CHAPTER 37

Structure From Motion: Geometric Concepts

Here is a foundational problem in computer vision, usually referred to as *structure from motion* or, almost always, *SFM*. You see many images of a scene. You must reconstruct the scene and the camera locations from this, and perhaps other, 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. For example, you might have a video of a stationary scene obtained with moving cameras. In the online case, you must maintain the best estimate you can of camera and point geometry as information arrives. A moving agent that can solve this problem online can build estimates of the geometry of the world and of how it has moved through the world. This is very useful to, for example, autonomous drones.

This chapter will focus on the offline case, in the context of a useful abstraction. Assume the scene consists of 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. Recover the geometry of the points and of the cameras from this information. This version of structure from motion is best seen as solving a very large and quite delicate optimization problem – minimize the reprojection error of the points by adjusting the geometry of points and cameras. Studying the optimization problem in abstract reveals important differences between cases where you know camera calibration and cases where you don't.

Just throwing the optimization problem into an optimizer doesn't work. A very good start point can constructed with the tools of previous chapters. Attacking the online case requires a very different set of tools (filtering methods, Chapter 21.3) and is dealt with in Chapter 21.3.

37.1 OUTLINE: POINTS AND CAMERAS FROM MULTIPLE PICTURES

The structure from motion problem in the form above is an abstraction for reconstructing the geometry of a scene from pictures. It is an abstraction, because the geometry might consist of much more than just scattered points. Bear with this abstraction for a while (Chapter 21.3 shows how to proceed to much more interesting geometries). Solutions to the underlying problem are hugely useful.

For example, 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* (Figure **??**).

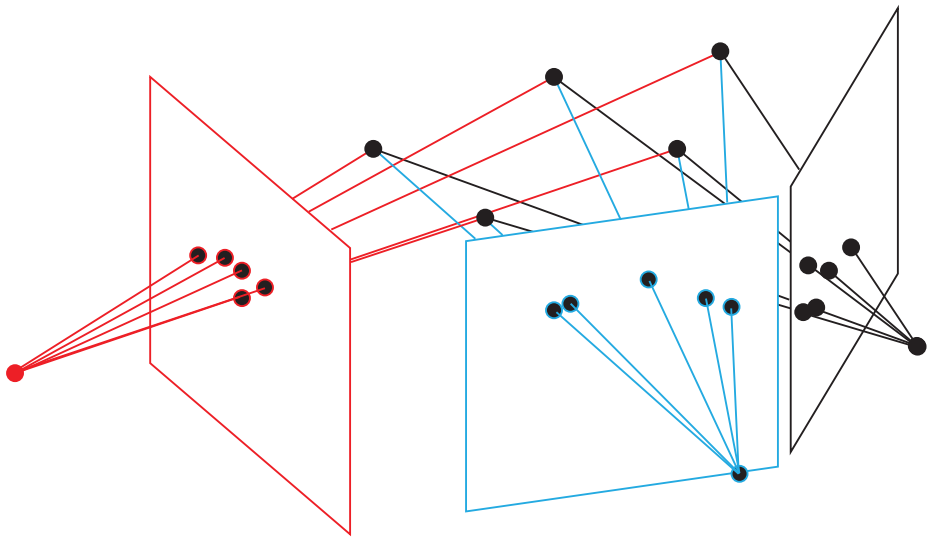

FIGURE 37.1: *In the abstract structure from motion problem, M (here 3) perspective cameras view N (5) points in 3D. You have the projections of the points, and must recover what you can of the geometry of the points and the relative configuration of the cameras. Small changes in the formulation of the problem – for example, what you know about camera calibration; which points can be seen in which cameras; and so on – can have quite drastic effects on what solutions you can obtain.*

Similarly, imagine you want to film a monster tearing up downtown Los Angeles. You can avoid problems with zoology and 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. The monster will be seen in the same camera as the background, and so it will look as if it were there. You can use this idea to, for example, place monsters in legacy footage and so on.

37.1.1 Elementary SFM

I start with a simple procedure for a straightforward case. Assume you see each point in every image and you have enough points. Assume you know the intrinsic matrix for two of these cameras. Then you can recover a start point for reprojection error in a straightforward way.

Procedure: 37.1 *Elementary SFM*

You have M images of N points. Each point appears in each image. You know the intrinsic matrix for two cameras.

- Take the two images from the cameras where you know the intrinsic matrix, construct the fundamental matrix connecting the two (Section 21.3). Since you know the intrinsics, 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.
- Finally, use the procedure of Section 21.3 to recover the configuration of each other camera with respect to the set of points. Notice this yields extrinsics *and* intrinsics.

Assume you have no numerical errors, etc. and each step works exactly. It is straightforward that this procedure yields reconstruction up to rotation, translation and scale **exercises**. It is straightforward that it will work for $M \geq 2$. It is rather less straightforward, but true, that it will work for $N \geq 4$ (you would need to be quite sophisticated about getting the essential matrix, in ways I have not described **exercises**).

Remember this: *Scale Ambiguity in Structure from Motion*

37.1.2 When You Don't Know Intrinsics

The elementary recipe of Section 21.3 assumed you knew the intrinsics for two cameras. If you know no intrinsics, this recipe will still work. Simply guess the intrinsics for the first two camera. For example, choose a 3×3 upper triangular matrix of full rank at random to be the intrinsic matrix for each camera. The recipe will still produce the solutions for camera matrices and point locations. However, the relationship between the results and the original geometry of the points may not be what you think (Section 21.3).

37.1.3 When Some Points are Missing in Some Images

You can extend this recipe to deal with cases where points do not appear in every image. This is easy if there is a pair of images that contains every point. You find this pair, then triangulate all the points as above to get a reconstruction. Now calibrate every other camera using the points you can see in that camera. It should

be obvious that if any camera views too few points, you won't be able to calibrate it **exercises**.

A more interesting extension occurs when there is no pair of images where every point appears in both. Choose a pair of images which have enough points in common for you to estimate the essential matrix (8, 7 or 4 depending on just how sophisticated you feel like being). Recover the two cameras and the common points as above. Start pools of working images, working cameras, and working points with the images, cameras and points.

Now you repeat the following steps. Choose some image which isn't in the working pool and has enough points in common with the points in the working pool. Calibrate the camera for that image using the common points, and add the image and the camera to the relevant working pools. You may now be able to expand the pool of working points. If the chosen camera has points in common with any camera in the working pool, *where you don't know the 3D configuration of the point*, you can reconstruct those points by triangulation. Do so, add these points to the working pool of points, and continue. Stop when you can't add any camera.

When you stop, there might be images that you can't add to the working pool. Imagine, for example, you two sets of images. In the first set, you have M_1 images of N_1 points, and in the second, you have M_2 images of N_2 completely different points. Start with two images in the first set. You will not be able to add any image in the second set, because you can't use points in the first set to calibrate cameras in the second set. This issue becomes interesting and important for large scale reconstruction, but I won't resolve it here (Section 21.3).

37.1.4 The Underlying Optimization Problem and Variants

The recipes above illustrate why you should be able to compute structure from motion, but they should look quite suspiciously impractical to you. The problem is the assumption that triangulation and calibration always give the right answer. If they don't give exactly the right answer, you will get different reconstructions when you choose images in different orders. For example, start with a pair of images with a very small baseline; the triangulation is probably bad, meaning your next calibration is dubious, and the whole exercise becomes unsatisfactory. But if, for the same dataset, your first pair of images has a large baseline, you might get quite a good reconstrution. There are two ways around this class of problem. One is you improve the reconstruction with an optimization problem (here). The other is you think quite carefully about what images you use at each stage (Section 21.3).

You can formalize structure from motion as minimizing 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, in image coordinates. Use $\delta_{i,j}$ to keep track of which point is visible in which camera, 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 write $\mathbf{r}_{i,j}$ for the reprojection error for point i in camera j, when this

is meaningful – that is, when point i is visible in image j. In this case,

$$\mathbf{r}_{ij} = (\mathbf{x}_{i,j} - L(\mathcal{P}_j, \mathbf{X}_i)).$$

It will not matter what value $\mathbf{r}_{i,j}$ has when you can't see point i in camera j. Then you could minimize reprojection error of all points visible in all cameras, which is

$$R(\mathcal{P}_1, \ldots, \mathcal{P}_M, \mathbf{X}_1, \ldots, \mathbf{X}_N) = \sum_{i,j} \delta_{i,j} \mathbf{r}_{i,j}^T \mathbf{r}_{i,j}$$

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

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

37.1.5 Uncalibrated SFM has a Projective Symmetry

Recall that you can reconstruct points and cameras without knowing intrinsics. Just use a random intrinsic matrix whenever you need one – it needs to be 3×3, upper triangular, and of full rank. Assume that all computations in the procedure were exact, and all steps worked perfectly. Then the result should have zero reprojection error and must be a solution to the optimization problem. You should suspect that you will get different reconstructions when you use different intrinsic matrices. This is the case, and it means the optimization problem has more than one solution – there are *ambiguities*.

Thinking about the multiple solutions to the optimization problem will reveal the relationship between what you reconstruct with random intrinsic matrices and the true solution. 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 that 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

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

Now apply a projective transformation $\mathcal{G}^{-1}$ to all the points $\mathbf{X}_i$ and apply $\mathcal{G}$ to all the cameras. You have $\mathcal{P}'_j = \mathcal{P}_j \mathcal{G}$ 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**). If you know nothing about the calibration of your cameras, you cannot tell whether $\mathcal{P}_j$ and $\mathcal{P}'_j$ is the right camera reconstruction. This means you cannot tell which reconstruction to choose – $\mathbf{X}_i$ or $\mathbf{X}'_i$. This yields an important result **exercises**.

Remember this: *If you can reconstruct points and cameras from multiple views using perspective cameras where you have no camera intrinsic 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.*

As I shall show, there are many circumstances where, even though you don't know camera intrinsics explicitly, you can recover them (Section 21.3). This means that it can take care to be sure that "you have no camera intrinsic information".

37.1.6 Fully Calibrated SFM has a Similarity Ambiguity

Now assume you know the j'th camera has intrinsic matrix $\mathcal{K}_i$. To reason about the symmetry, assume you have a reconstruction where the j'th camera is

$$\mathcal{P}_j = \mathcal{K}_j \left[\mathcal{E}_j \mid \mathbf{t}_j \right].$$

If you know the intrinsics of your cameras, you can tell whether $\mathcal{P}'_j$ is unacceptable. The only $\mathcal{P}'_j$ that are ambiguous are reconstructions that have the right intrinsics. This means the symmetry must leave the intrinsics of each camera unchanged. So $\mathcal{P}_j$ and $\mathcal{P}'_j = \mathcal{P}_j \mathcal{G}$ must have the same intrinsics matrix for every j. Write

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

so

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

Write $\mathcal{M}_j$ for the left 3×3 block of $\mathcal{P}'_j$. You can uniquely decompose $\mathcal{M}_j$ into a lower triangular matrix (write $LT(\mathcal{M}_j)$) times a rotation matrix. Now $LT(\mathcal{M}_j)$ must be $\mathcal{K}_j$, so

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

must be a rotation matrix for *any* choice of $\mathbf{t}_j$. 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 **exercises** and so $\mathcal{A}$ is a rotation matrix. Notice that d is not constrained, which means the ambiguity is a scaled Euclidean transformation **exercises**.

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.*

37.2 SELF CALIBRATION

37.2.1 Self Calibration from Vanishing Points

There are a variety of cases where you can actually obtain a calibration of a camera *from a picture*, assuming that you have some other information. Here is a simple example. Assume you know the vanishing points of directions in the image that are orthonormal. For example, Figure **??** shows a perspective image of a cube. Conveniently, you can construct the three vanishing points. If you *know* the directions are at right angles (they don't have to be to get this picture), you can compute some camera calibration information.

There are three vanishing points. Write $\mathbf{v}_i$ for the i'th vanishing point *in homogenous coordinates in the image plane*. This means there are three components in $\mathbf{v}_i$. You know them up to scale. So if the vanishing point was at location $(v_{i1}, v_{i2})^T$ in the image – which you can measure – you could write

$$\mathbf{v}_i = \lambda_i \begin{bmatrix} v_{i1} \\ v_{i2} \\ 1 \end{bmatrix}$$

where λ_i is unknown. Now assume that each vanishing point is the vanishing point of a coordinate direction. So $\mathbf{v}_1$ is the point at infinity along the X-axis, and so on. Write $\mathbf{E}_i$ for these three points *in homogenous coordinates in 3D*. You will have

$$\mathbf{E}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } \mathbf{E}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ and } \mathbf{E}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Write $\mathcal{C} = \mathcal{K}[\mathcal{R} \mid \mathbf{t}]$ for the (completely unknown) camera. Recall that $\equiv$ means "refers to the same point in homogeneous coordinates", so $\mathbf{a} \equiv \mathbf{b}$ means there is some $\lambda \neq 0$ so that $\lambda\mathbf{a} = \mathbf{b}$. Now

$$\mathbf{v}_i \equiv \mathcal{C}\mathbf{E}_i$$

but the 0 in $\mathbf{E}_i$'s fourth component means you can ignore the last column in the camera. So write

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ and } \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

and find

$$\mathbf{e}_i \equiv \mathcal{R}^{-1}\mathcal{K}^{-1}\mathbf{v}_i.$$

Now $\mathbf{e}_i^T\mathbf{e}_j = 0$ if $i \neq j$ (they're orthonormal). This means you have three independent constraints on $\mathcal{K}$.

$$\mathbf{v}_1^T\mathcal{K}^{-T}\mathcal{K}^{-1}\mathbf{v}_2 = 0 \text{ and } \mathbf{v}_1^T\mathcal{K}^{-T}\mathcal{K}^{-1}\mathbf{v}_3 = 0 \text{ and } \mathbf{v}_2^T\mathcal{K}^{-T}\mathcal{K}^{-1}\mathbf{v}_3 = 0.$$

Furthermore, because these constraints are equal to zero, the question of scaling the homogeneous coordinates doesn't matter **exercises**. It turns out that, if you assume there pixels are square and there is no skew, you can recover scale and principal point like this **exercises**.

37.2.2 Self Calibration in SFM

Now imagine you know that all cameras have the same calibration matrix $\mathcal{K}$, which you do not know. If you have enough images and enough points, the following argument shows you can recover $\mathcal{K}$. First, obtain some reconstruction, yielding cameras $\mathcal{Q}_j$ and points $\mathbf{Y}_i$. Now think about the camera for $j = 1$ (you can reindex the cameras if you don't like using the original first camera **exercises**). Write $[\mathcal{U}_1 \mid \mathbf{u}_1]$ for that camera. Now choose

$$\mathcal{G} = \begin{bmatrix} \mathcal{U}_1^{-1} & -\mathcal{U}_1\mathbf{u}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}$$

and check that $\mathcal{Q}_1\mathcal{G} = [\mathcal{I} \mid \mathbf{0}]$. This represents a canonical camera. Since you know $\mathcal{G}$, you can transform your reconstruction to form $\mathcal{P}_j = \mathcal{Q}_j\mathcal{G}$, $\mathbf{X}_i = \mathcal{G}^{-1}\mathbf{Y}_i$.

Now any transformation of the form

$$\mathcal{H} = \begin{bmatrix} \mathcal{K} & \mathbf{0} \\ -\mathbf{p}^T\mathcal{K} & 1 \end{bmatrix}$$

will produce $\mathcal{C}_1 = \mathcal{P}_1\mathcal{H} = [\mathcal{K} \mid \mathbf{0}] = \mathcal{K}[\mathcal{I} \mid \mathbf{0}]$, which is a canonical camera with the right intrinsics. Here $\mathcal{K}$ and $\mathbf{p}$ are unknown, but can be constrained.

You know $\mathcal{P}_j$, which I will write $[\mathcal{O}_j \mid \mathbf{o}_j]$ to emphasize you know the entries. Consider $\mathcal{C}_j = \mathcal{P}_j\mathcal{H} = \left[(\mathcal{O}_j - \mathbf{o}_j\mathbf{p}^T)\mathcal{K} \mid \mathbf{o}_j\right]$; this camera must have intrinsics $\mathcal{K}$, which means that for some unknown rotation $\mathcal{R}_j$, you get the equation

$$(\mathcal{O}_j - \mathbf{o}_j\mathbf{p}^T)\mathcal{K} = \mathcal{K}\mathcal{R}_j$$

and with some manipulation **exercises**, you get

$$(\mathcal{O}_j - \mathbf{o}_j\mathbf{p}^T)\mathcal{K}\mathcal{K}^T(\mathcal{O}_j - \mathbf{o}_j\mathbf{p}^T)^T = \mathcal{K}\mathcal{K}^T.$$

This is a set of non-linear equations in the entries of $\mathcal{K}$, whose coefficients are known. There are at most five unknowns in $\mathcal{K}$, and there are three in $\mathbf{p}$, for a total of five unknowns. If you have M cameras, you have $5(M-1)$ equations in these unknowns **exercises**. It turns out there are no inconvenient redundancies or dependencies in the equations, and they can be solved **exercises**. As a result, if you have at least three images where the cameras have the same intrinsics, you can recover the intrinsics. In turn, because you know the intrinsics, you can recover geometry up to scaled Euclidean transformations.