Structure From Motion Project

Phase I

April 9, 2021

Introduction

The goal of this document is to provide the mathematical and theoretical underpinnings of structure from motion.

1. Camera Calibration

Intrinsic calibration matrix (K) is required to extract the essential matrix from the fundamental matrix. K is calculated by performing camera calibration on a set of images on a chess board design as shown in Figure 1.

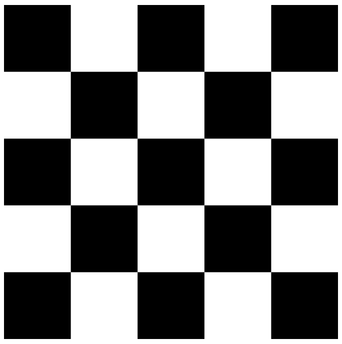

Figure 1: A *5x5* chess board pattern with *4x4* vertices. Transitions lines between black and white blocks are pixels with high local gradients.

1.1. Focal Length

Pixel 3a XL camera was used to capture images of a chess board

1.2. Principle Axis

Another

2. Feature Detection

Features are image properties that are invariant under a change of perspectives.

2.1. SIFT

The goal of SIFT (Scale Invariant Feature Transformation) is to extract distinctive invariant features. These features must be correctly matched against a large database of features from many images. Furthermore, we seek an extraction method invariant to image scale and rotation. Lastly, it needs to be robust to noise, change in viewpoint, illumination, and affine distortion.

SIFT searches for **keypoints** and **descriptors** in an image. Keypoints are points of interest where local changes occur in an image. Take the chess board in Figure 1 as an example. Lines that demarcate transitions between black and white blocks would be good keypoint candidates since they are distinct. Even more so, the vertices have gradients in multiple directions making them good candidates for invariant points.

SIFT utilises a *difference of Gaussian*(DoG) method. This method generates Gaussian blurred version of an original image by convolving the image, $f(x, y)$ with a Gaussian kernel $G_\sigma(x, y)$. Here, σ represents the spread of the Gaussian as well as the width of the kernel as given by

$$G_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(- \frac{x^2+y^2}{2\sigma^2}\Big)$$

We can then convolve this kernel with the image and get a blurred image given by

$$g(x,y) = G_\sigma(x,y) * f(x,y)$$

If we take two copies of an image and blur these copies $f_1(x, y)$ and $f_2(x, y)$ with the different kernel widths σ_1 and σ_2 we get

$$\begin{aligned} g_{\sigma 1} &= G_{\sigma 1}(x,y) * f_1(x,y) \\ g_{\sigma 2} &= G_{\sigma 2}(x,y) * f_2(x,y) \end{aligned}$$

The DoG method then looks at the difference between $g_1(x, y)$ and $g_2(x, y)$ and determines the extrema

$$\operatorname*{argmax}_{x,y} \left(g_1(x,y) - g_2(x,y)\right)$$

This generates high frequency responses at points where the difference in blurring is maximized. In this manner, the DoG method acts as a band-pass filter. The *descriptor* is given by a vector of dominant gradients near the keypoint. As shown in Figure 2, DoG produces local gradients in the image which are then weighted by a Gaussian window. Therefore, a keypoint descriptor is just a histogram in polar coordinates of local gradients.

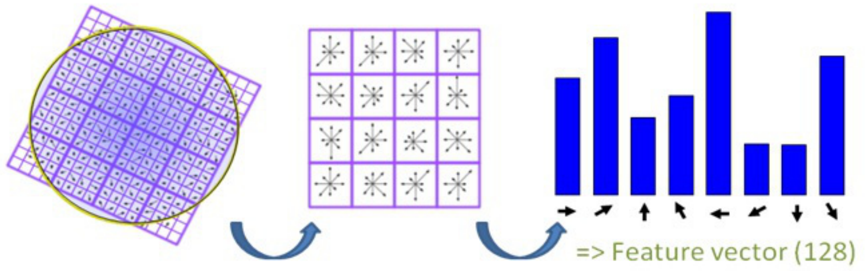

Figure 2: A keypoint descriptor is generated by accumulating local gradients into histograms for gradient directions. This histogram is represented by a feature vector. In this example, 128 directions are defined.

2.2. Descriptor Matching

SIFT can be utilized on multiple images to form a database of images with feature descriptors. It is then useful to combine images to find common features. One approach to do this is by the *k-nearest neighbours* approach, also called k-NN. In the k-NN algorithm, one is interested in finding the nearest feature descriptor vectors from one image to another. This is usually performed by taking the Euclidean distance between the two vectors in feature space. It is intuitive then that keypoints having unique histogram distributions allows for better matching. Uniqueness in this context can be quantified for example by taking the ratio of distance between the first to the second nearest neighbour. If the ratio is large then that implies feature vectors between images have unique features. According to D. Lowe in his pioneering paper on SIFT, more than 90% of false matches and less than 5% of correct matches are discarded by rejecting distance ratios greater than 0.8 making it an ideal cut-off. For this project, we use 0.75 as the maximum cut-off for additional denoising.

Other points of note on SIFT and Feature Matching:

- Having many outliers will affect the accuracy of the matching process. E.g. $\geq 20\%$ outliers
- Two most common matchers are Brute Force(BF) and Fast Library for Approximate Nearest Neighbours (FLANN). This project uses BF matcher as it is slower but exhaustive.
- See here for more details.

3. Fundamental Matrix

To begin with understanding the fundamental matrix, recall that in epipolar geometry two camera centres **C** and **C**$'$ form an epipolar plane π with the 3-space point **X**. As shown in Figure 3, π intersects with both cameras' image planes denoted by **l** and **l**$'$ - the epipolar lines. Furthermore, the camera baseline (line between the two camera centres) intersects each image plane at the epipoles **e**. The location of these epipoles are conversed regardless of the location of **X**. The point **x** must lie on the epipolar line **l'**. Note, the epipolar line does not inherently contain information about the location of the epipole. That is to say, knowing one epipolar line does not tell us the location of camera **C**$'$ in the perspective of **C**. However, the epipole can be found by checking where multiple epipolar lines converge.

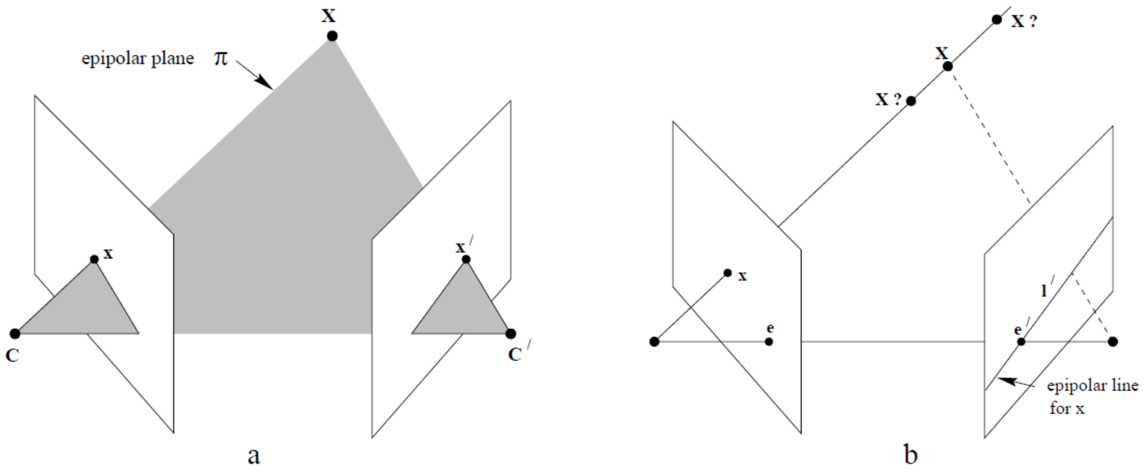

Figure 3: Point correspondence geometry. (a) The two cameras are indicated by their centres **C** and **C**$'$ image planes. The camera centres, 3-space point **X**, and its images **x** and **x**$'$ lie in a common plane π. (b) This ray is imaged as a line **l**$'$ in the second view. The 3-space point **X** which projects to **x** must lie on this ray, so the image of **X** in the second view must lie on **l**$'$.

Furthermore, for a line in space given by

$$ax + by + c = 0 \Rightarrow \boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

if the point **x**$'$ corresponding to **x** is on the epipolar line **l**$'$ then the following must hold true

$$x^T l = 0$$

This is simply because **l**$'$ lies on the image plane and **x** is perpendicular to the image plane as shown in Figure 4 below. Therefore, the inner product of **x** and **l** must equal zero.

In order to find a mapping of **x** onto **l**$'$, we first need to make a guess for a corresponding **x**$'$ in the second image frame. This was already done with SIFT and feature matching as described

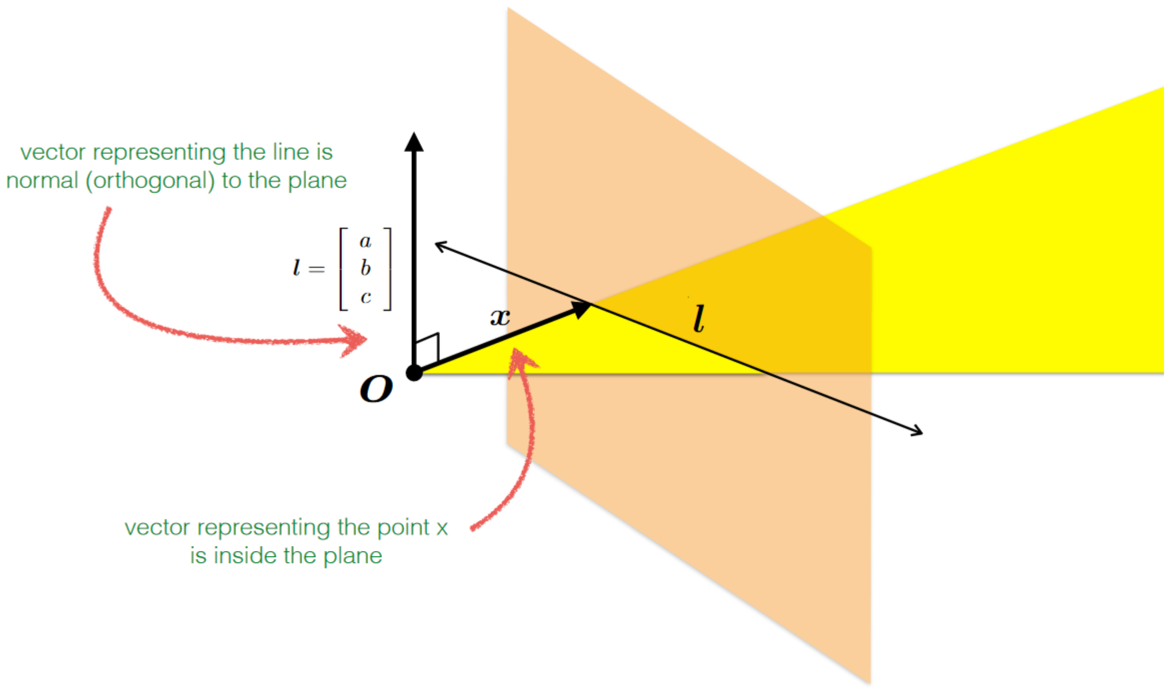
vector representing the line is normal (orthogonal) to the plane
$l = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$
x
l
O
vector representing the point x is inside the plane

Figure 4: Orthogonality between $\mathbf{x}$ and $\mathbf{l}$ are shown. The orange plane is the image plane for the camera and the yellow plane represents π, the epipolar plane.

above. As a recap, SIFT generates feature vectors containing a directional gradient histogram for every keypoint. These keypoint feature vectors are then compared between images using L2 norm and matches are found. The fundamental matrix F takes the ray going from $\mathbf{C}$ to $\mathbf{X}$ and translates that ray along the baseline to $\mathbf{C}'$. The fundamental matrix also rotates the ray within the epipolar plane π until it passes through $\mathbf{X}$. Of course, if the two cameras at $\mathbf{C}$ and $\mathbf{C}'$ were calibrated, we can extract the pose information directly from the fundamental matrix. However, since we will assume they are not calibrated, the fundamental matrix must operated on by the inverse of the intrinsic matrix in order to recover pose information.

Therefore, we now have two important pieces of information. The first is that $x^T l = 0$. The second thing we know is that F is the transformation that projects the ray from $\mathbf{Cx}$ to the epipolar line $\mathbf{l}'$ on the image plane of $\mathbf{C}'$. In other words, this means $\boldsymbol{F}x = l'$. Therefore, $x'^T \boldsymbol{F} x = 0$. This is called the *epipolar constraint.*

After obtaining point correspondences, we can relate the fundamental matrix to the point correspondences by the epipolar constraint as such

$$x'^T \boldsymbol{F} x = 0$$
$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Since F has eight unknowns (we impose $F_{33} = 1$), we need at at least eight point correspondences between two images from C and $\mathbf{C}'$ to fully define F. This is done via singular value decomposition (SVD) and Random Sampling Consensus (RANSAC). SVD uses eight independent point correspondences between two images to convert the $x'^T \boldsymbol{F} x = 0$ problem into a least squares problem of $AX = 0$ (Appendix A.3). RANSAC acts to clean up outliers by looking at the reprojection of a calculated F. For more information, see these slides.

Note what happens to F when camera centres are close to one another. The epipolar lines $\mathbf{l}$ and $\mathbf{l}'$ shrink until they are zero. This creates a rank deficient fundamental matrix.

Finally, a note on the relationship between homography matrix and fundamental matrix. Where the homography matrix deals with point to point mapping, the fundamental matrix deals with point to line mapping. Furthermore, homography matrices require planes mapped to planes, whereas fundamental matrices do not have this additional constraint. Therefore, the fundamental matrix is a generalized homography matrix.

4. Essential Matrix

The essential matrix is a 3×3 matrix that encodes epipolar geometry. In fact, the fundamental matrix is simply a generalization of the essential matrix when cameras are uncalibrated. A metric reconstruction of the scene may be computed using the essential matrix when the cameras are calibrated. When calculating the fundamental matrix, we made no assumptions about the focal length or principle axis of the camera. Because we can now assume that the essential matrix contains information for transforming from one homogeneous space to another independent of camera parameters, we can decompose $\mathbf{E}$ as the compound operation of translating and rotating points from x' to x. If we consider the combination of rotation and translation as poses $\mathbf{P}_1$ and $\mathbf{P}_2$, the essential matrix tells us how to obtain any camera pose $\mathbf{P}_1$ from a reference pose $\mathbf{P}_2$ (Figure 5).

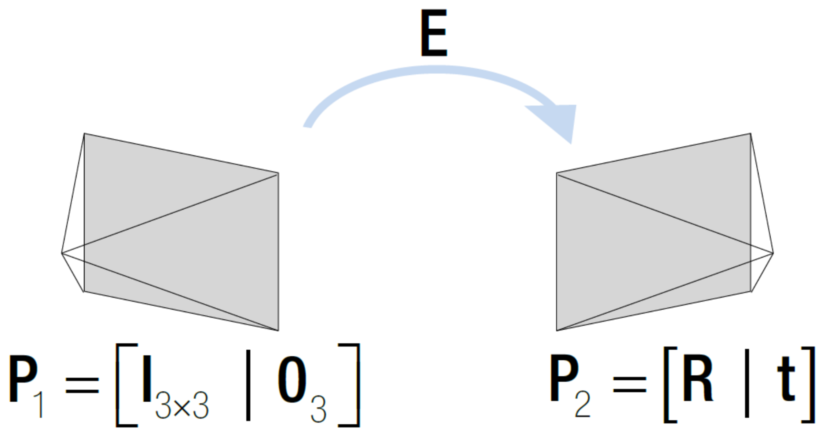

Figure 5: Similar to the fundamental matrix, the essential matrix encodes epipolar geometry. It acts as an instruction set for transformation from some reference pose ($\mathbf{P}_1$) to a desired pose ($\mathbf{P}_2$).

Let us consider how this is done. Starting with a 3D point in camera 1 ($\vec{X}_1$), we seek the corresponding 3D point in camera 2 ($\vec{X}_2$). We know that in order to get this, we have to rotate point $\vec{X}_1$ and translate it across the baseline as we did for the fundamental matrix. We write this operation as

$$\vec{X}_2 = \mathbf{R}\vec{X}_1 + \vec{t}$$

If we take the $\vec{t}$ as the vector from camera 2 to camera 1, then $\vec{t}$ and $\vec{X}_2$ are coplanar. Given that is the case, we can say the normal vector to the epipolar plane is

$$\vec{n} = \vec{t} \times \vec{X}_2 = [\mathbf{t}]_\times \vec{X}_2$$

Here, $\mathbf{t}_\times$ is a skew-symmetric matrix. Recall, a skew-symmetric matrix for a vector $\vec{t} \in \mathbb{R}^3$ is given by

$$\mathbf{t}_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

Using the relation above for $\vec{X}_2$ and $\vec{X}_1$, we can take the dot product of the normal vector and $(\vec{X}_2 - t)^T$ to get

$$\begin{aligned}0 &= (\vec{X}_2 - t)^T \vec{n} \\ 0 &= (\vec{X}_2 - t)^T [\mathbf{t}]_\times \vec{X}_2\end{aligned}$$

Note, we obtain zero as the dot product since the vector $(\vec{X}_2 - t)^T$ lies on the epipolar plane and $\vec{n}$ is normal to the plane. Now substituting, we obtain

$$\begin{aligned}0 &= (\mathbf{R}\vec{X}_1)^T [\mathbf{t}]_\times \vec{X}_2 \\ 0 &= \vec{X}_1^T \mathbf{R}^T [\mathbf{t}]_\times \vec{X}_2 \\ 0 &= \vec{X}_2^T [\mathbf{t}]_\times \mathbf{R} \vec{X}_1 \\ 0 &= \vec{X}_2^T \mathbf{E} \vec{X}_1\end{aligned}$$

Where $\mathbf{E} = \mathbf{t}_\times \mathbf{R}$. We have now derived the essential matrix from simply demanding that $\vec{X}_2$ be the compound operations of rotation and translations on $\vec{X}_1$!

Using the essential matrix as is requires the assumption that our cameras are calibrated. We know this to be untrue as we have found the intrinsic matrix earlier and it did not equal to the identity matrix. However, given that we have our cameras calibrated and received an intrinsic matrix $\mathbf{K}$, we can now obtain the essential matrix from the fundamental matrix. First, recall that using the intrinsic matrix $\mathbf{K}$, we can transform $\vec{X}_2$ and $\vec{X}_1$ from the camera coordinate system to $\vec{x}_2$ and $\vec{x}_1$ in the image coordinate system like so

$$\vec{x}_1 = \mathbf{K}\vec{X}_1 \text{ and } \vec{x}_2 = \mathbf{K}\vec{X}_2$$

Then,

$$\begin{aligned}\vec{X}_2^T \mathbf{E} \vec{X}_1 &= 0 \\ \vec{x}_2^T \mathbf{K}^{-T} \mathbf{E} \mathbf{K}^{-1} \vec{x}_1 &= 0 \\ \vec{x}_2^T \mathbf{F} \vec{x}_1 &= 0\end{aligned}$$

Notice, $\mathbf{F} = \mathbf{K}^{-1}\mathbf{E}\mathbf{K}^{-1}$. Therefore $\mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K}$. Great! However, the question still remains, how do we decompose the essential matrix into its rotation and translation components $\mathbf{P}_2$ as shown in Figure 5? Specifically, we want

$$\mathbf{E} = [\mathbf{R}|\mathbf{t}] = \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix}$$

To begin, let us start by computing the epipole in Figure 6 below from Mike's point of view. Notice from Figure 6 that regardless of the choice of $\vec{X}_1$, we get a corresponding line in image 2 from Mike's point of view that passes through the epipole. We can denote this by choosing an arbitrary $\vec{X}_1 = [\mathbf{1}]$ to get $\vec{X}_2^T \mathbf{E} = 0$.

We can see that if we know the epipole, we can find the translation $\vec{t}$ between the two views. By convention, we give the world coordinate center (in homogeneous coordinates) $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$ to the first view. Applying the essential matrix to this point should give us the epipole in the second view. Indeed,

$$\mathbf{P}_2 \begin{bmatrix}\mathbf{0} \\ 1\end{bmatrix} = [\mathbf{R}|\mathbf{t}] \begin{bmatrix}\mathbf{0} \\ 1\end{bmatrix} = \mathbf{t}$$

This becomes the epipole in Mike's point of view shown in Figure 6. Using $\vec{X}_2^T \mathbf{E} = 0$ we obtained above, we can see that $\vec{t}^T \mathbf{E} = 0$. That is to say, the left nullspace of the essential matrix is the epipole in image 2.

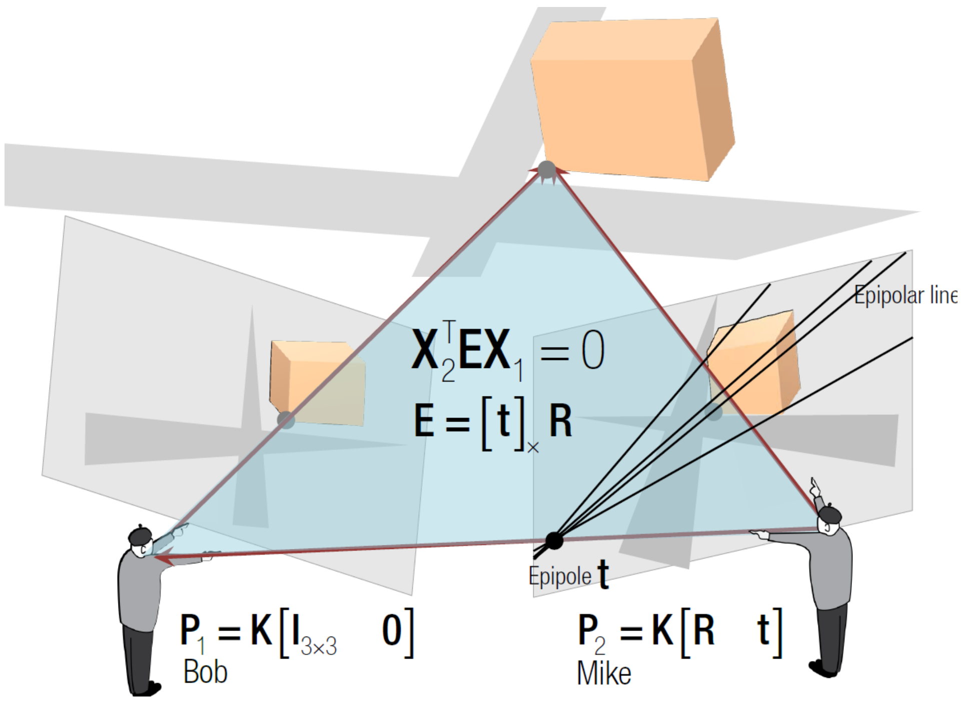

Figure 6: Epipole in image 2 is given by $\mathbf{t}$. Recall that in homogeneous coordinates, points on a plane such as image 2 from Mike's point of view can be represented by a line from the camera to that point.

We now have a linear least squares problem. Similar to our approach to the fundamental matrix, we can take the singular value decomposition of the essential matrix to get

$$\mathbf{E} = \mathbf{U} \begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0\end{bmatrix} \mathbf{V}^T$$

We constrain the rank of $\mathbf{E} = 2$ by setting the last column of the singular matrix to zero to account for the condition $rank(A) = min(m, n)$ for some $m \times n$ matrix A. In practice, $\mathbf{E}$ is an overdetermined system with two independent equations. While out of the scope of this documentation, it can be shown that the last row of the rotation matrix is the linear combination of the first two rows.

Now that we have $\mathbf{U} = \begin{bmatrix} | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \\ | & | & | \end{bmatrix}$, $\mathbf{t} = \mathbf{u}_3$ or $-\mathbf{u}_3$ due to the scale ambiguity of homogeneous coordinates. We can use the definition of a skew-symmetric matrix and the relation between $\vec{t}$ and $\mathbf{R}$ to obtain the following

$$\mathbf{E} = \mathbf{U}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{V}^T = [\mathbf{t}]_\times \mathbf{R} = \left(\mathbf{U}\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{U}^T\right)\mathbf{U}\mathbf{Y}\mathbf{V}^T$$

Here, we represented $\mathbf{R} = \left(\mathbf{U}\mathbf{Y}\mathbf{V}^T\right)$ using SVD on the rotation matrix and $[\mathbf{t}]_\times = \mathbf{U}\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{U}^T$ using the definition of the skew-symmetric matrix and the relationship between $\mathbf{U}$ and $\vec{t}$.

Next, we can

$$\mathbf{E} = \mathbf{U}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{V}^T = [\mathbf{t}]_\times \mathbf{R} = \mathbf{U}\begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{Y}\mathbf{V}^T$$

Notice that $\mathbf{U}\mathbf{U}^T$ vanishes to the identity matrix since $\mathbf{U}$ is a unitary matrix - meaning $\mathbf{U}^T = \mathbf{U}^{-1}$. Now we can see from the above equivalence relation that

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{Y}$$

$\mathbf{Y}$ can take on two possible solutions for the above to hold,

$$\mathbf{Y} = \begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, \text{or } \begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}^T$$

Accordingly, $\mathbf{R}$ can take on the values $\mathbf{R} = \mathbf{U}\mathbf{Y}\mathbf{V}^T$ or $\mathbf{U}\mathbf{Y}\mathbf{V}^T$. Therefore, we now have four possible configurations for rotation and translation (Figure 7).

$$\mathbf{P}_2 = [\mathbf{U}\mathbf{Y}\mathbf{V}^T|\mathbf{u}_3],\ [\mathbf{U}\mathbf{Y}^T\mathbf{V}^T|\mathbf{u}_3],\ [\mathbf{U}\mathbf{Y}\mathbf{V}^T|-\mathbf{u}_3],\ [\mathbf{U}\mathbf{Y}^T\mathbf{V}^T|-\mathbf{u}_3]$$

How do we determine the correct configuration? We can triangulate 3D points from our 2D points and measure the reprojection error. The reprojection error is obtained by projecting a 3D point back into an image and measuring the difference between the observed 2D point and the actual 2D point. More on triangulation and reprojection error in the next section. For now, if we treat triangulation as a function, the rotation and translation configuration that gives us the minimal reprojection error is the correct one.

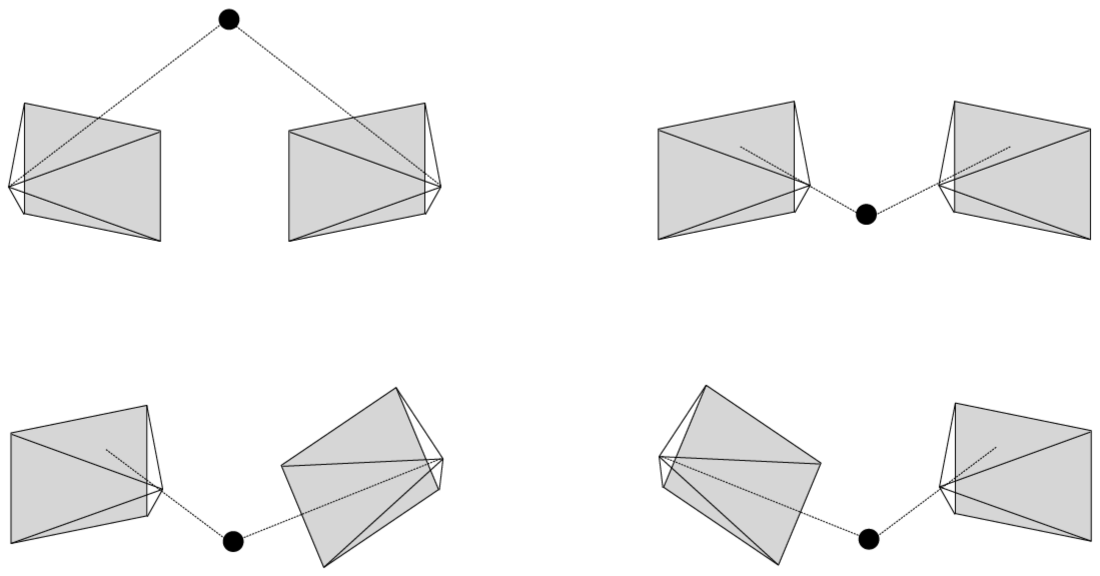

Figure 7: Four camera configurations are given by singular value decomposition of the essential matrix. The correct configuration can be resolved via point triangulation.

5. Point Triangulation

Point triangulation is the idea that we can take two rays (or points in homogeneous coordinates) from matching points in two views and obtain a third 3D point in world coordinates (Figure 8). The two matching keypoints in two views and the 3D point form a triangle on the epipolar plane - hence the name point triangulation. How can we solve triangulation mathematically? Specifically, what we seek is the following:

Given a set of noisy matched points $\{x_i, x'_i\}$ and camera matrices $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ and $\mathbf{P}'$, estimate the 3D object point $\mathbf{X}$. Notice that by posing our problem in this way, we are assuming knowledge of pose information is known. In order to establish a baseline (for the first two views), we will test all four pose candidates derived in the previous section (Section 4). For every subsequent view, a perspective n-point algorithm will be used to obtain pose information as described in the next section (Section 6).

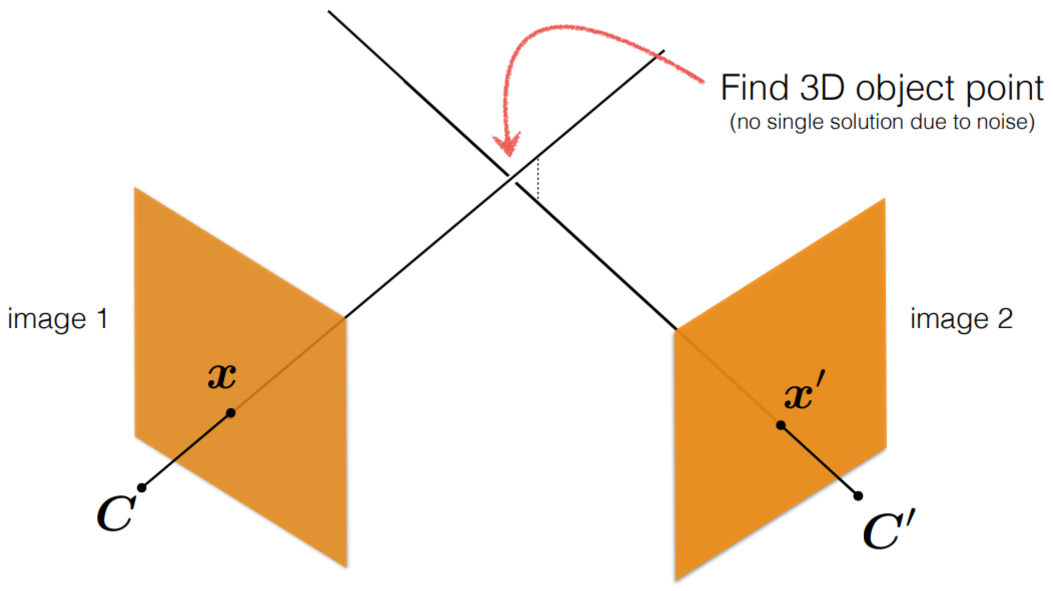

Figure 8: Point triangulation performed by intersection of two homoegeneous points from cameras $\mathbf{C}$ and $\mathbf{C}'$. Note that points x and x' correspond to the same feature point. Rays may not meet perfectly due to noise.

Note, that while we cannot find $\mathbf{X}$ from a single homogeneous point x_i, we can theoretically find it using two correspondences x and x'. However, in practice this is difficult to do in the presence of noise. The two pixels may not point to the exact same object location and therefore there would be no solution to this intersection of rays problem. We can still get *some* solution if we assume the object point to be at the lowest distance between the two rays. This should trigger one's intuition that some form of total least squares will be used for the triangulation process.

Let us begin our solution with the obvious relation between the camera point and the 3D world point as a transformation of the pose matrix

$$\vec{x} = \alpha \mathbf{P} \vec{X}$$

Where α is some scale factor to convert from homogeneous to inhomogenous coordinates. Expanding out the above relation we get

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This is a similarity relation -meaning a direct linear transform can be used to solve this system. We need to first remove the scale factor, convert to a linear system as we did with the fundamental matrix and then solve using SVD (Appendix A.1, A.3). Taking the cross-product with $\vec{x}$ from both sides,

$$\vec{0} = \alpha \left(\vec{x} \times \mathbf{P}\vec{X} \right)$$
$$\vec{0} = \vec{x} \times \mathbf{P}\vec{X}$$

Using the following identities of vector cross-product

$$\vec{a} \times \vec{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} \quad \text{and} \quad \vec{a} \times \vec{a} = 0$$

We can perform the following operations

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} — & \vec{p}_1^T & — \\ — & \vec{p}_2^T & — \\ — & \vec{p}_3^T & — \end{bmatrix} \begin{bmatrix} | \\ X \\ | \end{bmatrix}$$
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} \vec{p}_1^T\vec{X} \\ \vec{p}_2^T\vec{X} \\ \vec{p}_3^T\vec{X} \end{bmatrix}$$

Using $\vec{0} = \vec{x} \times \mathbf{P}\vec{X}$ obtained earlier, we eliminate the scale factor α

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \vec{p}_1^T\vec{X} \\ \vec{p}_2^T\vec{X} \\ \vec{p}_3^T\vec{X} \end{bmatrix} = \begin{bmatrix} y\vec{p}_3^T\vec{X} - \vec{p}_2^T\vec{X} \\ \vec{p}_1^T\vec{X} - x\vec{p}_3^T\vec{X} \\ x\vec{p}_2^T\vec{X} - y\vec{p}_1^T\vec{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Note that the third line in the final matrix above is a linear combination of the first two (x times the first line ply y times the second line). This gives us the rank 2 matrix

$$\begin{bmatrix} y\vec{p}_3^T\vec{X} - \vec{p}_2^T\vec{X} \\ \vec{p}_1^T\vec{X} - x\vec{p}_3^T\vec{X} \end{bmatrix} \vec{X} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

However, $\vec{X}$ is a 4-dimensional vector in homogeneous coordinates. This means we need two more equations to fully define the solution with no free variables. We can use corresponding 2D points from x' to fully define the system. Finally, we get

$$\begin{bmatrix} y\vec{p}_3^T\vec{X} - \vec{p}_2^T\vec{X} \\ \vec{p}_1^T\vec{X} - x\vec{p}_3^T\vec{X} \\ y'\vec{p'}_3^T\vec{X} - \vec{p'}_2^T\vec{X} \\ \vec{p'}_1^T\vec{X} - x'\vec{p'}_3^T\vec{X} \end{bmatrix} \vec{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

which is in the form of $\mathbf{AX} = \mathbf{0}$. Therefore, we can solve for $\vec{X}$ using SVD. Note that due to some error in targeting the same point, the problem can also be posed as $\mathbf{AX} = \mathbf{b}$ which accounts for some algebraic error. This system is non-linear and more difficult to solve but better reflect reality.

Lastly, it is possible to better triangulate a point using more than two cameras targeting the same 3D point as shown in Figure 9 below. We would simply end up with an over-constrained system shown below which can also be solved with SVD

$$\begin{bmatrix} \begin{bmatrix} \vec{x}_1 \\ 1 \end{bmatrix}_\times \mathbf{P}_1 \\ \begin{bmatrix} \vec{x}_2 \\ 1 \end{bmatrix}_\times \mathbf{P}_2 \\ \vdots \begin{bmatrix} \vec{x}_F \\ 1 \end{bmatrix}_\times \mathbf{P}_F \end{bmatrix} \begin{bmatrix} \vec{X} \\ 1 \end{bmatrix} = \vec{0}$$

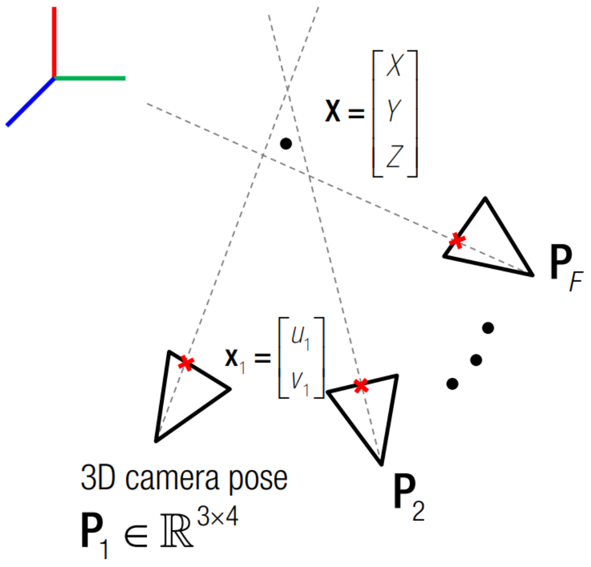
$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
$\mathbf{x}_1 = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix}$
$\mathbf{P}_F$
$\mathbf{P}_2$
3D camera pose
$\mathbf{P}_1 \in \mathbb{R}^{3\times4}$

Figure 9: Point triangulation performed by intersection of multiple homogeneous points from cameras $\mathbf{P}_1$ to $\mathbf{P_F}$. Similar to the two point correspondence scenario, noise in x and $\mathbf{P}$ means that rays corresponding to a single 3D point on the object ($\mathbf{X}$) will not necessarily intersect nicely.

Reprojection Error

Now that we can triangulate a 3D point from an image pair, we can return to disambiguation of the essential matrix poses. Recall that were four candidate poses - in essence, four combinations of rotation and translation that make up a unique $\mathbf{P}_2$- that when combined would return the essential matrix. In order to figure out the correct pose, we can disambiguate by triangulating points using each of the four $\mathbf{P}_2$ and select the one with the lowest reprojection error.

Reprojection error is the geometric error between a projected point and a measured one. More specifically, it looks at the difference between what pixel $\vec{\hat{x}}$ a 3D point $\vec{X}$ is projected to using $\vec{\hat{x}} = \mathbf{P}\vec{X}$ and the actual 2D point/pixel $\vec{x}$ that was used to triangulate that 3D point. Mathematically, for a pair of images with n triangulated 3D points, it can be written as

$$\sum_{i}^{n} d(\vec{x}_i, \vec{\hat{x}}_i)^2 + d(\vec{x}'_i, \vec{\hat{x}}'_i)^2$$

Here, $d(x, y)$ represents euclidean distance.

6. Perspective-n-Point

Let us summarize succinctly what we have done thus far:

1. We took images of a checker-board print with our camera and used the relation between 3D points and corresponding 2D points in our images to produce a robust camera intrinsic matrix **K**.

2. We then used SIFT to find invariant points (called keypoints) in our object image pairs and generated feature descriptors corresponding to those points. These feature descriptors were matched using a brute force matcher for each image pair to 'sift' for those with similar descriptors.

3. Upon finding these matching points between image pairs we calculated the fundamental matrix between two images which formed the baseline.

4. Using this fundamental matrix, we calculated the essential matrix which contains pose information (rotation and translation of one camera w.r.t. another). We have shown that in order to extract pose information from the essential matrix, we need to disambiguate it using triangulation.

5. We performed triangulation of 3D points using pose information obtained from the essential matrix and obtained 3D points for the baseline images as well as the correct pose for the second camera. We performed the latter pose disambiguation by minimizing the reprojection error.

The question now remains, how can we integrate more images? After all, while two images may be sufficient for reconstruction of a planar object, our goal is to reconstruct a three-dimensional object. This naturally requires integrating images from all angles with the at least the minimum possible coverage of the object. The approach to this question is called the perspective-n-point problem.

The perspective-n-point

Appendix A: Selected Mathematical Review

A.1. Singular Value Decomposition (SVD)

Singular value decomposition aims to decompose any $m \times n$ matrix $\mathbf{A}$ into its components $\mathbf{U\Sigma V}^T$. This section will not derive the SVD method but will show an example of implementation and how it can be used to solve a homogeneous linear system as required by many computer vision and AI problems. Let us start with an example of matrix

$$\mathbf{A} = \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} = \mathbf{U\Sigma V}^T$$

Here, $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices. Unitary matrices are transformations that preserve the relative angles of eigenvectors as well as their length. In essence, unitary transformations rotate vectors such that $\langle x, y\rangle = \langle \mathbf{U}x, \mathbf{U}y\rangle \quad \forall\, x, y \in \mathbb{R}^m$. Furthermore, unitary matrices possess the property that $\mathbf{U}^T = \mathbf{U}^{-1}$ - therefore, $\mathbf{UU}^T = \mathbf{I}_{m\times m}$. This property can also be generalized to complex valued matrices and their conjugate transpose given by $\mathbf{UU}^\dagger = \mathbf{I}_{m\times m}$

Note then that, the following properties hold

$$\mathbf{A}^T\mathbf{A} = \mathbf{V\Sigma}^T\mathbf{U}^T\mathbf{U\Sigma V}^T = \mathbf{V\Sigma}^T\mathbf{\Sigma V}^T \tag{1}$$
$$\mathbf{AV} = \mathbf{U\Sigma} \tag{2}$$

Starting with (1), we get

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 5 & -1 \\ 5 & 7 \end{bmatrix}\begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} = \begin{bmatrix} 26 & 18 \\ 18 & 74 \end{bmatrix}$$

Notice that this matrix is diagonalizable and that (1) implies we have a matrix we can write as $\mathbf{A} = \mathbf{PDP}^{-1}$ with the product of our singular matrix and its transpose equal to the diagonal matrix $\mathbf{D} = \mathbf{\Sigma}^T\mathbf{\Sigma}$. Furthermore, since $\mathbf{V}$ is unitary, we can say that (1) gives us the diagonalization of $\mathbf{A}^T\mathbf{A}$ assuming $\mathbf{A}$ has m unique eigenvalues. These eigenvalues will then be the singular values of $\mathbf{\Sigma}^T\mathbf{\Sigma}$ and their corresponding eigenvectors will give the column space of $\mathbf{V}$.

To find these eigenvalues (λ_i) and their corresponding eigenvectors ($\vec{v}_i$) we take the determinant from the eigenvalue equation for a linear transformation (see Appendix A.2)

$$\begin{aligned}
\mathbf{A}^T\mathbf{A}\vec{v} &= \lambda\vec{v} \\
\mathbf{A}^T\mathbf{A} &= \lambda\mathbf{I}\vec{v} \\
\left(\mathbf{A}^T\mathbf{A} - \lambda\mathbf{I}\right)\vec{v} &= \vec{0} \\
\det\left(\mathbf{A}^T\mathbf{A} - \lambda\mathbf{I}\right) &= \det\left(\begin{bmatrix} 26-\lambda & 18 \\ 18 & 74-\lambda \end{bmatrix}\right) \\
0 &= \lambda^2 - 100\lambda + 1600 \\
0 &= (\lambda - 80)(\lambda - 20)
\end{aligned}$$

Therefore our two eigenvalues are $\lambda = 20$ and $\lambda = 80$. We can find the corresponding eigenvectors to be (See A.2)

$$\vec{v}_{20} = \begin{bmatrix} \frac{-3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{bmatrix} \quad \text{and} \quad \vec{v}_{80} = \begin{bmatrix} \frac{1}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} \end{bmatrix}$$

This tells us that

$$\mathbf{V} = \begin{bmatrix} \frac{-3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \end{bmatrix} \quad \text{and} \quad \mathbf{\Sigma} = \begin{bmatrix} 20 & 0 \\ 0 & 80 \end{bmatrix} = \begin{bmatrix} 2\sqrt{5} & 0 \\ 0 & 4\sqrt{5} \end{bmatrix}$$

Using now what we obtained in (2), $\mathbf{AV} = \mathbf{U\Sigma}$, we can get $\mathbf{U}$ by decomposing the left side of the equation by our singular values

$$\mathbf{AV} = \begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix} \begin{bmatrix} \frac{-3}{\sqrt{10}} & \frac{1}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} & \frac{3}{\sqrt{10}} \end{bmatrix} = \begin{bmatrix} -\sqrt{10} & 2\sqrt{10} \\ \sqrt{10} & 2\sqrt{10} \end{bmatrix}$$

To get unitary matrix $\mathbf{U}$

$$\mathbf{U\Sigma} = \begin{bmatrix} -\sqrt{10} & 2\sqrt{10} \\ \sqrt{10} & 2\sqrt{10} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} 2\sqrt{5} & 0 \\ 0 & 4\sqrt{5} \end{bmatrix}$$

Solving this system yields elements of $\mathbf{U}$

$$\mathbf{U} = \begin{bmatrix} -1\sqrt{2} & 1\sqrt{2} \\ 1\sqrt{2} & 1\sqrt{2} \end{bmatrix}$$

A.2. Solution to the Eigenvalue Problem

Applying a linear transformation to a vector space can be thought of as changing the direction and magnitude of all the basis in the space. Eigenvectors ($\vec{v}_i$) are the set of non-zero vectors that remain in the same direction and scale by an amount λ_i after a linear transformation. Therefore, we can write for a linear transformation $\mathbf{A}$,

$$\begin{aligned} \mathbf{A}\vec{v} &= \lambda\vec{v} \\ \mathbf{A}\vec{v} &= \lambda\mathbf{I}\vec{v} \\ (\mathbf{A} - \lambda\mathbf{I})\,\vec{v} &= \vec{0} \end{aligned}$$

This is called the *eigenvalue problem.* To get the eigenvectors we are looking for, we can take the inverse of the left side matrix $(\mathbf{A} - \lambda\mathbf{I})$

$$\vec{v} = (\mathbf{A} - \lambda\mathbf{I})^{-1}\,\vec{0}$$

Here, we have a problem. If $(\mathbf{A} - \lambda\mathbf{I})^{-1}$ represents a finite operator then a finite operator on the null vector should always result in the null vector $\vec{0}$. This means that $\vec{v}$ will always equal the null vector no matter the composition of $\mathbf{A}$. This implies that a finite operator will not work if we seek a non-trivial solution to the eigenvalue problem. What we need is an infinite operator. Recall from matrix theory that

$$\mathbf{M}^{-1} = \frac{\text{cofactor}\,\mathbf{M}^{-1}}{\det\mathbf{M}}$$

This means that if we want an infinite operator $\mathbf{M}^{-1}$ we need to set the determinant of $\mathbf{M}$ to zero. Note that we now have an indeterminant term in our equation $\vec{v} = \infty \times 0$ however infinite operators are outside the scope of this documentation. For now, we can take our chances by setting the determinant to zero and exploring where that leads us.

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

For a matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

We get

$$\begin{aligned} \det(\mathbf{A} - \lambda\mathbf{I}) = \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} &= 0 \\ \lambda^2 - 4\lambda + 3 &= 0 \end{aligned}$$

This equation is called the *characteristic equation* and its roots yield us the eigenvalues of $\mathbf{A}$.

$$(\lambda - 1)(\lambda - 3) = 0$$

Therefore, $\lambda = 1$ and $\lambda = 3$ are the eigenvalues of $\mathbf{A}$. To get the eigenvectors we can plug back in our eigenvalues λ_i to get their corresponding eigenvectors $\vec{v}_1$ and $\vec{v}_3$

Starting with $\lambda = 1$, we get

$$
\begin{aligned}
(\mathbf{A} - \lambda \mathbf{I})\,\vec{v} &= 0 \\
\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
1v_1 + 1v_2 &= 0 \\
v_1 &= -v_2
\end{aligned}
$$

Therefore, we have a free parameter k_1 such that $\vec{v}_1 = k_1 \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. We can choose any value for k_1 and this relation will hold. It is convention to choose a value for k_1 that yields unit vectors. Here we will set $k_1 = 1$ to get our first eigenvector

$$\vec{v}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Similarly, for $\lambda = 3$, we get

$$\vec{v}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

A.3. Solving Total Least Squares with Eight-Point Algorithm using SVD

Suppose for a given system in bilinear form $\mathbf{x}'\mathbf{F}\mathbf{x} = 0$ we wish to find $\mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}$ for two homogeneous coordinate systems $\mathbf{x}' = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

This expands to

$$\begin{aligned} x_m x'_m F_{11} + x_m y'_m F_{12} + x_m F_{13} + \\ y_m x'_m F_{21} + y_m y'_m F_{22} + y_m F_{23} + \\ x'_m F_{31} + y'_m F_{32} + F_{33} = 0 \end{aligned}$$

We can set up a homogeneous linear system with 9 unknowns by combining terms

$$\begin{bmatrix} x_1 x'_1 & x_1 y'_1 & x_1 & y_1 y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_M x'_M & x_M y'_M & x_M & y_M y'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = \vec{0}$$

This means we need a total of $M = 8$ corresponding points from the two coordinate systems to solve this system of equations. Hence, the name eight-point algorithm. Notice that this number arises due to our choice of using homogeneous coordinates. Furthermore, unlike a homography estimation, here each point pair contributes a single equation as opposed to two. The question is now posed as a $\mathbf{MX} = \mathbf{0}$ problem. Since we are not interested in the trivial solution $\mathbf{X} = \mathbf{0}$, we add the constraint $\|X\| = 1$. Given the constraint, we can now solve this problem using SVD of $\mathbf{M}^T\mathbf{M}$. Entries of $\mathbf{X}$ are the elements of the column of $\mathbf{V}$ (in $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$) corresponding to the least singular value in $\mathbf{\Sigma}$. See here for derivation of least-squares solution to homogeneous equations.