CSE 252A: Motion I

Lecturer: David Kriegman

Scribed by Owen Jow on November 13, 2018

We can get a lot of information about a scene from motion cues: where things are (in a segmentation sense or a 3D sense), where they'll be, how they're moving, etc. With *random dot kinematograms*, for instance, we use motion to localize object structure in a clutter of dots.

We might also obtain multiple views from a moving camera and treat it as a stereo problem...

1 Structure from Motion (SfM)

In SfM, we estimate 3D point locations and the relative camera pose for each view.[1]

Note: where motion is concerned, there is both *discrete motion*, where the views are widely separated, and *continuous (infinitesimal) motion*, where we can rely on only small changes in the image and take derivatives such as change in brightness over time or space. Typical SfM is a discrete motion problem and comes down to correspondence. So in this section we are discussing discrete motion.

1.1 *How many images/points do we need?*

If we have M images and N points, there are $6(M-1)$ unknowns for each of the camera transformations relative to the first camera's frame (3 for a rotation, 3 for a translation), $3N$ unknowns for the x, y, z location of each point, and finally one less unknown because of the scale ambiguity (the scene might be faraway and large, or nearby and small). In total there are $6(M-1)+3N-1$ unknowns.

Meanwhile, we get $2MN$ measurements (for the u, v coordinates of each point in each image).

Thus a solution is possible when $2MN \geq 6(M-1)+3N-1$. (If $M=2$, N must be ≥ 5.)

1.2 Two-View SfM

1. Detect feature points; *find 8 correspondences across the images.*
2. Compute the essential matrix $\mathbf{E}$ using the eight-point algorithm.
   - Factorize into $\mathbf{R}$ and $\mathbf{t}$.
3. Perform stereo matching using $\mathbf{E}$.
4. Reconstruct the 3D positions of corresponding feature points.

[1] This is the difference between SfM and stereo up to this point – now we don't know what the camera poses are.

1.3 RANSAC

To match features (typically points) with the goal of estimating a particular transformation, we can use *RANdom SAmple Consensus*, or RANSAC. RANSAC is used to fit a model to data in the case where some of the data might not be part of the model (*outliers*). In our case, some of our pairs of features might not actually be matches. And one bad match will ruin our model.

To do model fitting in the presence of outliers, we can repeat the following process (up to) N times:

- Select some points (the minimum number needed to estimate the model) at random.
- Create a model based on those points.
- Count the total number of points that agree with the model up to some threshold (*inliers*).
  - If the number of inliers is greater than some threshold, we can terminate.

At the end of the day, re-estimate the model using the largest recorded set of inliers.

1.3.1 How many iterations?

If the proportion of outliers in the data is e, then the probability p of getting a size-s sample set (where s is the minimum size) of *only* inliers within N iterations is

$$\underbrace{1-p}_{\substack{\text{prob. that no random}\\ \text{sample is outlier-free}}} = (1 - \underbrace{(1-e)^s}_{\substack{\text{prob. of}\\ \text{getting all inliers}}})^N$$

$$p = 1 - (1 - (1-e)^s)^N$$

meaning that if we want a guarantee p of 0.99, we should set N to

$$N = \frac{\log(1-0.99)}{\log\left(1-(1-e)^s\right)}$$

1.3.2 RANSAC for the Essential Matrix

In this case, the model is $\mathbf{E}$, we need 5-8 correspondences between two images, we can compute inlier/outlier distance for one correspondence as the distance of one point from the epipolar line of the other point, and we will guess that there are a lot of outliers (just pick a large N).

2 Small Motion

Instead of processing images from "over here" and "over there," let's process images that come from "continuous" motion (i.e. images which exhibit only small, "infinitesimal" motion).

We would like to know where points have moved (from one image to another image).

If we have this information for every location in the image, we can represent it as a motion field.

2.1 Motion Fields

A motion field is the projection of *actual* 3D scene motion onto the image.

It is the "instantaneous velocity of all points in the image."

It can be caused by camera movement, object movement/articulation/deformation, or a combination of movements. We won't really worry about articulation and deformation. For our purposes, a motion field is caused by the camera moving around a rigid scene.

2.1.1 Looming Motion

- moving toward something (things closer to you appear to move faster)
- motion direction streams outward from a point, the *focus of expansion*

From a looming motion field, we can calculate the *direction of camera motion* (the focus of expansion is the intersection of the camera's velocity vector with the image plane) and the *time to collision* (if we assume the camera moves at a constant speed) based on how far apart in time the frames were.

2.2 3D Motion Types

We would like to relate the motion field to the 3D motion of the camera or object.

- **Rigid motion:** rotation and translation of rigid body, represented as linear velocity $\mathbf{t}$ and angular velocity ω since things are rotating and translating simultaneously.

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \quad \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

  - We can represent pose (position/orientation) with a rotation matrix & translation vector.
  - **Angular velocity**: rotating around an axis of direction ω, magnitude of rotation is $\|\omega\|$
  - The overall motion (velocity) of point $\mathbf{p}$ is

$$\dot{\mathbf{p}} = -\mathbf{t} - \omega \times \mathbf{p}$$

For perspective projection $u = fx/z$, $v = fy/z$, motion is

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

If $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, we can substitute $\dot{\mathbf{p}} = -\mathbf{t} - \omega \times \mathbf{p}$:

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

to obtain a motion field equation ($\dot{u}$ and $\dot{v}$ are how fast point (u, v) is moving in the image).

There are special cases of this equation:

- **Pure translation (no rotation):** eliminate ω, leaving

$$\dot{u} = \frac{t_z u - t_x f}{z}$$
$$\dot{v} = \frac{t_z v - t_y f}{z}$$

  To find the focus of expansion, we can solve for the point (u, v) where $\dot{u}, \dot{v} = 0$ (point of no motion). Sometimes this might be at infinity, e.g. for motion parallel to the image plane.

- **Pure rotation:** eliminate $\mathbf{t}$, leaving

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

  Note that these equations are independent of depth (z). Here we can think of the image plane as a sphere; the motion field will be lines of latitude on the sphere around the axis of rotation.

Finally, we can use these equations to estimate depth. If we know $\mathbf{t}$, ω, and f, we can solve for z at each image point (u, v) given the measured motion $\dot{u} = du/dt$, $\dot{v} = dv/dt$:

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

(We can use either the $\dot{u}$ equation or the $\dot{v}$ equation; the above uses the $\dot{u}$ equation.)