_t = 0 \]

Goal: solve for \( u, v \) at every pixel. Can compute \( I_x, I_y, I_t \) from the images.
The Lucas-Kanade Method

Problem: one equation, two unknowns.

**Assumption 3:** flow is constant in the neighborhood around each pixel.

→ Get one equation for every point in a window around each pixel.

\[
\begin{bmatrix}
(I_x)_1 & (I_y)_1 \\
\vdots & \vdots \\
(I_x)_n & (I_y)_n
\end{bmatrix}
\begin{bmatrix}
u \\ v
\end{bmatrix} =
\begin{bmatrix}
-(I_t)_1 \\
\vdots \\
-(I_t)_n
\end{bmatrix}
\]

[\text{linear least squares}]

\[
\begin{bmatrix}
(I_x)_1 & (I_y)_1 \\
\vdots & \vdots \\
(I_x)_n & (I_y)_n
\end{bmatrix}
\begin{bmatrix}
u \\ v
\end{bmatrix} =
\begin{bmatrix}
-(I_t)_1 \\
\vdots \\
-(I_t)_n
\end{bmatrix}
\]

\[
\begin{bmatrix}
I_x^2 & I_xI_y \\
I_xI_y & I_y^2
\end{bmatrix}
\begin{bmatrix}
u \\ v
\end{bmatrix} =
\begin{bmatrix}
-I_xI_t \\
-I_yI_t
\end{bmatrix}
\]

the second moment matrix strikes again
Notes

- Use the pseudoinverse to solve for \([u, v]^T\)

\[
\begin{bmatrix}
  u \\
  v
\end{bmatrix}
= \left( \sum_{x,y \in \mathcal{W}} \begin{bmatrix}
  I_x^2 & I_x I_y \\
  I_x I_y & I_y^2
\end{bmatrix} \right) ^ \dagger \left( \sum_{x,y \in \mathcal{W}} \begin{bmatrix}
  -I_x I_t \\
  -I_y I_t
\end{bmatrix} \right)
\]

- Run this for every pixel (loops are fine)
- Make sure you compute the y-gradient with respect to an upward axis
  - If you use `np.gradient` to compute the image gradients, negate the y-gradient you get back
[3b] RANSAC for Estimating the Focus of Expansion

Idea: repeatedly

- Sample two flow vectors
- Estimate the focus of expansion as their intersection
- Check the consistency of the estimate across all flow vectors
[3b] Ray-Ray Intersection

- Solve for $t_1$ and $t_2$. Derivation of the exact solution is left as an exercise. :)
- Note that there are other ways to compute intersections. Use whichever method you like.
[3b] Distance from a Point to a Ray

When checking consistency, you’ll need to compute the perpendicular distance from your estimated focus of expansion to each of the rays (or to some subset of them).

Here’s a reference for that.

[3b] Example Results

My parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance threshold</td>
<td>100</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>100</td>
</tr>
<tr>
<td>Random seed</td>
<td>15</td>
</tr>
</tbody>
</table>

My bestInliersNumList plot:

Estimated focus of expansion is $(y, x) = (137, 107)$
Flow vector at that location is $(u, v) = (0.174968, 0.094821)$