CHAPTER 36

Structure and Motion

Here is a foundational problem in computer vision. Assume there are N points in 3D; write $\mathbf{X}_i$ for the i'th such point. You obtain M images using perspective cameras. Not every point necessarily appears in every image, but you know which point appears in which image and where. Use $\delta_{i,j}$ to keep track, so

$$\delta_{i,j} = \begin{cases} 1 & \text{if the } i\text{'th point appears in the } j\text{'th image} \\ 0 & \text{otherwise} \end{cases}$$

and write $\mathbf{x}_{i,j}$ for the measurement you observe if the i'th point appears in the j'th image. Now recover the geometry of the points and of the cameras from this information.

Solutions to this problem can be separated into two major threads. In the offline case, you have all the data you need before you start computing. Imagine you want to check there aren't too many large patches of rust on a bridge. It is difficult and expensive to send out workers safely. Instead, fly a drone and take pictures. Now reconstruct the geometry of the bridge (the points) and the locations of the drone (the cameras). Render this geometry in 3D to inspect it. If the reconstruction is good enough, straightforward tools will allow you to measure the size of the rust patches *in the rendering*. Similarly, imagine you want to film a monster tearing up downtown Los Angeles. You can avoid problems with insurance companies by flying a camera over the downtown; reconstructing the geometry and the locations of the camera; and rendering the geometry with a CGI monster in place. Alternatively, you want to render a CGI monster into legacy footage of a city. Reconstruct the geometry and camera from the legacy footage, then ignore the geometry, and render the monster using the camera infomation you have recovered. The monster will be seen in the same camera as the background, and so it will look as if it were there.

In the online case, you must maintain the best estimate you can of camera and point geometry as information arrives. A moving agent – robot, drone, autonomous car – can use a solution to (a) build a map of their surroundings and (b) determine how it has moved with respect to the map. It turns out that, with care, online solutions can be very fast in very low-compute environments. For example, solutions to the online case have led to better-than-human performance in drone racing.

36.1 OUTLINE: POINTS AND CAMERAS FROM MULTIPLE PICTURES

This chapter will focus on the offline case. This case can be attacked with the tools of previous chapters, and reveals requirements to obtain solutions and ambiguities in solutions. The online case requires filtering methods (Chapter 21.3) and is dealt with in Chapter 21.3.

36.1.1 The Underlying Optimization Problem and Variants

You can obtain a solution in principle to the offline case in a familiar way. You (a) set up a large optimization problem and (b) obtain a start point. Setting up the optimization problem is simple and revealing. You must minimize the reprojection error of all points visible in all cameras, as a function of points and cameras. Write the camera matrix for the j'th camera $\mathcal{P}_j$. Write $L(\mathcal{P}_j, \mathbf{X}_i)$ for the predicted location of point i in camera j. This is measured in the coordinate system of image j. Then you could minimize reprojection error of all points visible in all cameras, which is

$$R(\mathcal{P}_1, \ldots, \mathcal{P}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N) = \sum_{ij} \delta_{ij} \left[(\mathbf{x}_{ij} - L(\mathcal{P}_i, \mathbf{X}_j))^T (\mathbf{x}_{ij} - L(\mathcal{P}_i, \mathbf{X}_j)) \right].$$

Solving this optimization problem at very large scales requires care and good start points. Procedures will occupy several sections (Section 21.3 for start points; Section 21.3 for the issues that arise at large scale).

Studying the problem without a solution is informative. The reprojection error as formulated above assumes that nothing is known about the cameras. Now assume each camera is calibrated, and write $\mathcal{K}_i$ for the intrinsics and $\mathcal{E}_i$ for the extrinsics of each camera. Then the problem becomes to minimize

$$R(\mathcal{E}_1, \ldots, \mathcal{E}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N) = \sum_{ij} \delta_{ij} \left[(\mathbf{x}_{ij} - L(\mathcal{K}_i\mathcal{E}_i, \mathbf{X}_j))^T (\mathbf{x}_{ij} - L(\mathcal{K}_i\mathcal{E}_i, \mathbf{X}_j)) \right].$$

subject to the constraints on each $\mathcal{E}_i$ required to ensure it is a Euclidean transformation. Now assume you do not know the calibration for each camera, but you do know that all cameras have the same calibration. Then the problem becomes to minimize

$$R(\mathcal{K}, \mathcal{E}_1, \ldots, \mathcal{E}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N) = \sum_{ij} \delta_{ij} \left[(\mathbf{x}_{ij} - L(\mathcal{K}\mathcal{E}_i, \mathbf{X}_j))^T (\mathbf{x}_{ij} - L(\mathcal{K}\mathcal{E}_i, \mathbf{X}_j)) \right].$$

36.1.2 Master Recipe for a Start Point

I start with a clean recipe for a straightforward case. Assume you see each point in every image and you have enough points. Assume you know the calibration of each camera. Then you can recover a start point for reprojection error in a straightforward way, in the box below. This recipe isn't practical in almost any real case, but it is quite easy to identify obstacles to success in real cases then fix them.

Procedure: 36.1 *Master recipe for offline structure from motion*

Assumptions: You have M images of N points. Each point appears in each image. You know the intrinsic matrix $\mathcal{K}_i$ for each camera. Reconstruct an estimate of the point positions in 3D and of each camera's extrinsics by:

- Choose two images.
- Construct the fundamental matrix connecting the two (Section 21.3).
- Since you know the calibration, turn this into an essential matrix (Section 21.3).
- Use the odometry of Section 21.3 to determine where the second camera is with respect to the first.
- Choose a scale for the translation, and then triangulate the points.
- Now use the procedure of Section 21.3 to recover the configuration of each other camera with respect to the set of points.

Use this estimate as a startpoint to minimize the reprojection error

$$R(\mathcal{K}, \mathcal{E}_1, \ldots, \mathcal{E}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N) = \sum_{ij} \delta_{ij} \left[(\mathbf{x}_{ij} - L(\mathcal{K}\mathcal{E}_i, \mathbf{X}_j))^T (\mathbf{x}_{ij} - L(\mathcal{K}\mathcal{E}_i, \mathbf{X}_j)) \right].$$

It is straightforward that this master recipe yields reconstruction up to rotation, translation and scale **exercises**. It is straightforward that it will work $M \geq 2$. It is slightly less straightforward, but true, that it will work for $N \geq 4$.

36.1.3 Inspecting the Master Recipe

The master recipe is an idealization that exposes what kinds of procedure are required. Inspecting it closely suggests some quite important points. The recipe implies some kind of ordering between views (choose the first two, then insert the rest), but this ordering isn't necessarily natural. A natural ordering can occur (for example, if the views come from a moving video camera) but it doesn't always occur. The recipe uses the first two views to determine the 3D configuration of each point, then just inserts cameras; but further views should improve the estimate of configuration.

You do not need to see every point in every view. It is enough to have at least two views of each point as long as points are shared between views in a way I will develop below. Imagine you do not see every point in every view. If you look at the master recipe, there are two steps that are affected by the number of points the cameras share. The first two cameras you choose will need to share enough points that you can compute a fundamental matrix. When you insert a camera, you will

need to see enough points with known 3D position to recover the configuration of a camera with respect to the reconstruction.

Once you have inserted a camera, you may be able to reconstruct more points. These will be points that you can see in the new camera, and that you have seen in some other camera, but only once. You now have two views of these points, and can triangulate them.

From this it follows that the number of points shared across cameras can be very important. You can build a graph where each node is a camera, and there is an edge between pairs of cameras that see points in common. Weight each edge by the number of points the cameras share. This graph is quite informative. You should start with a pair of cameras which has main pairs of points in common – equivalently, an edge with a high weight. Mark the two nodes you started with as used. You could then insert cameras and points by visiting nodes that are (a) unmarked and (b) connected to several marked nodes with edges with high weight. Equivalently, you are seeking to insert cameras where there are many points whose 3D configuration you know.

You can estimate the extrinsics of a camera with known calibration using as few as three reference points. Now remove every edge from the graph that has weight less than three. You will be able to reconstruct the points in each connected component of the resulting graph in some scaled Euclidean coordinate system specific to that connected component, but you will not be able to compute the transformation from one reconstruction to the other. Here is why. Start with a pair of cameras (edge) in some connected component. Continue to insert cameraThe master recipe reconstructs points and cameras with respect to the initial coordinate system,

but there are not enough points shared in any edge (pair of cameras) between the

If the resulting graph has k connected components, , you will not be able to produce a single reconstruction.

Any edge that has fewer than eight (or five, depending on the estimation procedure

Finally, there is some possibility of ambiguity in the reconstruction.

36.1.4 Internal Symmetries of the Reprojection Error

Imagine there are two different sets of cameras and points, $\{\mathcal{P}_1, \ldots, \mathcal{P}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N\}$ and $\{\mathcal{P}'_1, \ldots, \mathcal{P}'_M, \mathbf{X}'_1, \ldots, \mathbf{X}'_N\}$, so that $L(\mathcal{P}_j, \mathbf{X}_i) = L(\mathcal{P}'_j, \mathbf{X}'_i)$. Then the reprojection error of these two sets must be the same. If one set is a minimum of the reprojection error, the other will be too. This is an *internal symmetry*. You can now obtain information about possible ambiguities by assuming you have a solution, then asking what symmetries it has.

Assume $\{\mathcal{P}_1, \ldots, \mathcal{P}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N\}$ is a solution. Each camera must be a projective camera, and is at some location in space. Write each camera matrix in terms of its intrinsics $\mathcal{K}_j$, the canonical projection matrix $\mathcal{C}_p$, and its extrinsics

$$\mathcal{T}_j = \begin{bmatrix} \mathcal{E}_j & \mathbf{t}_j \\ \mathbf{0}^T & 1 \end{bmatrix}$$

to get

$$\mathcal{P}_j = \mathcal{K}_j \mathcal{C}_p \mathcal{T}_j = [\mathcal{K}_j \mathcal{E}_j \mid \mathcal{K}_j \mathbf{t}_j] .$$

Now apply a projective transformation $\mathcal{G}^{-1}$ to all the points $\mathbf{X}_i$ and apply $\mathcal{G}$ to all the cameras, yielding $\mathcal{P}'_j = \mathcal{P}_j \mathcal{G}$ and $\mathbf{X}'_i = \mathcal{G}^{-1}\mathbf{X}_i$. You should verify that $L(\mathcal{P}_j, \mathbf{X}_i) = L(\mathcal{P}'_j, \mathbf{X}'_i)$.(**exercises**).

Now look at the effect of $\mathcal{G}$ on your cameras. Write

$$\mathcal{G} = \begin{bmatrix} \mathcal{A} & \mathbf{b} \\ \mathbf{c} & d \end{bmatrix}$$

so

$$\mathcal{P}'_j = \left[\mathcal{K}_j \left(\mathcal{E}_j \mathcal{A} + \mathbf{t}_j \mathbf{c}^T\right) \mid \mathcal{K}_j \left(\mathcal{E}_i \mathbf{b} + s\mathbf{t}_j\right)\right] .$$

If you know nothing about the calibration of any camera, this information yields no constraint.

Remember this: *If you can reconstruct points and cameras from multiple views using perspective cameras where you have no camera calibration information, the reconstruction has a projective symmetry. If* $\{\mathcal{P}_1, \ldots, \mathcal{P}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N\}$ *is a solution, then*

$$\left\{\mathcal{P}_1 \mathcal{G}, \ldots, \mathcal{P}_M \mathcal{G}, \mathcal{G}^{-1}\mathbf{X}_1, \ldots, \mathcal{G}^{-1}\mathbf{X}_N\right\}$$

is also a solution for $\mathcal{G}$ *any invertible* 4×4 *matrix.*

Now assume you know the calibration of each camera. This constrains the possible $\mathcal{P}'_j$ in an important way. Write $\mathrm{LT}(\mathcal{M})$ for the lower triangular matrix you get when you decompose $\mathcal{M}$ into a lower triangular matrix and a rotation. If you decompose the left 3×3 block of $\mathcal{P}'_j$ like this, the lower triangular matrix is the camera intrinsics. These are known, so you must have

$$\mathrm{LT}(\mathcal{K}_j \left(\mathcal{E}_j \mathcal{A} + \mathbf{t}_j \mathbf{c}^T\right)) = \mathcal{K}_j .$$

(**exercises**). Check that this means

$$\left(\mathcal{E}_j \mathcal{A} + \mathbf{t}_j \mathbf{c}^T\right)^T \left(\mathcal{E}_j \mathcal{A} + \mathbf{t}_j \mathbf{c}^T\right)$$

is the identity (**exercises**). Assuming there are many different cameras, and their positions aren't special, $\mathbf{c}$ must be $\mathbf{0}$ (**exercises**). This means that $\mathcal{A}^T\mathcal{A}$ is the identity and so $\mathcal{A}$ is a rotation matrix. Notice that d is not constrained, which means the ambiguity is a scaled Euclidean transformation.

Remember this: *If you can reconstruct points and cameras from multiple views using perspective cameras where you know the intrinsics of each camera, the reconstruction has a scaled Euclidean symmetry. If* $\{\mathcal{P}_1, \ldots, \mathcal{P}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N\}$ *is a solution, then*

$$\left\{\mathcal{P}_1\mathcal{G}, \ldots, \mathcal{P}_M\mathcal{G}, \mathcal{G}^{-1}\mathbf{X}_1, \ldots, \mathcal{G}^{-1}\mathbf{X}_N\right\}$$

is also a solution for $\mathcal{G}$ *a scaled Euclidean transformation.*

Obtaining a start point is straightforward *in principle*.

36.1.5 Requirements and Ambiguities

36.2 LARGE SCALE PROCEDURES

36.3 APPLICATION: VISUALIZING CITIES USING SFM

36.4 APPLICATION: CONSTRUCTION MONITORING USING SFM