CSE 252A: Motion II

Lecturer: David Kriegman

Scribed by Owen Jow on November 15, 2018

1 SfM Recap

1.1 Two-Frame, Wide Baseline

With RANSAC, SfM looks like

1. Detect feature points in each image.
2. Hypothesize matches based on, e.g., NCC.
3. With these hypothesized matches, estimate $\mathbf{E}$ using RANSAC (and the eight-point algorithm).
4. Compute epipolar geometry for inliers. Estimate 3D structure.

2 Small Motion Recap

$$\dot{u} = \frac{t_z u - t_x f}{z} - \omega_y f + \omega_z v + \frac{\omega_x uv}{f} - \frac{\omega_y u^2}{f}$$
$$\dot{v} = \frac{t_z v - t_y f}{z} + \omega_x f - \omega_z u - \frac{\omega_y uv}{f} + \frac{\omega_x v^2}{f}$$

- **Forward problem.** If we know the camera motion $\mathbf{t}$, ω and the scene geometry x, y, z and the focal length f, we can compute the image and the motion field.
- **Inverse problem.** If we know the camera motion $\mathbf{t}$, ω and the scene motion $\dot{u}$, $\dot{v}$ and the focal length f, we can determine the 3D scene geometry i.e. z.
- **Pure translation ($\omega = 0$).** The motion field emanates from or converges at the *focus of expansion*, the image point where $\dot{u} = \dot{v} = 0$. We can solve for this if we know $\mathbf{t}$ and f.
- **Pure rotation ($\mathbf{t} = 0$).** The motion field depicts a rotation about the axis of rotation ω.
- **Euclid.** Image motion is smaller when the motion in 3D is farther away (i.e. when z is greater).

3 Optical Flow

Given just an image sequence, we would like to compute the motion field. There are two main approaches for this: *feature-based* and *direct* (rooted in differential techniques).

The feature-based approach is pretty simple: just detect features in consecutive frames and match them, and the difference in each feature point's position is its movement. This assumes *color constancy* (*brightness constancy* for grayscale images): points should look the same across frames.

The other approach is *optical flow*. This will find motion densely over the image. (With feature-based methods, we only get a sparse field.) Optical flow refers to "apparent motion of brightness patterns." We care about how things move in image space, not how things physically move.

- If we had a textureless ball rotating in space, there would be a rotation-based motion field, but no optical flow since the image doesn't change.
- If lights move, there might be apparent motion but no actual motion in the visible 3D scene.

3.1 Optical Flow Constraint Equation

Say we have a frame at time t, and a frame at time $t + \Delta t$. Consider a point which is at (x, y) in the first frame. It will be at $(x + u\Delta t, y + v\Delta t)$ in the second frame, where u, v is the velocity. This velocity is the optical flow we want to compute.

Assuming brightness constancy,

$$I(x + u\Delta t, y + v\Delta t, t + \Delta t) = I(x, y, t)$$

(Our images are functions of time now. We can think of them as 3D volumes as in *Long-term Temporal Convolutions for Action Recognition* by Varol et al.)

We also assume small motion, so we can take a Taylor series of the LHS and linearize:

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

Then, subtracting $I(x, y, t)$ from both sides and dividing by Δt,

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

Assuming the "Δt"s are small or infinitesimal, this becomes

$$\frac{dx}{dt} \frac{\partial I}{\partial x} + \frac{dy}{dt} \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$
$$u I_x + v I_y + I_t = 0$$

This is the **optical flow constraint equation**.

Note that we can measure $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$. We want to solve for the velocities $\frac{dx}{dt}$ and $\frac{dy}{dt}$.

- To compute $\frac{\partial I}{\partial t}$,
  we can convolve a stacked set of $(t-1, t, t+1)$ images with $[1, 0, -1]$ over the time dimension.

But – we have two unknowns and only one equation (leading to the *aperture problem*[1], where when we look only locally at a pixel, there are different motions that are consistent with what we see). To get more equations, we need to include more points.

[1] where motion direction is ambiguous due to a too-small or narrowly shaped aperture

3.2 Estimating Optical Flow

To find flow over the whole image, we can solve a global optimization problem to find the flow for all points, regularizing for smoothness. The algorithm for this is the *Horn-Schunck method*.

$$\min \iint \left[(I_x u + I_y v + I_t)^2 + \lambda(\|\nabla u\|^2 + \|\nabla v\|^2) \right] dx\, dy$$

Alternatively, we can incorporate *spatial coherence* and assume that points in local windows all move the same way. Then we have an equation for every pixel in a window. By finding a least-squares solution to these equations, we obtain u, v for a point or all points in a window.

This is known as the *Lucas-Kanade method*.

$$\min_{u,v} \sum_{x,y \in W} \left[I_x(x,y)u + I_y(x,y)v + I_t(x,y) \right]^2$$

We can solve this minimization problem[2] for u and v (take derivatives, set to 0) or find the least-squares solution to the collection of equations $I_x(x,y)u + I_y(x,y)v = -I_t(x,y)$.

Either way, the formulation comes down to

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

which is equivalent to

$$\left(\sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} = - \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} I_t$$

$$\left(\sum \nabla I \nabla I^T \right) \mathbf{u} = - \sum \nabla I I_t$$

or

$$A^T A \begin{bmatrix} u \\ v \end{bmatrix} = A^T b \quad \text{if} \quad A = \begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \quad \text{and} \quad b = - \begin{bmatrix} I_t(\mathbf{x}_1) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}$$

all where we are solving for $\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$.

However, we run into trouble when $A^T A$ is not invertible (e.g. if image brightness is constant, the window is one pixel, or the gradient is constant everywhere in the patch). For best results, we want our patch to be a corner or a texture region (then $A^T A$ will be full rank).

We also run into trouble when our assumptions are violated, of course.

3.3 Iterative LK

To help achieve small motion, we can use a multi-scale algorithm.[3]

1. For each of the two frames being compared, progressively downsample by a factor of 2.

[2] We want to minimize this because $I_x u + I_y v + I_t$ is supposed to be 0.

[3] Motivation: everything is small, sub-pixel motion if you downsample your image enough.

2. Run Lucas-Kanade on the lowest-resolution pair of $(t, t+1)$ images, giving us a motion field at that coarsest resolution.
3. Upsample the motion field and warp the "next highest"-resolution t image based on the upsampled motion field. The hope is that the motion remaining is now sub-pixel.
4. Run Lucas-Kanade again at this next level, adding the correction to the current motion field. Then upsample this and warp the next-level t frame and repeat...

Basically, we get a rough sense of the motion at first, and fine-tune it as we move toward the finer scales of our image. Every time, we add the correction in flow to the previous motion field.

3.4 Other Variations

- Another variation is to assume a different motion model (instead of pure translation). For example, we might use a similarity,[4] affine, or projective transformation as our motion model. This can allow us to estimate complex models more effectively.
- We can also use an estimation method that is more robust than least-squares (maybe something less sensitive to outliers).

[4] rigid transformation (e.g. rotation, translation) plus scale