Course Notes 4: Reconstruct 3D Geometry[*]

1 Introduction

So far, we have covered camera models (i.e., image formation), camera calibration (i.e., recovering camera intrinsic and extrinsic parameters), and how adding additional viewpoints of a scene can greatly enhance our knowledge of the said scene. In the previous notes, we focused on epipolar geometry to relate points of one image plane to points in the other without knowing any information about the 3D scene. We also discussed how to compute the fundamental matrix from point correspondences. In this lecture note, we will discuss how to recover information about the 3D scene from multiple 2D images. We mainly use two images to explain the methodology.

2 Camera Matrices from the Fundamental Matrix

We can extract an accurate, initial estimate of camera matrices by using the essential matrix, which is a special case of the fundamental matrix for normalized coordinates. Recall that, by using the essential matrix E, we assume that we have calibrated the camera and thus know the intrinsic camera matrix K[1]. We can either compute the essential matrix E either from the normalized image coordinates directly or from its relationship with the fundamental matrix F and intrinsic matrix K:

$$E = K^T F K \tag{2.1}$$

Because the essential matrix assumes that we have calibrated cameras, we should remember that it only has five degrees of freedom, as it only encodes the extrinsic parameters: the rotation R and translation $\mathbf{t}$ between the cameras. Luckily, this is exactly the information that we want to extract to create our motion matrix. First, recall that the essential matrix E can be represented as

$$E = [\mathbf{t}_\times] R \tag{2.2}$$

As such, perhaps we can find a strategy to factor E into its two components. First, we should notice that the cross-product matrix $[\mathbf{t}_\times]$ is skew-symmetric. We define two

[*]The contents are from
- D. A. Forsyth and J. Ponce. Computer Vision: A Modern Approach (2nd Edition).
- R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision (2nd Edition)
- K. Hata and S. Savarese. Course notes of Stanford CS231A

[1]We assume the two cameras have the same intrinsic parameters, which is quite common in practice.

matrices that we will use in the decomposition:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.3}$$

One important property we will use later is that $Z = \text{diag}(1, 1, 0)W$ is up to a sign. Similarly, we will also use the fact that $ZW = ZW^T = \text{diag}(1, 1, 0)$ is up to a sign.

As a result of eigenvalue decomposition, we can create a block decomposition of a general skew-symmetric matrix known up to scale. Thus, we can write $[\mathbf{t}_\times]$ as

$$[\mathbf{t}_\times] = UZU^T \tag{2.4}$$

where U is some orthogonal matrix. Therefore, we can rewrite the decomposition as:

$$E = U\text{diag}(1, 1, 0)(WU^T R) \tag{2.5}$$

Looking at this expression carefully, we see that it closely resembles the singular value decomposition $E = UDV^T$, where D contains two equal singular values. If we know E up to scale and we assume that it takes the form $E = U\text{diag}(1, 1, 0)V^T$, then we arrive at the following factorizations of E:

$$[\mathbf{t}_\times] = UZU^T, \quad R = UWV^T \text{ or } UW^TV^T \tag{2.6}$$

We can prove that the given factorizations are valid by inspection. We can also prove that there are no other factorizations. The form of $[\mathbf{t}_\times]$ is determined by the fact that its left null space must be the same as the null space of E. Given unitary/orthogonal matrices U and V, any rotation R can be decomposed into UXV^T where X is another rotation matrix. After substituting these values, we get $ZX = \text{diag}(1, 1, 0)$ up to scale. Thus, X must be equal to W or W^T.

Note that this factorization of E only guarantees that the matrices UWV^T and UW^TV^T are orthogonal. To ensure that R is a valid rotation, we simply make sure that the determinant of R is positive:

$$R = (\det UWV^T)UWV^T \text{ or } (\det UW^TV^T)UW^TV^T \tag{2.7}$$

Similar to how the rotation R can take on two potential values, the translation vector $\mathbf{t}$ can also take on several values. From the definition of cross product, we know that

$$\mathbf{t} \times \mathbf{t} = [\mathbf{t}_\times]\mathbf{t} = UZU^T\mathbf{t} = 0 \tag{2.8}$$

Knowing that U is unitary, we can find that $\|[\mathbf{t}_\times]\|F = \sqrt{2}$. Therefore, our estimate of $\mathbf{t}$ from this factorization will come from the above equation and the fact that E is known up to scale. This means that

$$\mathbf{t} = \pm U \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \pm\mathbf{u}_3 \tag{2.9}$$

where $\mathbf{u}_3$ is the third column of U. By inspection, we can also verify that we get the same results by reformatting $[\mathbf{t}_\times] = UZU^T$ into the vector $\mathbf{t}$ known up to a sign.

As illustrated in Figure 1, there are four potential $R, \mathbf{t}$ pairings since there exist two options for both R and $\mathbf{t}$. Intuitively, the four pairings include all possible pairings of

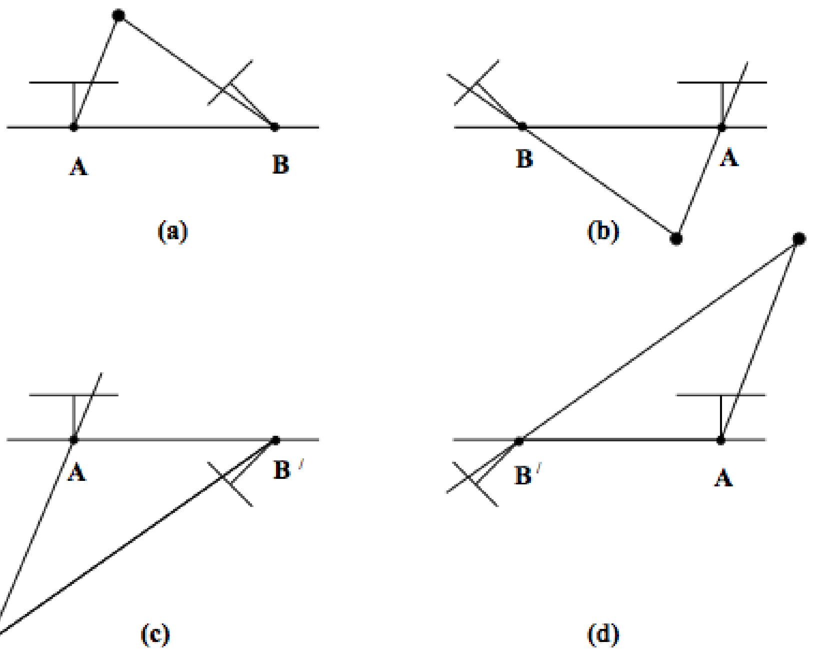

Figure 1: There are four possible solutions for extracting the relative camera rotation R and translation $\mathbf{t}$ from the essential matrix. However, only in (a) is the reconstructed point in front of both of the cameras. (Figure taken from Hartley and Zisserman textbook page 260)

rotating a camera in a certain direction or rotating the camera in the opposite direction combined with the option of translating it in a certain direction or the opposite direction. Therefore, under ideal conditions, we would only need to triangulate one point to determine the correct $R, \mathbf{t}$ pair. For the correct $R, \mathbf{t}$ pair, the triangulated point $\hat{\mathbf{P}}$ exists in front of both cameras, which means that it has a positive z-coordinate with respect to both camera reference systems. Due to measurement noise, we often do not rely on triangulating only one point, but will instead triangulate many points and determine the correct $R, \mathbf{t}$ pair as the one that contains the most of these points in front of both cameras.

3 Triangulation

One of the most fundamental problems in multiple view geometry is the problem of *triangulation*, the process of determining the location of a 3D point given its projections into two or more images.

In the triangulation problem with two views, we have two cameras with known camera intrinsic parameters K and K' respectively. We also know the relative orientations and offsets $R, \mathbf{t}$ of these cameras with respect to each other. Suppose that we have a point $\mathbf{P}$ in 3D, which can be found in the images of the two cameras at $\mathbf{p}$ and $\mathbf{p}'$ respectively. Although the location of $\mathbf{P}$ is currently unknown, we can measure the exact locations of $\mathbf{p}$ and $\mathbf{p}'$ in the image. Because K, K', R, and $\mathbf{t}$ are known, we can compute the two lines of sight $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$, which are defined by the camera centers $\mathbf{O_1}$, $\mathbf{O_2}$ and the image locations $\mathbf{p}$, $\mathbf{p}'$. Therefore, $\mathbf{P}$ can be computed as the intersection of $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$.

Although this process appears both straightforward and mathematically sound, it

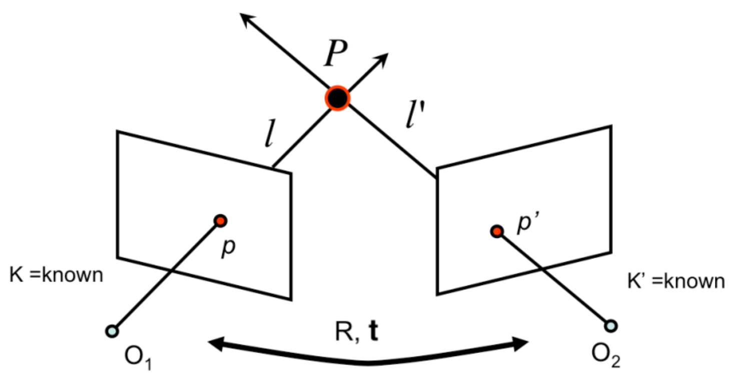

Figure 2: The setup of the triangulation problem when given two views.

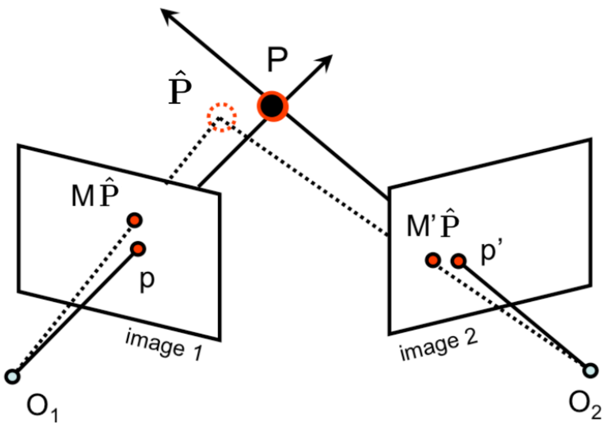

Figure 3: The triangulation problem in real-world scenarios often involves minimizing the reprojection error.

does not work very well in practice. In the real world, because the observations $\mathbf{p}$ and $\mathbf{p}'$ are noisy and the camera calibration parameters are not precise, finding the intersection point of $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$ may be problematic. In most cases, it will not exist at all, as the two lines may never intersect.

A few variant methods are available for finding the optimal location of a 3D point given its projections in two views. For the discussion of some of these methods, please refer to Hartley and Strum (1997)[2]. In the following, we will discuss two commonly used methods: the linear method and the non-linear method.

3.1 A linear method for triangulation

In this section, we describe a simple linear triangulation method that solves the lack of an intersection point between rays. We are given two points in the images that correspond to each other $\mathbf{p} = M\mathbf{P} = (x, y, 1)$ and $\mathbf{p}' = M'\mathbf{P} = (x', y', 1)$. By the definition of the cross product, $\mathbf{p} \times (M\mathbf{P}) = 0$. We can explicitly use the equalities generated by the cross product to form three constraints:

$$\begin{aligned} x(\mathbf{m}_3^T\mathbf{P}) - (\mathbf{m}_1^T\mathbf{P}) &= 0 \\ y(\mathbf{m}_3^T\mathbf{P}) - (\mathbf{m}_2^T\mathbf{P}) &= 0 \\ x(\mathbf{m}_2^T\mathbf{P}) - y(\mathbf{m}_1^T\mathbf{P}) &= 0 \end{aligned} \tag{3.1}$$

[2]R. Hartley and P. Strum. Triangulation. Computer vision and image understanding. 1997.

where $\mathbf{m}_i^T$ denotes the i-th row of M. Similar constraints can be formulated for $\mathbf{p}'$ and M'. Using the constraints from both images, we can formulate a linear equation of the form $A\mathbf{P} = 0$ where

$$A = \begin{bmatrix} x\mathbf{m}_3^T - \mathbf{m}_1^T \\ y\mathbf{m}_3^T - \mathbf{m}_2^T \\ x'\mathbf{m}_3'^T - \mathbf{m}_1'^T \\ y'\mathbf{m}_3'^T - \mathbf{m}_2'^T \end{bmatrix} \tag{3.2}$$

This equation can be solved using SVD to find the best linear estimate of the point $\mathbf{P}$. Another interesting aspect of this method is that it can handle triangulating from multiple views as well. To do so, one simply appends additional rows to A corresponding to the added constraints by the new views.

3.2 A nonlinear method for triangulation

Compared with the linear method, the triangulation problem for real-world scenarios is often mathematically characterized as solving a minimization problem:

$$\min_{\hat{\mathbf{P}}} \|M\hat{\mathbf{P}} - \mathbf{p}\|^2 + \|M'\hat{\mathbf{P}} - \mathbf{p}'\|^2 \tag{3.3}$$

In the above equation, we seek to find a $\hat{\mathbf{P}}$ in 3D that best approximates $\mathbf{P}$ by finding the best least-squares estimate of the *reprojection error* of $\hat{\mathbf{P}}$ in both images. The reprojection error for a 3D point in an image is the distance between the projection of that point in the image and the corresponding observed point in the image plane. In the case of our example in Figure 3, since M is the projective transformation from 3D space to image 1, the projected point of $\hat{\mathbf{P}}$ in image 1 is $M\hat{\mathbf{P}}$. The matching observation of $\hat{\mathbf{P}}$ in image 1 is $\mathbf{p}$. Thus, the reprojection error for point $\mathbf{P}$ in image 1 is the distance $\|M\hat{\mathbf{P}} - \mathbf{p}\|$. The overall reprojection error found in Equation 3.3 is the sum of the reprojection errors across all the points in the image. For cases with more than two images, we would simply add more distance terms to the objective function.

$$\min_{\hat{\mathbf{P}}} \sum_i \|M\hat{\mathbf{P}}_i - \mathbf{p}_i\|^2 \tag{3.4}$$

In practice, there exists a variety of very sophisticated optimization techniques that result in good solutions to the problem[3].

4 Structure from motion

At the end of Section 3.1, we hinted how we can go beyond two views of a scene to gain information about the 3D scene. We will now explore the extension of the geometry of two cameras to multiple cameras. By combining observations of points from multiple views, we will be able to simultaneously determine both the 3D structure/geometry of the scene and the parameters of the camera in what is known as *structure from motion*.

Here, we formally introduce the *structure from motion* problem. Suppose we have m cameras with camera transformations M_i encoding both the intrinsic and extrinsic parameters for the cameras. Let $\mathbf{X}_j$ be one of the n 3D points in the scene. Each 3D point may be visible in multiple cameras at the location $\mathbf{x}_{ij}$, which is the projection of

[3] `https://en.wikipedia.org/wiki/Non-linear_least_squares`

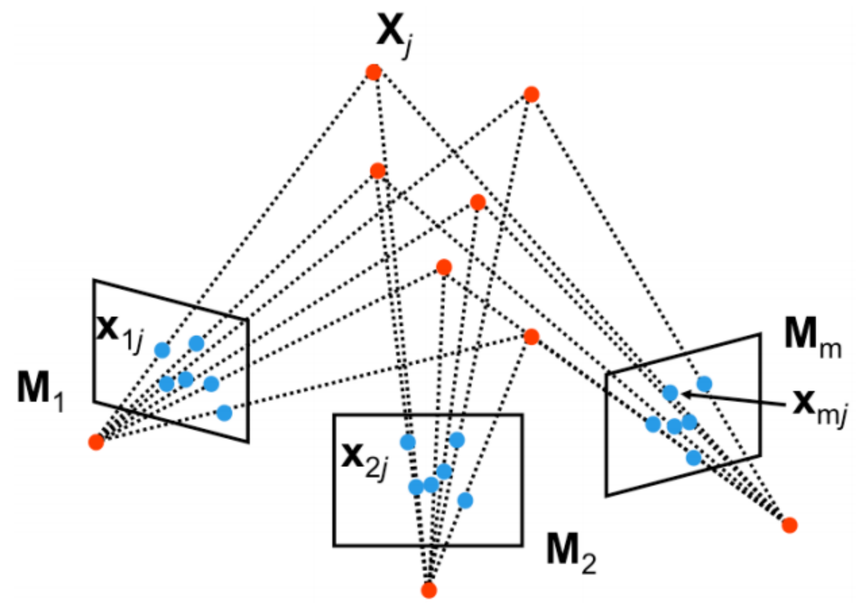

Figure 4: The setup of the general structure from motion problem.

$\mathbf{X}_j$ to the image of the camera i using the projective transformation M_i. Structure from motion is to recover both the structure of the scene (the n 3D points $\mathbf{X}_j$) and the motion of the cameras (the m projection matrices M_i) from all the observations $\mathbf{x}_{ij}$.

In the general case with projective cameras, each camera matrix M_i contains 11 degrees of freedom, as it is defined up to scale:

$$M_i = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & 1 \end{bmatrix} \tag{4.1}$$

We can set up the general structure from motion problem as estimating both the m motion matrices M_i and n 3D points $\mathbf{X}_j$ from mn observations $\mathbf{x}_{ij}$. Because cameras and points can only be recovered up to a 4×4 projective transformation up to scale (15 parameters), we have $11m + 3n - 15$ unknowns in $2mn$ equations. From these facts, we can determine the number of views and observations that are required to solve for the unknowns.

4.1 An example structure from motion pipeline

After extracting the camera matrices from the essential matrix, the estimates of the 3D scenes can be known up to scale. The 3D points can be computed from the estimated camera matrices via the triangulation methods described earlier.

The extension to the multi-view case can be done by chaining pairwise cameras. We can use the essential matrix or the algebraic approach (see Appendix B) to obtain solutions for the camera matrices and the 3D points for any pair of cameras, provided that there are enough point correspondences. The reconstructed 3D points are associated with the point correspondences available between the camera pair. Those pairwise solutions may be combined and optimized together in an approach called bundle adjustment as we will see next.

4.2 Bundle adjustment

Bundle adjustment is a nonlinear method for solving the structure from motion problem. In the optimization, we aim to minimize the reprojection error, which is the pixel distance between the projection of a reconstructed point into the estimated cameras and its corresponding observations for all the cameras and all the points. Previously, when discussing nonlinear optimization methods for triangulation, we focused primarily on the two-camera case, in which we naturally assumed that each camera saw all the correspondences between the two. However, since bundle adjustment handles several cameras, it only calculates the reprojection error for only the observations that can be seen by each camera. Ultimately though, this optimization problem is very similar to the one we introduced when talking about nonlinear methods for triangulation.

Two common approaches for solving bundle adjustment's nonlinear optimization include the Gauss-Newton algorithm and the Levenberg-Marquardt algorithm. You can refer to Appendix A for details on the Gauss-Newton algorithm and refer to the Hartley and Zisserman textbook for more details on the Levenberg-Marquardt algorithm.

In conclusion, bundle adjustment has some important advantages and limitations when compared with some other methods (e.g., the algebraic approach described in Appendix B). It is particularly useful because it can handle a large number of views smoothly and also handle cases when particular points are not observable by every image. However, the main limitation is that it is a particularly large minimization problem, as the parameters grow with the number of views. Additionally, it requires a good initial condition since it relies on nonlinear optimization techniques. For this reason, bundle adjustment is often used as the final step of most structure from motion implementations (e.g., after the algebraic approach), as the algebraic approach may provide a good initial solution for the optimization problem.

4.3 Reconstruction ambiguity

In this section, we discuss the inherent ambiguities involved in the reconstruction of a scene from point correspondences. This topic will be discussed in a general context, without reference to a specific method of carrying out the reconstruction.

Without some knowledge of a scene's placement with respect to a 3D coordinate frame, it is generally not possible to reconstruct the absolute position or orientation of a scene from a pair of views (or in fact from any number of views). This is true independently of any knowledge which may be available about the internal parameters of the cameras, or their relative placement. For instance, the exact latitude and longitude of a house (or any scene) cannot be computed, nor is it possible to determine whether its corridor runs north-south or east-west. This may be expressed by saying that the scene is determined at best up to a Euclidean transformation (rotation and translation) with respect to the world frame.

Only slightly less obvious is the fact that the overall scale of the scene cannot be determined. For example, it is impossible based on the images alone to determine the height of a building. It may be 4 meters, 3 meters ... It is even possible that this is an image of a doll's house and the corridor is 10 cm wide. Our common experience leads us to expect that ceilings are approximately 3m from the floor, which allows us to perceive the real scale of the scene. This extra information is an example of subsidiary knowledge of the scene not derived from image measurements. Without such knowledge therefore the scene is determined by the image only up to a similarity transformation (rotation, translation, and scaling).

Appendix A Gauss-Newton method for triangulation

The general nonlinear least-squares problem is to find an $\mathbf{x} \in \mathbb{R}^n$ that minimizes

$$\|\mathbf{r}(\mathbf{x})\|^2 = \sum_{i=1}^{m} r_i(\mathbf{x})^2 \tag{A.1}$$

where $\mathbf{r}$ is any residual function $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$ such that $\mathbf{r}(\mathbf{x}) = f(\mathbf{x}) - \mathbf{y}$ for some function f, input $\mathbf{x}$, and observation $\mathbf{y}$. The nonlinear least-squares problem reduces to the regular, linear least-squares problem when the function f is linear. However, recall that, in general, our camera matrices are not affine. Because the projection into the image plane often involves a division by the homogeneous coordinate, the projection into the image is generally nonlinear.

Notice that if we set $\mathbf{e}_i$ to be a 2×1 vector $\mathbf{e}_i = M\hat{\mathbf{P}}_i - \mathbf{p}_i$, then we can reformulate our optimization problem to be:

$$\min_{\hat{\mathbf{P}}} \sum_i \|\mathbf{e}_i(\hat{\mathbf{P}})\|^2 \tag{A.2}$$

which can be perfectly represented as a nonlinear least-squares problem.

In the following, we will explain how we can use the popular Gauss-Newton algorithm to find an approximate solution to this nonlinear least-squares problem. First, let us assume that we have a somewhat reasonable estimate of the 3D point $\hat{\mathbf{P}}$, which we can compute by the previous linear method. The key insight of the Gauss-Newton algorithm is to update our estimate by correcting it towards an even better estimate that minimizes the reprojection error. At each step we want to update our estimate $\hat{\mathbf{P}}$ by some $\delta_{\mathbf{P}}$: $\hat{\mathbf{P}} \leftarrow \hat{\mathbf{P}} + \delta_{\mathbf{P}}$.

But how do we choose the update parameter $\delta_{\mathbf{P}}$? The key insight of the Gauss-Newton algorithm is to linearize the residual function near the current estimate $\hat{\mathbf{P}}$. In the case of our problem, this means that the residual error $\mathbf{e}$ of a point $\mathbf{P}$ can be thought of as:

$$\mathbf{e}(\hat{\mathbf{P}} + \delta_{\mathbf{P}}) \approx \mathbf{e}(\hat{\mathbf{P}}) + \frac{\partial \mathbf{e}}{\partial \mathbf{P}} \delta_{\mathbf{P}} \tag{A.3}$$

Subsequently, the minimization problem transforms into

$$\min_{\delta_{\mathbf{P}}} \|\frac{\partial \mathbf{e}}{\partial \mathbf{P}} \delta_{\mathbf{P}} - (-\mathbf{e}(\hat{\mathbf{P}}))\|^2 \tag{A.4}$$

When we formulate the residual like this, we can see that it takes the format of the standard linear least-squares problem. For the triangulation problem with N images, the linear least-squares solution is

$$\delta_{\mathbf{P}} = -(J^T J)^{-1} J^T \mathbf{e} \tag{A.5}$$

where

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_N \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 - M_1\hat{\mathbf{P}} \\ \vdots \\ \mathbf{p}_N - M_N\hat{\mathbf{P}} \end{bmatrix} \tag{A.6}$$

and

$$J = \begin{bmatrix} \frac{\partial \mathbf{e}_1}{\partial \hat{P}_1} & \frac{\partial \mathbf{e}_1}{\partial \hat{P}_2} & \frac{\partial \mathbf{e}_1}{\partial \hat{P}_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{e}_N}{\partial \hat{P}_1} & \frac{\partial \mathbf{e}_N}{\partial \hat{P}_2} & \frac{\partial \mathbf{e}_N}{\partial \hat{P}_3} \end{bmatrix} \tag{A.7}$$

Recall that the residual error vector of a particular image $\mathbf{e}_i$ is a 2×1 vector because there are two dimensions in the image plane. Consequently, in the simplest two camera case ($N = 2$) of triangulation, this results in the residual vector $\mathbf{e}$ being a $2N \times 1 = 4 \times 1$ vector and the Jacobian J being a $2N \times 3 = 4 \times 3$ matrix. Notice how this method handles multiple views seamlessly, as additional images are accounted for by adding the corresponding rows to the $\mathbf{e}$ vector and J matrix. After computing the update $\delta_{\mathbf{P}}$, we can simply repeat the process for a fixed number of steps or until it numerically converges. One important property of the Gauss-Newton algorithm is that our assumption that the residual function is linear near our estimate gives us no guarantee of convergence. Thus, it is always useful in practice to put an upper bound on the number of updates made to the estimate.

Appendix B The algebraic approach to structure from motion

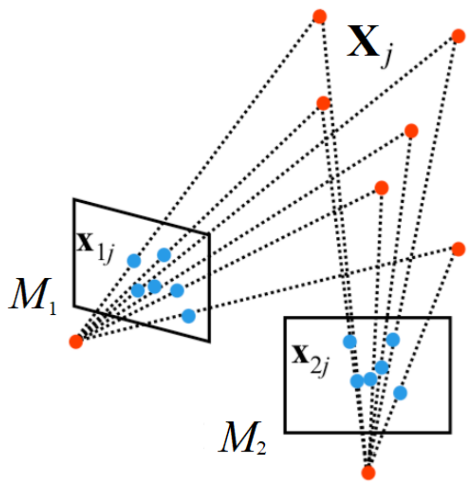

Figure 5: In the algebraic approach, we consider sequential camera pairs to determine camera matrices M_1 and M_2 up to a perspective transformation.

.

We will now cover the *algebraic approach*, which leverages the concept of the fundamental matrix F for solving the structure from motion problem for two cameras. As shown in Figure 5, the main idea of the algebraic approach is to compute two camera matrices M_1 and M_2, which can only be computed up to a perspective transformation H. Since each M_i can only be computed up a perspective transformation H, we can always consider a H such that the first camera projection matrix M_1H^{-1} is canonical. Of course, the same transformation must also be applied to the second camera which leads to the form shown:

$$M_1 H^{-1} = [I \;\; \mathbf{0}] \qquad M_2 H^{-1} = [A \;\; \mathbf{b}] \tag{B.1}$$

To accomplish this task, we must first compute the fundamental matrix F using the eight-point algorithm covered in the previous course notes. We now will use F to estimate the projective camera matrices M_1 and M_2. To do this estimation, we define $\mathbf{P}$ to be the corresponding 3D point for the corresponding observations in the images $\mathbf{p}$ and $\mathbf{p}'$. Since we have applied H^{-1} to both camera projection matrices, we must also apply H to the structure, giving us $\widetilde{\mathbf{P}} = H\mathbf{P}$. Therefore, we can relate the pixel coordinates $\mathbf{p}$ and $\mathbf{p}'$ to the transformed structure as follows:

$$\begin{aligned} \mathbf{p} &= M_1\mathbf{P} = M_1 H^{-1} H\mathbf{P} = [I \mid \mathbf{0}]\widetilde{\mathbf{P}} \\ \mathbf{p}' &= M_2\mathbf{P} = M_2 H^{-1} H\mathbf{P} = [A \mid \mathbf{b}]\widetilde{\mathbf{P}} \end{aligned} \tag{B.2}$$

An interesting property between the two image correspondences $\mathbf{p}$ and $\mathbf{p}'$ occur by some creative substitutions:

$$\begin{aligned} \mathbf{p}' &= [A|\mathbf{b}]\widetilde{\mathbf{P}} \\ &= A[I|\mathbf{0}]\widetilde{\mathbf{P}} + \mathbf{b} \\ &= A\mathbf{p} + \mathbf{b} \end{aligned} \tag{B.3}$$

Using Equation B.3, we can write the cross product between $\mathbf{p}'$ and $\mathbf{b}$ as:

$$\mathbf{p}' \times \mathbf{b} = (A\mathbf{p} + \mathbf{b}) \times \mathbf{b} = A\mathbf{p} \times \mathbf{b} \tag{B.4}$$

By the definition of cross product, $\mathbf{p}' \times \mathbf{b}$ is perpendicular to $\mathbf{p}'$. Therefore, we can write:

$$\begin{aligned} 0 &= \mathbf{p}'^T(\mathbf{p}' \times \mathbf{b}) \\ &= \mathbf{p}'^T(A\mathbf{p} \times \mathbf{b}) \\ &= \mathbf{p}'^T(\mathbf{b} \times A\mathbf{p}) \\ &= \mathbf{p}'^T[\mathbf{b}_\times]A\mathbf{p} \end{aligned} \tag{B.5}$$

Looking at this constraint should remind you of the general definition of the fundamental matrix $\mathbf{p}'^T F\mathbf{p} = 0$. If we set $F = [\mathbf{b}_\times]A$, then extracting A and $\mathbf{b}$ simply breaks down to a decomposition problem.

Let us begin by determining $\mathbf{b}$. Again, by the definition of cross product, we can simply write $F\mathbf{b}$ as

$$F\mathbf{b} = [\mathbf{b}_\times]A\mathbf{b} = \mathbf{b} \times (A\mathbf{b}) = \mathbf{0}. \tag{B.6}$$

Since F is singular, $\mathbf{b}$ can be computed as a least-square solution of $F\mathbf{b} = \mathbf{0}$, with $||\mathbf{b}|| = 1$, using SVD.

Once $\mathbf{b}$ is known, we can now compute A. If we set $A = -[\mathbf{b}_\times]F$, then we can verify that this definition satisfies $F = [\mathbf{b}_\times]A$:

$$\begin{aligned} [\mathbf{b}_\times]A' &= -[\mathbf{b}_\times][\mathbf{b}_\times]F \\ &= (\mathbf{b}\mathbf{b}^T - |\mathbf{b}|^2 I)F \\ &= \mathbf{b}\mathbf{b}^T F + |\mathbf{b}|^2 F \\ &= 0 + 1 \cdot F \\ &= F \end{aligned} \tag{B.7}$$

Consequently, we determine the two expressions for our camera matrices $M_1 H^{-1}$ and $M_2 H^{-1}$:

$$\tilde{M}_1 = [I \quad \mathbf{0}] \qquad \tilde{M}_2 = [-[\mathbf{b}_\times] F \quad \mathbf{b}] \tag{B.8}$$

Before we conclude this section, we want to give a geometrical interpretation for $\mathbf{b}$. We know $\mathbf{b}$ satisfies $F\mathbf{b} = \mathbf{0}$. Remember the epipolar constraints we derived in the previous course notes, which found that the epipoles in an image are the points that map to zero when transformed by the fundamental matrix (i.e. $F\mathbf{e}_2 = \mathbf{0}$ and $F^T\mathbf{e}_1 = \mathbf{0}$). We can see, therefore, that $\mathbf{b}$ is an epipole. This provides a new set of equations for the camera projection matrices (Eqs. B.9).

$$\tilde{M}_1 = [I \quad \mathbf{0}] \qquad \tilde{M}_2 = [-[\mathbf{e}_\times] F \quad \mathbf{e}] \tag{B.9}$$