MR21 (CS7.503) - Take Home End Semester Exam

Avneesh Mishra *
avneesh.mishra@research.iiit.ac.in

November 28, 2021

Take home end semester exam for **Mobile Robotics** (CS7.503 by **Prof. K. Madhava Krishna**) Monsoon 2021 batch. This document contains the answers to `Theory Questions`.

Contents

List of Figures

*Roll No.: 2021701032

1 Fundamental Matrix

Brief Description and Notation

Please refer to Figure 1.1 for the notations and meaning.

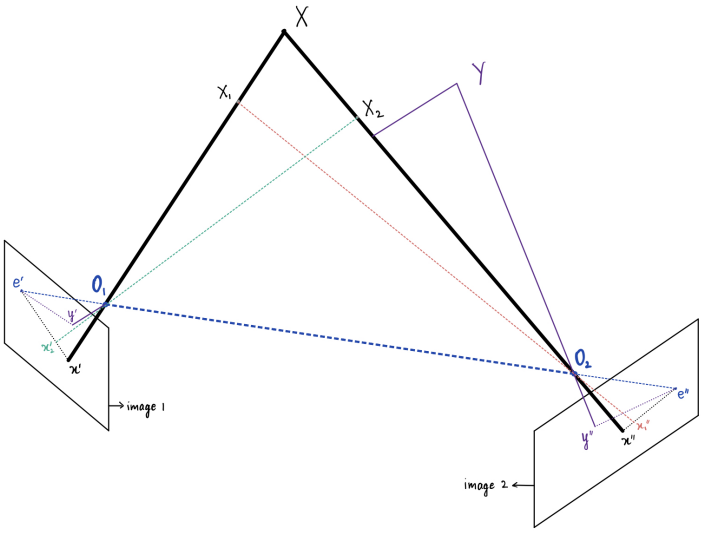


Figure 1.1: Epipolar Geometry

Points X and Y are points in the real world whose image falls at (pixel locations) x' and y' in image 1 and x'' and y'' in image 2. The origins of these cameras is located at O_1 and O_2 respectively. The line joining O_1 and O_2 is called the **epipolar axis**. It intersects the images at e' and e'' respectively.

It is clear that the image of X_1 in camera 1 will also fall on x' (same line), similarly the image of X_2 in camera 2 will also fall on x''. However, the image of X_1 in camera 2 will fall at x_1'', which is on the line joining x'' and e''. This line is called the **epipolar line**. Similarly, the image of X_2 in camera 1 will fall at x_2' (which is also on the line joining e' and x'). When X_1 moves along $\overline{XO_1}$, its image x_1'' traces a line in the second image (the *epoipolar line*). Same can be said for X_2 and the first image.

Since O_1O_2X form a plane (called the **epipolar plane**), the triple product of these three vectors will be zero, that is $\langle O_1X \;\; O_1O_2 \;\; O_2X\rangle = \mathbf{0}$, where $\left\langle \vec{a} \;\; \vec{b} \;\; \vec{c} \right\rangle = \vec{a} \cdot \left(\vec{b} \times \vec{c}\right)$ is the volume of the parallelepiped formed by the three vectors. From camera projection properties we can write

$$x' = \mathbf{K}'\mathbf{R}' \left[\mathbf{I} \mid -\mathbf{X}_{O'}\right] X \qquad\qquad x'' = \mathbf{K}''\mathbf{R}'' \left[\mathbf{I} \mid -\mathbf{X}_{O''}\right] X \tag{1.1}$$

Where $\mathbf{K}'$, $\mathbf{R}' \left[\mathbf{I} \mid -\mathbf{X}_{O'}\right]$ and $\mathbf{K}''$, $\mathbf{R}'' \left[\mathbf{I} \mid -\mathbf{X}_{O''}\right]$ are camera intrinsic and extrinsic parameters (for camera 1 and camera 2) respectively. Note that all the above terms are in *homogeneous coordinates*. The vector X can be assumed to be unit-scale.

We know that $\overrightarrow{O_1X} = X - X_{O'} \equiv \mathbf{R}'^{-1}\mathbf{K}'^{-1}x'$ and $\overrightarrow{O_2X} = X - X_{O''} \equiv \mathbf{R}''^{-1}\mathbf{K}''^{-1}x''$. Another reduction is $\overrightarrow{O_1O_2} = b$ for the baseline vector (joining the two camera centers). Therefore, the triple product constraint mentioned above can be reduced to

$$\langle O_1X \;\; O_1O_2 \;\; O_2X\rangle = \mathbf{0} \Rightarrow (X - X_{O'}) \cdot (b \times (X - X_{O''})) \equiv \left(\mathbf{R}'^{-1}\mathbf{K}'^{-1}x'\right) \cdot \left(b \times \left(\mathbf{R}''^{-1}\mathbf{K}''^{-1}x''\right)\right) = 0$$

Using $a \cdot b = a^\top b$ and $a \times b = [a]_\times b$ (wehre $[a]_\times$ is the cross product skew symmetric matrix), we can reduce the above equation to

$$\begin{aligned}\langle O_1X \;\; O_1O_2 \;\; O_2X\rangle &= \left(\mathbf{R}'^{-1}\mathbf{K}'^{-1}x'\right) \cdot \left(b \times \left(\mathbf{R}''^{-1}\mathbf{K}''^{-1}x''\right)\right) = \left(\mathbf{R}'^{-1}\mathbf{K}'^{-1}x'\right)^\top [b]_\times \left(\mathbf{R}''^{-1}\mathbf{K}''^{-1}x''\right) \\ &= x'^\top \left(\mathbf{K}'^{-\top}\mathbf{R}'^{-\top} [b]_\times \mathbf{R}''^{-1}\mathbf{K}''^{-1}\right) x'' = x'^\top \mathbf{F} x'' = 0\end{aligned} \tag{1.2}$$

The equation 1.2 is the basis for two points (in different images) to be projected to the same point in the 3D world. Note that if the points correspond, then they satisfy the equation. This is not true the other way round as we'll see later.

1.1 Epipole condition

Epipolar line

Say we have a point x' in one image, and we want to find the correlating point x'' in a second image. Assume that we have $\mathbf{F}$ (the fundamental matrix) relating the two images. This matrix can be obtained from methods like the eight-point correspondence where eight non-coplaner point correspondences in the two images are known.

Referring to figure 1.1, our job would become much easier if we know the *epipolar line* of x' in the second image (the line $e''x''$). Let us call this line l'' (since it's in the second image).

For a true x'' to lie on l'', it must satisfy $x'' \cdot l'' = x''^\top l'' = l''^\top x'' = 0$. From equation 1.2, we know that $x'\mathbf{F}x'' = 0$. Matching the two results, we get $l''^\top = x'\mathbf{F} \Rightarrow l'' = \mathbf{F}^\top x'$ as the equation of the epipolar line in the second image (of the point x' in the first image). Now, a search along this line in the second image has higher chances of yielding the true x''.

Epipoles

We know that the epipolar line in the second image (of a point x' in the first image) is given by $l'' = \mathbf{F}^\top x'$.

We know that the epipole e'' (in the second image) is the projection of O_1 in the second image. That is $e'' = \mathbf{P}''X_{O''}$ (where $\mathbf{P}'' = \mathbf{K}''\mathbf{R}''\left[\mathbf{I} \mid -\mathbf{X}_{O''}\right]$ is the second camera's projection matrix). We also know that the epipole e'' lies on line l'', since all epipolar lines intersect at the epipoles (this is seen by considering another epipolar plane with a 3D point Y in figure 1.1). We therefore have $l''^\top e'' = 0$.

For any point x' in the first image, there will be an epipolar line l'' in the second image. We therefore have

$$l''^\top e'' = \left(\mathbf{F}^\top x'\right)^\top e'' = x'^\top \mathbf{F} e'' = \left(\mathbf{F}e''\right)^\top x' = 0 \tag{1.3}$$

We have two conditions: x' can be any valid point in image 1 (in homogeneous coordinates) *and* equation 1.3 always has to hold true. The only possibility where both these conditions hold true is when $\left(\mathbf{F}e''\right)^\top = \mathbf{0}^\top$ (it is a row of three zeros). Therefore, we have

$$\left(\mathbf{F}e''\right)^\top = \mathbf{0}^\top \Rightarrow \mathbf{F}e'' = \mathbf{0} \tag{1.4}$$

We therefore get $\mathbf{F}e'' = \mathbf{0}$ where e'' is the epipole of the first camera seen in the second image. In other words, the epipole e'' is the null space of matrix $\mathbf{F}$. This can be found by obtaining the eigenvector with the least (ideally zero) eigenvalue.

1.2 Fundamental Matrix relations

Equation 1.2 gives the fundamental matrix relating image 1 to image 2 (simply because point x' in image 1 comes before point x'' which is in image 2). Transposing it gives

$$\left(x'^\top\right)_{1,3} \mathbf{F}_{3,3} \left(x''\right)_{3,1} = 0 \Rightarrow \left(x'^\top \mathbf{F} x''\right)^\top = 0 \Rightarrow x''^\top \mathbf{F}^\top x' = 0 \tag{1.5}$$

Therefore, the fundamental matrix relating the second image to the first is given by the *transpose*. That is, if ${}^1_2\mathbf{F} = \mathbf{F}$, then ${}^2_1\mathbf{F} = \mathbf{F}^\top$.

This difference mainly arises from the derivation of 1.2. The condition on the triple product still holds true, even if we swap O_1 and O_2. This is equivalent to centering the equations about the second camera / image instead of the first. We can prove this as follows

$$\begin{aligned}\langle O_2X \;\; O_2O_1 \;\; O_1X\rangle &= \left(\mathbf{R}''^{-1}\mathbf{K}''^{-1}x''\right) \cdot \left(-b \times \left(\mathbf{R}'^{-1}\mathbf{K}'^{-1}x'\right)\right) = \left(\mathbf{R}''^{-1}\mathbf{K}''^{-1}x''\right)^\top [-b]_\times \left(\mathbf{R}'^{-1}\mathbf{K}'^{-1}x'\right) \\ &= x''^\top \left(\mathbf{K}''^{-\top}\mathbf{R}''^{-\top} [-b]_\times \mathbf{R}'^{-1}\mathbf{K}'^{-1}\right) x' = x''^\top \, {}^2_1\mathbf{F} x' = 0\end{aligned} \tag{1.6}$$

Note that $\mathbf{R}^\top = \mathbf{R}^{-1}$ (for both $\mathbf{R}'$ and $\mathbf{R}''$) since they're orthogonal matrices. Also note that $[-b]_\times = -[b]_\times = [b]_\times^\top$ since it is a skew symmetric matrix.

$${}^2_1\mathbf{F} = \left(\mathbf{K}''^{-\top}\mathbf{R}''^{-\top} [-b]_\times \mathbf{R}'^{-1}\mathbf{K}'^{-1}\right) = \left(\mathbf{K}'^{-\top}\mathbf{R}'^{-\top} [b]_\times \mathbf{R}''^{-1}\mathbf{K}''^{-1}\right)^\top = {}^1_2\mathbf{F}^\top \tag{1.7}$$

The equations 1.5 and 1.7 formally prove the transpose relationship.

1.3 Fundamental and Essential Matrix

The fundamental matrix is used for uncalibrated cameras where we are not interested in knowing the intrinsic parameters of the two cameras. The **essential matrix** is used for relating images from two *calibrated* cameras, that is, we know the intrinsic matrices explicitly.

We can get the projected rays (in 3D space) for the pixel homogeneous coordinates (in both the images) as ${}^{k}X' = \mathbf{K}'^{-1}x'$ and ${}^{k}X'' = \mathbf{K}''^{-1}x''$. Say that the pixels correspond, that is, these rays intersect each other. The equation 1.2 for the fundamental matrix can then be written as

$$\begin{aligned}
&\rightarrow x'^{\top}\left(\mathbf{K}'^{-\top}\mathbf{R}'^{-\top}\left[b\right]_{\times}\mathbf{R}''^{-1}\mathbf{K}''^{-1}\right)x'' = \left(\mathbf{K}'^{-1}x'\right)^{\top}\left(\mathbf{R}'^{-\top}\left[b\right]_{\times}\mathbf{R}''^{-1}\right)\left(\mathbf{K}''^{-1}x''\right) = 0 \\
&\Rightarrow {}^{k}X'^{\top}\ \left(\mathbf{R}'^{-\top}\left[b\right]_{\times}\mathbf{R}''^{-1}\right)\ {}^{k}X'' = {}^{k}X'^{\top}\ \mathbf{E}\ {}^{k}X'' = 0 \Rightarrow \mathbf{E} = \mathbf{R}'^{-\top}\left[b\right]_{\times}\mathbf{R}''^{-1}
\end{aligned} \tag{1.8}$$

Basucally, the essential matrix relates images from calibrated cameras (whose projected rays can be resolved). Also, correlating equations 1.2 and 1.8 we get

$$\mathbf{F} = \left(\mathbf{K}'^{-\top}\mathbf{R}'^{-\top}\left[b\right]_{\times}\mathbf{R}''^{-1}\mathbf{K}''^{-1}\right) = \mathbf{K}'^{-\top}\left(\mathbf{R}'^{-\top}\left[b\right]_{\times}\mathbf{R}''^{-1}\right)\mathbf{K}''^{-1} = \mathbf{K}'^{-\top}\ \mathbf{E}\ \mathbf{K}''^{-1} \tag{1.9}$$

The equation 1.9 gives the relation between fundamental matrix $\mathbf{F}$ and essential matrix $\mathbf{E}$.

Also, note that the fundamental matrix has 7 degrees of freedom whereas the essential matrix has 5 degrees of freedom.

2 Relationships Between Image Planes

2.1 Homography relation for pure rotation

A camera's projection equation can be given as (revisit equation 1.1)

$$\mathbf{x} = \mathbf{K}\mathbf{R}[\mathbf{I} \mid -\mathbf{X}_O]\ \mathbf{X} \tag{2.1}$$

Where $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{R}$ is the rotation matrix of the camera (expressed in the real world coordinates), $\mathbf{X}_O$ is the origin of the camera's projection center and the point $\mathbf{X}$ is the point in scene expressed in homogeneous coordinates. The point $\mathbf{x}$ is the location of the point in the projected image plane (also in homogeneous coordinates).

Note that since there is a dimansion lost ($\mathbf{x}$ is $\mathbb{P}^2$ whereas $\mathbf{X}$ is $\mathbb{P}^3$), we cannot truly recover the point $\mathbf{X}$ from just a pixel location $\mathbf{x}$ in the image. However, we can recover the *line* passing through the camera center that yields the point $\mathbf{x}$ (for any point on that line). This line can be rotated and projected back as a pixel.

Assume that the initial frame of the camera is given by $\{1\}$ (image pixels represented by $\mathbf{x}'$), the world frame is given by $\{0\}$ and the new camera frame (after *strict* rotation) is given by $\{2\}$ (image pixels represented by $\mathbf{x}''$). Writing the projection equations, we get

$$\mathbf{x}' = \mathbf{K}\,{}^1_0\mathbf{R}[\mathbf{I} \mid -{}_0\mathbf{X}_O]\ {}_0\mathbf{X} \qquad\qquad \mathbf{x}'' = \mathbf{K}\,{}^2_0\mathbf{R}[\mathbf{I} \mid -{}_0\mathbf{X}_O]\ {}_0\mathbf{X} \tag{2.2}$$

Note that the camera center and the point (${}_0\mathbf{X}_O$ and ${}_0\mathbf{X}$ in $\mathbb{P}^3$) are represented in the world frame (frame $\{0\}$), and are therefore unchanged. Also note that the two poses of the camera are related as

$${}^0_2\mathbf{R} = {}^0_1\mathbf{R}\,{}^1_2\mathbf{R} \Rightarrow {}^2_0\mathbf{R} = {}^0_2\mathbf{R}^\top = {}^1_2\mathbf{R}^\top\,{}^0_1\mathbf{R}^\top \Rightarrow {}^2_0\mathbf{R} = {}^2_1\mathbf{R}\,{}^1_0\mathbf{R} \tag{2.3}$$

Where ${}^2_1\mathbf{R}$ is $\{1\}$'s orientation expressed in $\{2\}$. Substituting the result of equation 2.3 in equation 2.2, and noting that we're dealing with homogeneous coordinates here (uniformly scaled values are the same), we get

$$\begin{aligned}
&\rightarrow \mathbf{x}' = \mathbf{K}\,{}^1_0\mathbf{R}[\mathbf{I} \mid -{}_0\mathbf{X}_O]\ {}_0\mathbf{X} \Rightarrow \mathbf{K}^{-1}\mathbf{x}' \equiv\ {}^1_0\mathbf{R}[\mathbf{I} \mid -{}_0\mathbf{X}_O]\ {}_0\mathbf{X} \\
&\rightarrow \mathbf{x}'' = \mathbf{K}\,{}^2_0\mathbf{R}[\mathbf{I} \mid -{}_0\mathbf{X}_O]\ {}_0\mathbf{X} \Rightarrow \mathbf{x}'' = \mathbf{K}\,{}^2_1\mathbf{R}\,\left({}^1_0\mathbf{R}\,[\mathbf{I} \mid -{}_0\mathbf{X}_O]\ {}_0\mathbf{X}\right) \\
&\Rightarrow \mathbf{x}'' = \mathbf{K}\,{}^2_1\mathbf{R}\,\mathbf{K}^{-1}\mathbf{x}' = \mathbf{H}\,\mathbf{x}' \qquad \text{where } \mathbf{H} = \mathbf{K}\,{}^2_1\mathbf{R}\,\mathbf{K}^{-1}
\end{aligned} \tag{2.4}$$

The equation 2.4 gives the resulting homography $\mathbf{H}$ (for pure rotation), relating pixels $\mathbf{x}'$ in first image to pixels $\mathbf{x}''$ in the second image.

When there is a camera translation involved, then this **does not** hold. Precisely because the camera projection center has moved (it will no longer remain ${}_0\mathbf{X}_O$, it'll become something else). The substitution for $\mathbf{x}'$ done in 2.4 will not hold good.

Geometrically, since we've moved, the line $\mathbf{K}^{-1}\mathbf{x}'$ will no longer project to a single point $\mathbf{x}''$ in the new image. Instead, this line will project to another (special) line in the new image (called the *epipolar line*, see figure 1.1). We will then not be able to generate a unique $\mathbf{x}''$ (note that the task here is to *generate* a new image, not correspond).

2.2 Relating pixels b/w frames

Methods using which you can relate a pixel $\mathbf{x}'$ in image 1 (pixel x_{i1} frame $I1$) to a pixel $\mathbf{x}''$ in image 2 (pixel x_{j2} in frame $I2$) are described hereon.

Fundamental Matrix

As described in equation 1.2, the fundamental matrix relates two pixels in cases when we do *not* know the camera intrinsic parameters (the cameras are uncalibrated). This relation is given by

$$x'^\top \mathbf{F} x'' = 0 \qquad \text{where } \mathbf{F} = \mathbf{K}'^{-\top}\mathbf{R}'^{-\top}\,[b]_\times\,\mathbf{R}''^{-1}\mathbf{K}''^{-1} \tag{2.5}$$

Note that if the pixel lie on each other's epipoles (not necessarily mapping to the same point in the 3D world), even then the relation holds true. Therefore, we can note that

If the pixels correspond to the same point, the equation for fundamental matrix holds true. But if the equation for fundamental matrix holds true, the pixels need not correspond to the same point in the 3D world.

It is also important to note that this relation comes from epipolar geometry and the matrix $\mathbf{F}$ degenerates (becomes zeros) if the baseline $\mathbf{b}$ is zero (meaning that there is no translation).

Essential Matrix

As described in equation 1.8, the essential matrix relates two pixels in cases when we *know* the camera intrinsic parameters $\mathbf{K}'$ and $\mathbf{K}''$ (cameras are calibrated). This relation is given by

$$\left(\mathbf{K}'^{-1}\mathbf{x}'\right)^{\top} \mathbf{E} \left(\mathbf{K}''^{-1}\mathbf{x}''\right) = 0 \qquad \text{where } \mathbf{E} = \mathbf{R}'^{-\top} [b]_{\times} \mathbf{R}''^{-1} \tag{2.6}$$

This relation, like the fundamental matrix also comes from epipolar geometry (the same triple product equal to 0 condition). Therefore, it too holds true only when the baseline is non-zero (where there is translation between the two frames).

Rotational homographies

If the cameras are calibrated (we know the intrinsics), and there has only been rotation (no translation) about the camera's projection center (from one view to another), then the fundamental and essential matrix degenerate. The relation is then derived by rotational homography described below

$$\mathbf{x}'' = \mathbf{H}\,\mathbf{x}' \qquad \text{where } \mathbf{H} = \mathbf{K}\,{}^{2}_{1}\mathbf{R}\,\mathbf{K}^{-1} \tag{2.7}$$

This is equivaluent to projecting the rotated ray (formed by inverse projection of pixel $\mathbf{x}'$ in frame 1) into the frame 2 and getting the corresponding pixel.

Stereo Pairs

Another interesting point to note is that when the cameras are arranged in a stereo pair, with there being no rotation and the baseline having a value only in X (only horizontal shift in the cameras), the fundamental (and essential) matrices reduce to skew-symmetric matrices and the epipolar lines become horizontal (the null space is only possible for horizontal lines).

In this case, knowing the coordinates of the pixels $\mathbf{x}'$ and $\mathbf{x}''$ (as say: x', y' and x'', y''), can yield us the 3D coordinates of the corresponding point (in the 3D world, if the pixels correspond to the same point). We need to know the focal length (say c) of the cameras for this (the cameras have to be calibrated) and the images need to be un-distorted (all lense effects removed). Also not that $y'' = y'$ (the images should be horizontally aligned).

The point in the 3D world has the coordinates X, Y, Z, given by

$$Z = \frac{c\,B}{-(x'' - x')} \qquad X = \frac{x'\,B}{-(x'' - x')} \qquad Y = \frac{y' + y''}{2}\frac{B}{-(x'' - x')} = \frac{y'\,B}{-(x'' - x')} \tag{2.8}$$

The X offsets of the pixels make a special image called the disparity map. Note that this still is up to an affine transform, there can be scaling, rotation and translation in the point in the real 3D world. This can be resolved through correspondences.

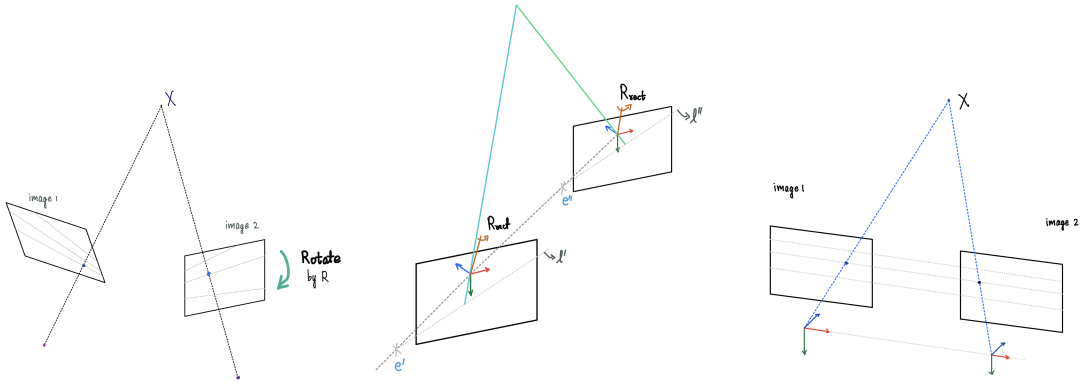
Correspondence search

If we're looking for correspondences (we have $\mathbf{x}'$ in first image and want to look for corresponding $\mathbf{x}''$ in the second image), the following methods could be useful

1. If the cameras form a stereo pair, then searching along the same horizontal line will be enough.
2. If the fundamental matrix is known, the epipolar lines can be found and a linear search can be done.
3. Coreners can be detected on the image (using gradient kernels) and neighborhood searches can be done.

2.3 Stereo Rectification

Refer to Figure 2.1 for the process information (how different stages look).



(a) Stage 1 (b) Stage 2 (c) Stage 3

Figure 2.1: Stereo Rectification Process
In the first stage (sub-fig. a), the cameras have a rotation and a translation. Notice how the epipolar lines are arranged (grey dotted lines).
A rotation (**first homography**) to the second image gives stage 2 (sub-fig. b). Here, the images have no rotation (same orientation of cameras), but still there are (finite) epipoles, as shown by the grey line.
Another rotation (**second homography**) has to be applied to both the images, so that they become co-planar and that the second camera's location is at the +X axis of the first. This way, all epipolar lines will become horizontal (as shown in sub-fig. c).

Before getting started, we need to calibrate the cameras (so that we know the intrinsics). After the calibration, we proceed with the following steps

1. **First homography**: Here, we estimate the *essential matrix* using 5 or 8 correspondences (depends on the 5-point or 8-point algorithm being used). Once we know $\mathbf{E}$, we represent the essential matrix as $\mathbf{E} = [\mathbf{b}]_\times \mathbf{R}^\top$. Here, we assume the first camera's frame as the reference. Using SVD of $\mathbf{E}$ (and resolving it into two parts), it is possible to get $\mathbf{R}$ (and the baseline translation vector $\mathbf{t} = \mathbf{b}$). Therefore, the first homography is to align the two images (align the second image with the first). The system changes from figure 2.1a to b. Note from 2.4 that the second image is transformed as $\mathbf{x}_2'' = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\,\mathbf{x}''$.

2. **Second homography**: From figure 2.1b, it is clear that even when the images are aligned, they needn't have poles at infinity. We need the poles at infinity and we also need the epipolar lines to be horizontal. Specifically, we need the epipoles at $\mathbf{e}' = [e_x' \; 0 \; 0]$ and $\mathbf{e}'' = [e_x'' \; 0 \; 0]$ (in $\mathbb{P}^2$, in both the images), basically on the X axis at infinity.

   Note that epipoles are the null space of the fundamental matrix $\mathbf{F}$ (between the images). We need to construct a rotation matrix such that the new frames have only translational offset. Let's say that for stage 2, as shown in figure 2.1b, we get the epipoles as $\mathbf{e}' = [e_x' \; e_y' \; 1]$ (scaled to unit scaling factor). We resolve the rotation matrix $\mathbf{R}_{\text{rect}}$ as

$$\mathbf{r}_1 = \widehat{\mathbf{K}^{-1}\mathbf{e}''} = \frac{\overrightarrow{\mathbf{t}}}{\|\mathbf{t}\|} \qquad \mathbf{r}_2 = [0\;0\;1] \times \widehat{\mathbf{t}} = \frac{1}{\sqrt{t_x^2 + t_y^2}} [-t_y \; t_x \; 0]^\top \qquad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{2.9}$$

   Note that these are intuitively derived unit vectors to rotate the frame (of the left and right cameras) in figure b to the frame (of the left and right cameras) in figure 2.1c. The final rotation matrix is

$$\mathbf{R}_{\text{rect}} = \begin{bmatrix} \mathbf{r}_1^\top \\ \mathbf{r}_2^\top \\ \mathbf{r}_3^\top \end{bmatrix} \tag{2.10}$$

Note that this homography of $\mathbf{H} = \mathbf{K}\,\mathbf{R}_{\text{rect}}\,\mathbf{K}^{-1}$ has to be applied to both the images. This brings us from figure 2.1b to figure 2.1c.

Finally, image one has become $\mathbf{x}'_3 = \mathbf{K}\,\mathbf{R}_{\text{rect}}\,\mathbf{K}^{-1}\,\mathbf{x}'$.

And image two has become $\mathbf{x}''_3 = \mathbf{K}\,\mathbf{R}_{\text{rect}}\,\mathbf{K}^{-1}\,\mathbf{x}''_2$, which is the same as $\mathbf{x}''_3 = \mathbf{K}\,\mathbf{R}_{\text{rect}}\,\mathbf{R}\,\mathbf{K}^{-1}\,\mathbf{x}''$.

Also, note that since the basis vector was chosen as the $\mathbf{r}_1$, it will map the epipole to X axis (at infinity), other two $\mathbf{r}$ vectors will give zero as the dot product (they're by design perpendicular).

3. There may be a scaling to get the pixels to the *image pixel* representation (by scaling the last element in $\mathbb{P}^2$ vectors to 1).

The first two steps above are called homographies because they largely deal with homogeneous coordinates (both as input and output) and also because they map points in one image directly to points in the other. This was also seen in equation 2.4.

3 Camera Calibration

3.1 DLT Algorithm

The **Direct Linear Transform** is a method to extract a homogeneous relation between corresponding points. In the context of camera calibration, the aim is to estimate the camera extrinsic and intrinsic parameters in the form of the projection matrix. For this, we are given the point correspondences (pairs of relations, relating the 3D point coordinates in the real world with the 2D image coordinates). That is, we're given a set of $\mathbf{X}_i \leftrightarrow \mathbf{x}_i = [X_i, Y_i, Z_i, 1] \leftrightarrow [x_i, y_i, 1]$ as correspondences. Note that the equation for projection is given by

$$\mathbf{x}_i = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\mathbf{X}_o\right]\ \mathbf{X}_i \Rightarrow \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\mathbf{X}_o\right] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ W_i \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ W_i \end{bmatrix} \tag{3.1}$$

Where $\mathbf{P} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\mathbf{X}_o\right]$ is a 3-by-4 camera projection matrix. This projection equation is rearranged into a form where finding the null space (or approximate) gives the resultant projection matrix as a vector. We can do this form conversion as shown below

$$\mathbf{x}_i = \mathbf{P}\mathbf{X}_i \Rightarrow \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{a}^\top \\ \mathbf{b}^\top \\ \mathbf{c}^\top \end{bmatrix} \mathbf{X}_i \equiv \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\rightarrow x_i = \frac{\mathbf{a}^\top \mathbf{X}_i}{\mathbf{c}^\top \mathbf{X}_i} \Rightarrow x_i\, \mathbf{c}^\top \mathbf{X}_i = \mathbf{a}^\top \mathbf{X}_i \Rightarrow -\mathbf{X}_i^\top \mathbf{a} + x_i\, \mathbf{X}_i^\top \mathbf{c} = 0 \Rightarrow \begin{bmatrix} -\mathbf{X}_i^\top & \mathbf{0}^\top & x_i\, \mathbf{X}_i^\top \end{bmatrix}_{1,12} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}_{12,1} = 0$$

$$\rightarrow y_i = \frac{\mathbf{b}^\top \mathbf{X}_i}{\mathbf{c}^\top \mathbf{X}_i} \Rightarrow y_i\, \mathbf{c}^\top \mathbf{X}_i = \mathbf{b}^\top \mathbf{X}_i \Rightarrow -\mathbf{X}_i^\top \mathbf{b} + y_i\, \mathbf{X}_i^\top \mathbf{c} = 0 \Rightarrow \begin{bmatrix} \mathbf{0}^\top & -\mathbf{X}_i^\top & y_i\, \mathbf{X}_i^\top \end{bmatrix}_{1,12} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}_{12,1} = 0$$

$$\Rightarrow \begin{bmatrix} -\mathbf{X}_i^\top & \mathbf{0}^\top & x_i\, \mathbf{X}_i^\top \\ \mathbf{0}^\top & -\mathbf{X}_i^\top & y_i\, \mathbf{X}_i^\top \end{bmatrix}_{2,12} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}_{12,1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}_{2,1} \Rightarrow \begin{bmatrix} \mathbf{q}_{x_i}^\top \\ \mathbf{q}_{y_i}^\top \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.2}$$

The equation 3.2 basically yields two equations for every corresponding point $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$. Since $\mathbf{P}$ has 12 elements (with 11 Degrees of Freedom, since it's a homogeneous matrix), we need *at least* six correspondences.

We get the following equation by stacking all the m correspondences

$$\begin{bmatrix} \mathbf{q}_{x_1}^\top \\ \mathbf{q}_{y_1}^\top \\ \vdots \\ \mathbf{q}_{x_i}^\top \\ \mathbf{q}_{y_i}^\top \\ \vdots \\ \mathbf{q}_{x_m}^\top \\ \mathbf{q}_{y_m}^\top \end{bmatrix}_{2m,12} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}_{12,1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}_{2m,1} \Rightarrow \mathbf{Q}_{(2m,12)} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}_{(12,1)} = \mathbf{0}_{(2m,1)} \Rightarrow \mathbf{Q}\,\mathbf{h} = \mathbf{0} \tag{3.3}$$

Note that m should at least be 6 for a valid null-space (however, that's not enough as we'll see later). The vector $\mathbf{h}$ in equation 3.3, which contains the rows of $\mathbf{P}$ stacked as a column vector, can basically be estimated as the null space of $\mathbf{Q}$. In reality, the equation 3.3 rarely holds exactly true. There's always noise in observations, loss due to digital sampling, etc. due to which the value is a small number (close to zero).

Hence, the aim (in finding $\mathbf{h}$) could be to *minimize* $\|\mathbf{Q}\,\mathbf{h}\|$ (the norm). In order to find a unique $\mathbf{h}$, as well as to avoid the trivial solution of $\mathbf{h} = \mathbf{0}$, we can add a constraint of $\|\mathbf{h}\| = 1$. This minimization can be done through *Singular Value Decomposition*. We get the solution of $\mathbf{h}$ being the last column of $\mathbf{V}$ (the right singular vectors; eigenvectors of $\mathbf{Q}^\top\,\mathbf{Q}$).

3.2 Solution through SVD

Performing the SVD of $\mathbf{Q}$, we get $\mathbf{Q} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ where $\mathbf{D}$ is a diagonal matrix. The new optimization objective becomes to minimize $\left\|\mathbf{U}\mathbf{D}\mathbf{V}^\top \mathbf{h}\right\|$ with the constraint $\|\mathbf{h}\| = 1$.

This objective can be modified, knowing that $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, as follows

$$\begin{aligned} & \min \|\mathbf{Q}\,\mathbf{h}\| && \|\mathbf{h}\| = 1 \\ \Rightarrow & \min \left\|\mathbf{U}\mathbf{D}\mathbf{V}^\top \mathbf{h}\right\| && \|\mathbf{h}\| = \left\|\mathbf{V}^\top \mathbf{h}\right\| = \|\mathbf{y}\| = 1 \\ \Rightarrow & \min \|\mathbf{D}\,\mathbf{y}\| && \|\mathbf{y}\| = 1 \end{aligned}$$

Note that $\mathbf{D}\,\mathbf{y}$ will just be a differently scaled multiples of the singular values of $\mathbf{Q}$. Hence, the minimum of $\|\mathbf{D}\,\mathbf{y}\|$ will be the smallest singular value (the constraint is $\|\mathbf{y}\| = 1$). We get the smallest singular value in the norm when $\mathbf{y}_{(12,1)} = [0, 0, \cdots, 0, 1]^\top$.

We know $\mathbf{V}^\top \mathbf{h} = \mathbf{y} \Rightarrow \mathbf{h} = \mathbf{V}\,\mathbf{y}$. Putting $\mathbf{y} = [0, 0, \cdots, 0, 1]^\top$ basically means that $\mathbf{h}$ is the *last column* of $\mathbf{V}$. Note that $\mathbf{V}$ is formed by the eigenvectors of $\mathbf{Q}^\top \mathbf{Q}$ and the last column has the smallest corresponding eigenvalue (they're related to singular values in the same manner).

3.3 Points on a plane

Note that in equation 3.2 the value $\mathbf{X}_i^\top = [X_i \;\; Y_i \;\; Z_i \;\; 1]$. The equation of a plane (in 3D) is given by $\alpha X + \beta Y + \gamma Z + \delta = 0$. This means that if all points $\mathbf{X}_i$ lie on a plane, it'll be possible to resolve columns of $\mathbf{Q}$ as a linear combination of other columns (say, resolve Z_i column as a linear combination of X_i, Y_i and 1 columns).

This means that the matrix $\mathbf{Q}$ will loose rank (note that it'll loose rank by multiples of 2, since the equations are in x, y pairs) and therefore no longer yield a vector $\mathbf{h}$ that can be uniquely resolved. If $\mathbf{Q}$ is full rank, then the solutions for $\mathbf{h}$ lie on a 1 DoF line in a 12 DoF space. This space will degenerate and the solutions for $\mathbf{h}$ will be *many*, $\mathbf{Q}$ loses rank. Hence, it is essential that we have at least 6 points and that they do **not** lie on a planar surface.

Even in the step for doing SVD, if $\mathbf{Q}$ is rank deficient (as discussed in the condition earlier), then the last column of $\mathbf{V}$ will be zeros: the least singular value (the last diagonal entry of $\mathbf{D}$) will become 0, therefore the last row of $\mathbf{V}^\top$ will be zeros or non-existent (depending on the implementation of SVD). Hence, solving for $\mathbf{h}$ will then become impossible.

4 SLAM/SfM

4.1 SAM Optimization in SE2

Assuming that other than the observations and poses, nothing else is given.

Let's say that we have m observations. That is, we have m transformations ${}^1_0\mathbf{T},\ {}^2_0\mathbf{T},\ \cdots,{}^m_0\mathbf{T}$, each being a 3-by-3 homogeneous transformation matrix (space is SE-2). For *each* observation, say that we observe the n 2D points ${}_0\mathbf{x}_1,\ {}_0\mathbf{x}_2,\ \cdots,\ {}_0\mathbf{x}_n$ in the particular observation (each point is a 3-by-1 homogeneous vector in $\mathbb{R}^2$). This smoothing problem becomes a **least squares optimization problem** (without any motion or sensor model).

Our goal is to find / smoothen the transforms and the map using just this data. This is done by minimizing a loss function, using a jacobian. The loss that we want to minimize is the error in point transformations across all observations. Precisely, the error is given by

$$L = \sum_{i=1}^{m}\sum_{j=1}^{n} \left\| {}_i\mathbf{x}_j - {}^i_0\mathbf{T}\,{}_0\mathbf{x}_j \right\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} \|\mathbf{r}_{ij}\|_2^2 \tag{4.1}$$

Where ${}_0\mathbf{x}_j$ is the map. We transform it to the ith frame and see the difference from what we actually observed. We want to minimize this error across all such transformations (across all observations). Each observation / transform has 3 variables associateed with it (x, y, θ in the SE-2 plane). Each point has two observations to optimize (the x, y value on map). Hence, the equation has $3m + 2n$ parameters to optimize. We ultimately have $2n$ equations over m frames, that is $2mn$ equations and $3m+2n$ parameters to optimize.

The function for point ${}_i\mathbf{x}_j$ (point j in frame / observation i) can be written as

$${}_i\mathbf{x}_j = \mathbf{f}_{ij}\left(\mathbf{R}_i, \mathbf{t}_i,\ {}_0\mathbf{x}_j\right) \tag{4.2}$$

Note that the residual and the above function can be written in terms of minimal parameters of $\xi = (x, y, \theta)$ instead of the transform $\mathbf{T}$ (else we'll be solving for $9m + 2n$ variables over a complex manifold). The residual vector is $\mathbf{r}_{ij}(\mathbf{T}(\xi_i), \mathbf{x}_j)$ and has the shape $2mn, 1$ (x, y equations for each of n point in each of m observations). Each residual has 5 parameters (3 for ξ and 2 for point). For shorthand, consider $\mathbf{T}_i = {}^i_0\,\mathbf{T}$. The Jacobian is given by

$$[\mathbf{J}_{i,j}]_{2,5} = \left[\underbrace{\left(\frac{\partial \mathbf{r}_{ij}}{\partial \mathbf{T}_i}\frac{\partial \mathbf{T}_i}{\partial \xi_i}\right)_{2,3}}_{\text{Localization}}\ \underbrace{\left(\frac{\partial \mathbf{r}_{ij}}{\partial \mathbf{x}_j}\right)_{2,2}}_{\text{Mapping}}\right]_{2,5} = [\mathbf{J}_{ij_L} \mid \mathbf{J}_{ij_M}] \tag{4.3}$$

Therefore the Jacobian for the whole problem will be of shape $3m + 2n, 2mn$ (since it's a derivative of residual w.r.t. all parameters).

$$\mathbf{J} = \left[\begin{array}{cccc||cccc}
(\mathbf{J}_{11_L})_{2,3} & 0_{2,3} & \cdots & 0_{2,3} & (\mathbf{J}_{11_M})_{2,2} & 0_{2,2} & \cdots & 0_{2,2} \\
(\mathbf{J}_{12_L})_{2,3} & 0_{2,3} & \cdots & 0_{2,3} & 0_{2,2} & (\mathbf{J}_{12_M})_{2,2} & \cdots & 0_{2,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
(\mathbf{J}_{1n_L})_{2,3} & 0_{2,3} & \cdots & 0_{2,3} & 0_{2,2} & 0_{2,2} & \cdots & (\mathbf{J}_{1n_M})_{2,2} \\
0_{2,3} & (\mathbf{J}_{21_L})_{2,3} & \cdots & 0_{2,3} & (\mathbf{J}_{21_M})_{2,2} & 0_{2,2} & \cdots & 0_{2,2} \\
0_{2,3} & (\mathbf{J}_{22_L})_{2,3} & \cdots & 0_{2,3} & 0_{2,2} & (\mathbf{J}_{22_M})_{2,2} & \cdots & 0_{2,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0_{2,3} & (\mathbf{J}_{2n_L})_{2,3} & \cdots & 0_{2,3} & 0_{2,2} & 0_{2,2} & \cdots & (\mathbf{J}_{2n_M})_{2,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0_{2,3} & 0_{2,3} & \cdots & (\mathbf{J}_{m1_L})_{2,3} & (\mathbf{J}_{m1_M})_{2,2} & 0_{2,2} & \cdots & 0_{2,2} \\
0_{2,3} & 0_{2,3} & \cdots & (\mathbf{J}_{m2_L})_{2,3} & 0_{2,2} & (\mathbf{J}_{m2_M})_{2,2} & \cdots & 0_{2,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0_{2,3} & 0_{2,3} & \cdots & (\mathbf{J}_{mn_L})_{2,3} & 0_{2,2} & 0_{2,2} & \cdots & (\mathbf{J}_{mn_M})_{2,2}
\end{array}\right]_{2mn,3m+2n} \tag{4.4}$$

Note that the $\|$ in the middle is just to separate out the localization ($3m$ columns in the left) and the mapping ($2m$ columns in the right). It is actually just a single matrix.

If we're given more constraints, like odometry, motion model or sensor model, those equations too can be added to the list of constraints, residuals for them too can be computed and then they too can be brought into the Jacobian. Even loop closure equations can be brought in that way.

Ultimately, our state updates can be given as

$$\underbrace{\delta\psi}_{3m+2n,1} = -\underbrace{\left(\mathbf{J}^\top\mathbf{\Omega}\mathbf{J}\right)^{-1}}_{3m+2n,3m+2n}\underbrace{\mathbf{J}^\top\mathbf{\Omega}^\top}_{3m+2n,2mn}\underbrace{\mathbf{r}}_{2mn,1} \tag{4.5}$$

Where $\delta\psi$ is the change to all localization and the mapping parameters (in the mentioned order). The new matrix $\mathbf{\Omega}$ is a diagonal matrix of shape $2mn, 2mn$. It is inverse of covariance of each equation / constraint. It can be through of as a confidence matrix (ith diagonal element shows the confidence of equation i).

4.2 Monocular SLAM

It is assumed that we have an algorithm that can extract features from images. Any corner detection algorithm that works on image gradients should also work. It is also assumed that there is no distortion in the images (we're using a simple pin-hole affine camera).

Projection Matrix

The primary thing needed for this is to calibrate the camera and at least obtain the projection. If any image has many points not in a plane, this can be done using DLT algorithm, as seen through equations 3.2 and 3.3. RANSAC can also be used to estimate the true parameters better (after removing trivial solutions as outliers). This will return $\mathbf{P} = \mathbf{KR}[\mathbf{I} \mid -\mathbf{X}_O]$. We then perform a substitution on this and obtain $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{X}_O$ as follows

$$\mathbf{P} = \mathbf{KR}\left[\mathbf{I} \mid -\mathbf{X}_O\right] = \left[\mathbf{KR} \mid -\mathbf{KR}\,\mathbf{X}_O\right] = \left[\mathbf{H} \mid \mathbf{h}\right]$$

$$\mathbf{X}_O = -\mathbf{H}^{-1}\mathbf{h} \qquad\qquad \mathbf{H}^{-1} = \left(\mathbf{KR}\right)^{-1} = \mathbf{R}^{-1}\,\mathbf{K}^{-1} = \mathbf{R}^\top\,\mathbf{K}^{-1} \tag{4.6}$$

The QR Decomposition of $\mathbf{H}^{-1}$ will give $\mathbf{H}^{-1} = QR$ where Q is an orthogonal matrix and R is an upper / right triangular matrix. Matching from above, we obtain the matrices $\mathbf{R} = Q^\top$ and $\mathbf{K} = R^{-1}$. In the end, we at least have $\mathbf{P}$ (the projection matrix) and if things go well, have $\mathbf{K}$ (camera intrinsics), $\mathbf{R}$, and $\mathbf{X}_O$ (camera extrinsics).

Bundle Adjustment

Instead of trying out anything we can set the whole thing as a *huge* optimization problem of residuals. This takes the outputs of the feature extraction algorithm.

Assuming that there are m camera poses and n points / pixels in each pose (with correspondences in the 3D scene). We need to estimate the m camera projection matrices $\mathbf{P}_i$ and n coordinates $\mathbf{X}_j$ (in $\mathbb{R}^3$), given those coordinates in the images (pixel coordinates). Each $\mathbf{P}$ is a $3, 4$ matrix (12 elements), each $\mathbf{X} = [X, Y, Z]$ is a point in 3D world and each $\mathbf{x} = [x, y]$ is a pixel in the image (pixel coordinates itself).

By converting the homographic equation to geometric (bringing the scaling parameter λ) and then scaling everything to unit scaling factor (eliminating the λ values), we get the following minimization problem

$$
\begin{aligned}
&\rightarrow L = \sum_{i=1}^{m}\sum_{j=1}^{n} \left\| \lambda_{ij}\mathbf{P}_i\mathbf{X}_j - \mathbf{x}_{ij} \right\|^2 \\
&\rightarrow \underset{\mathbf{X}_j,\mathbf{P}_i}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{j=1}^{n} \left(\left[\frac{\mathbf{P}_{i[1,:]}^{\top}\mathbf{X}_j}{\mathbf{P}_{i[3,:]}^{\top}\mathbf{X}_j} - x_{ij} \right]^2 + \left[\frac{\mathbf{P}_{i[2,:]}^{\top}\mathbf{X}_j}{\mathbf{P}_{i[3,:]}^{\top}\mathbf{X}_j} - y_{ij} \right]^2 \right) \\
&\rightarrow \underset{\mathbf{X}_j,\mathbf{P}_i}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{j=1}^{n} \left\| \frac{\left(\mathbf{P}_{i[1:2,:]}^{\top}\right)_{(2,4)} \left(\mathbf{X}_j\right)_{(4,1)}}{\left(\mathbf{P}_{i[3,:]}^{\top}\mathbf{X}_j\right)_{(1,1)}} - \left(\mathbf{x}_{ij}\right)_{(2,1)} \right\|_2^2 = \underset{\mathbf{X}_j,\mathbf{P}_i}{\operatorname{argmin}} \sum_{i=1}^{m}\sum_{j=1}^{n} \left\| \hat{\mathbf{x}}_{ij} - \mathbf{x}_{ij} \right\|_2^2
\end{aligned} \tag{4.7}
$$

Note that in the final optimization equation 4.7, the points $\mathbf{X}$ are in homogeneous coordinates (shape $4,1$ each) and the points $\mathbf{x}$ are pixel coordinates (shape $2,1$). Also, note that the residual is then actually a two element vector (for every i and j). One for x and another for y (in the image). We define the residual vector as

$$
\mathbf{r}_1 = \begin{bmatrix} \hat{\mathbf{x}}_{11} - \mathbf{x}_{11} = \hat{x}_{11} - x_{11} \\ \hat{y}_{11} - y_{11} \\ \hat{\mathbf{x}}_{12} - \mathbf{x}_{12} = \hat{x}_{12} - x_{12} \\ \hat{y}_{12} - y_{12} \\ \vdots \\ \hat{\mathbf{x}}_{1n} - \mathbf{x}_{1n} = \hat{x}_{1n} - x_{1n} \\ \hat{y}_{1n} - y_{1n} \end{bmatrix}_{2n,1} \qquad \mathbf{r}_2 = \begin{bmatrix} \hat{\mathbf{x}}_{21} - \mathbf{x}_{21} \\ \hat{\mathbf{x}}_{22} - \mathbf{x}_{22} \\ \vdots \\ \hat{\mathbf{x}}_{2n} - \mathbf{x}_{2n} \end{bmatrix}_{2n,1} \qquad \cdots \qquad \mathbf{r}_m = \begin{bmatrix} \hat{\mathbf{x}}_{m1} - \mathbf{x}_{m1} \\ \hat{\mathbf{x}}_{m2} - \mathbf{x}_{m2} \\ \vdots \\ \hat{\mathbf{x}}_{mn} - \mathbf{x}_{mn} \end{bmatrix}_{2n,1}
$$

The above equations give the residual vector for each image (there are m images). We combine them as a single long vector as

$$
\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_m \end{bmatrix}_{2nm} \tag{4.8}
$$

Note that the following short hand is used in the residual vector above

$$
\hat{\mathbf{x}}_{ij} = \begin{bmatrix} \hat{x}_{ij} = \frac{\mathbf{P}_{i[1,:]}^{\top}\mathbf{X}_j}{\mathbf{P}_{i[3,:]}^{\top}\mathbf{X}_j} \\ \hat{y}_{ij} = \frac{\mathbf{P}_{i[2,:]}^{\top}\mathbf{X}_j}{\mathbf{P}_{i[3,:]}^{\top}\mathbf{X}_j} \end{bmatrix}_{2,1} \qquad \mathbf{x}_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix}_{2,1}
$$

Now that the residual is defined in equation 4.8, we define the cost function as $F(\mathbf{k}) = \mathbf{r}^{\top}\mathbf{r}$, where $\mathbf{k}$ contains all the parameters that are needed to be optimized. These are the m projection matrices $\mathbf{P}_i$ (flattened, will be 12 elements each) and n 3D points in the scene (3 elements per point). Hence, there are $12m + 3n$ parameters that need to be optimized. Thus, the $\mathbf{k}$ vector is

$$
\mathbf{k}_{12m+3n} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \underbrace{(\mathbf{P}_i)_{1,12}}_{\mathbf{P}_{i[1,:]}^{\top},\mathbf{P}_{i[2,:]}^{\top},\mathbf{P}_{i[3,:]}^{\top}} & \cdots & \mathbf{P}_m & | & \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \underbrace{\mathbf{X}_j}_{X_j,Y_j,Z_j} & \cdots & \mathbf{X}_n \end{bmatrix}^{\top} \tag{4.9}
$$

Let's define some jacobians for short hand

$$
\begin{aligned}
\mathbf{J}_{ij_M} &= \frac{\partial \hat{x}_{ij}}{\partial \mathbf{P}_i} = \begin{bmatrix} \frac{\partial \hat{x}_{ij}}{\partial P_{i_{11}}} & \frac{\partial \hat{x}_{ij}}{\partial P_{i_{12}}} \cdots \frac{\partial \hat{x}_{ij}}{\partial P_{i_{34}}} \end{bmatrix}_{1,12} \qquad \mathbf{J}_{ij_S} = \frac{\partial \hat{x}_{ij}}{\partial \mathbf{X}_j} = \begin{bmatrix} \frac{\partial \hat{x}_{ij}}{\partial X_j} & \frac{\partial \hat{x}_{ij}}{\partial Y_j} & \frac{\partial \hat{x}_{ij}}{\partial Z_j} \end{bmatrix}_{1,3} \\
\mathbf{G}_{ij_M} &= \frac{\partial \hat{y}_{ij}}{\partial \mathbf{P}_i} = \begin{bmatrix} \frac{\partial \hat{y}_{ij}}{\partial P_{i_{11}}} & \frac{\partial \hat{y}_{ij}}{\partial P_{i_{12}}} \cdots \frac{\partial \hat{y}_{ij}}{\partial P_{i_{34}}} \end{bmatrix}_{1,12} \qquad \mathbf{G}_{ij_S} = \frac{\partial \hat{y}_{ij}}{\partial \mathbf{X}_j} = \begin{bmatrix} \frac{\partial \hat{y}_{ij}}{\partial X_j} & \frac{\partial \hat{y}_{ij}}{\partial Y_j} & \frac{\partial \hat{y}_{ij}}{\partial Z_j} \end{bmatrix}_{1,3}
\end{aligned} \tag{4.10}
$$

Note that the jacobians $\mathbf{J}_{ij_M}$ and $\mathbf{G}_{ij_M}$ are for *motion* (because they embed how the camera is moving in the scene) and that the jacobians $\mathbf{J}_{ij_S}$ and $\mathbf{G}_{ij_S}$ are for *structure* (because they embed how the scene should change). The jacobian is given by

$$\mathbf{J} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{k}}\right)_{2mn,12m+3n} = \left[\left\{\frac{\partial r_i}{\partial k_j}\right\}_{(\text{row}=i,\text{col}=j)}\right] \tag{4.11}$$

This is expanded as follows

$$\mathbf{J} = \left[\begin{array}{cccc||ccccc}
(\mathbf{J}_{11_M})_{1,12} & 0_{1,12} & \cdots & 0_{1,12} & \| & (\mathbf{J}_{11_S})_{1,3} & 0_{1,3} & \cdots & 0_{1,3} \\
(\mathbf{G}_{11_M})_{1,12} & 0_{1,12} & \cdots & 0_{1,12} & \| & (\mathbf{G}_{11_S})_{1,3} & 0_{1,3} & \cdots & 0_{1,3} \\
(\mathbf{J}_{12_M})_{1,12} & 0_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & (\mathbf{J}_{12_S})_{1,3} & \cdots & 0_{1,3} \\
(\mathbf{G}_{12_M})_{1,12} & 0_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & (\mathbf{G}_{12_S})_{1,3} & \cdots & 0_{1,3} \\
\vdots & \vdots & \vdots & \vdots & \| & \vdots & \vdots & \vdots & \vdots \\
(\mathbf{J}_{1n_M})_{1,12} & 0_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & 0_{1,3} & \cdots & (\mathbf{J}_{1n_S})_{1,3} \\
(\mathbf{G}_{1n_M})_{1,12} & 0_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & 0_{1,3} & \cdots & (\mathbf{G}_{1n_S})_{1,3} \\
0_{1,12} & (\mathbf{J}_{21_M})_{1,12} & \cdots & 0_{1,12} & \| & (\mathbf{J}_{21_S})_{1,3} & 0_{1,3} & \cdots & 0_{1,3} \\
0_{1,12} & (\mathbf{G}_{21_M})_{1,12} & \cdots & 0_{1,12} & \| & (\mathbf{G}_{21_S})_{1,3} & 0_{1,3} & \cdots & 0_{1,3} \\
0_{1,12} & (\mathbf{J}_{22_M})_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & (\mathbf{J}_{22_S})_{1,3} & \cdots & 0_{1,3} \\
0_{1,12} & (\mathbf{G}_{22_M})_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & (\mathbf{G}_{22_S})_{1,3} & \cdots & 0_{1,3} \\
\vdots & \vdots & \vdots & \vdots & \| & \vdots & \vdots & \vdots & \vdots \\
0_{1,12} & (\mathbf{J}_{2n_M})_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & 0_{1,3} & \cdots & (\mathbf{J}_{2n_S})_{1,3} \\
0_{1,12} & (\mathbf{G}_{2n_M})_{1,12} & \cdots & 0_{1,12} & \| & 0_{1,3} & 0_{1,3} & \cdots & (\mathbf{G}_{2n_S})_{1,3} \\
\vdots & \vdots & \vdots & \vdots & \| & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \| & \vdots & \vdots & \vdots & \vdots \\
0_{1,12} & 0_{1,12} & \cdots & (\mathbf{J}_{m1_M})_{1,12} & \| & (\mathbf{J}_{m1_S})_{1,3} & 0_{1,3} & \cdots & 0_{1,3} \\
0_{1,12} & 0_{1,12} & \cdots & (\mathbf{G}_{m1_M})_{1,12} & \| & (\mathbf{G}_{m1_S})_{1,3} & 0_{1,3} & \cdots & 0_{1,3} \\
0_{1,12} & 0_{1,12} & \cdots & (\mathbf{J}_{m2_M})_{1,12} & \| & 0_{1,3} & (\mathbf{J}_{m2_S})_{1,3} & \cdots & 0_{1,3} \\
0_{1,12} & 0_{1,12} & \cdots & (\mathbf{G}_{m2_M})_{1,12} & \| & 0_{1,3} & (\mathbf{G}_{m2_S})_{1,3} & \cdots & 0_{1,3} \\
\vdots & \vdots & \vdots & \vdots & \| & \vdots & \vdots & \vdots & \vdots \\
0_{1,12} & 0_{1,12} & \cdots & (\mathbf{J}_{mn_M})_{1,12} & \| & 0_{1,3} & 0_{1,3} & \cdots & (\mathbf{J}_{mn_S})_{1,3} \\
0_{1,12} & 0_{1,12} & \cdots & (\mathbf{G}_{mn_M})_{1,12} & \| & 0_{1,3} & 0_{1,3} & \cdots & (\mathbf{G}_{mn_S})_{1,3}
\end{array}\right] \tag{4.12}$$

The jacobian shown above has $2mn$ rows ($2n$ rows, goes m times; this is purely because we constructed $\mathbf{r}$ that way). It has $12m + 3n$ columns. The first $12m$ columns are before the $\|$ and are for the motion part (estimating $\mathbf{P}$, from which we'll get our camera poses) and the remaining $3n$ columns are for the structure part (estimating $\mathbf{X}$ from which we'll get the environment's map).

Optimization The optimization of this could (theoretically) be done using an LM optimizer, with the update equations being

$$\begin{aligned}
\delta\mathbf{k} &= -\underbrace{\left(\mathbf{J}^\top\mathbf{J} + \lambda\mathbf{I}\right)^{-1}}_{(12m+3n,12m+3n)} \underbrace{\mathbf{J}^\top}_{(12m+3n,2mn)} \underbrace{\mathbf{r}}_{(2mn,1)} \\
\mathbf{k} &\leftarrow \mathbf{k} + \delta\mathbf{k}
\end{aligned} \tag{4.13}$$

However, if the values of m and n are large, this is best left to a *sparse optimization library* (sparse because most elements in $\mathbf{J}$ are zeros). Notice how only two paramters in each line are non-zero (every row has only 15 non-zero entries), we can optimize by operating on these values only (it'll give storage and computational advantages). Also, the first term of $\delta\mathbf{k}$ will also be mostly sparse (there are many decomposition methods that give direct inverse for such sparse matrices). However, we have to provide initial estimates for the optimization.

Initial Estimates Usually, estimates from sensors such as GPS, Odometry and IMU give the approximate robot location. Using triangulation, one can get the initial guesses for the 3D points. The projection

matrix can be guessed using the DLT method (if applicable), else if the cameras are calibrated, then we only need to estimate the camera pose to get initial estimates (which our sensors would directly give).

However, if there are no other sensors, then one could estimate $\mathbf{F}$ through the first set of images (using the 8-point algorithm). Initial estimates can be guessed by considering changes in the successive frames only, and by performing certain decomposition procedures on $\mathbf{F}$ (that allow us to extract rough estimates of $\mathbf{K}$, and incremental $\mathbf{b}$ and $\mathbf{R}$). However, we will not escape the issue of scale this way (the entire scene and setup can be scaled and there will be no difference in the results). We will get solution upto a similarity transform. Certain scale constraints can be applied to overcome such issues (we'll need 7 constraints). An approximated baseline length could yield better results for the starting point.

Earlier, a method to estimate $\mathbf{P}$ using RANSAC was discussed. If that works well, we could use triangulation (see where the rays approximately intersect in the real world) and use those estimates as the starting points. This is probably our best shot (without any other sensor data).

However, all these vision techniques for initial guess will fail if the correspondences are not found or when there is very high motion blur.