Structure From Motion Project

Phase I

April 9, 2021

Introduction

The goal of this project is to recreate a 3D model from a 3D printed part using structure from motion. Since the 3D printed part is based on a CAD model in SolidWorks, the CAD model can act as the ground truth.

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.

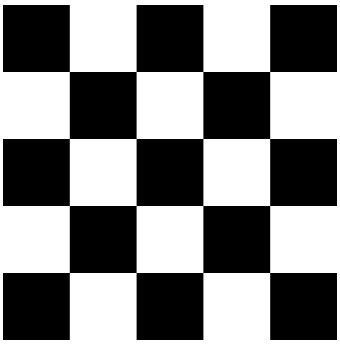

Figure 1: A *5x5* chess board pattern with *4x4* vertices. Transitions lines between black and white blocks are pixels with high local gradients.

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

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

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

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

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.

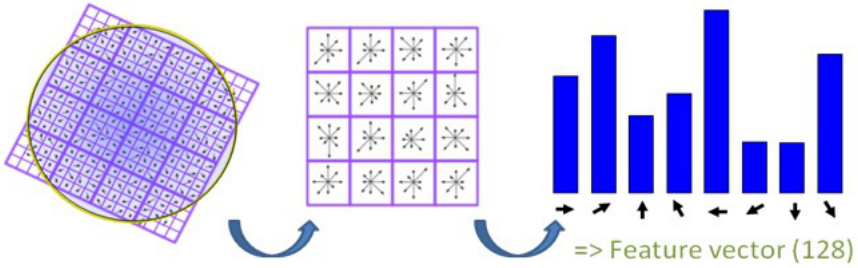

Figure 2: A keypoint descriptor is generated by accumulating local gradients into histograms for gradient directions. This histogram is represented by a feature vector. In this example, 128 directions are defined.

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.

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 $\mathbf{C}$ and $\mathbf{C}'$ form an epipolar plane π with the 3-space point $\mathbf{X}$. As shown in Figure 3, π intersects with both cameras' image planes denoted by $\mathbf{l}$ and $\mathbf{l}'$ - the epipolar lines. Furthermore, the camera baseline (line between the two camera centres) intersects each image plane at the epipoles $\mathbf{e}$. The location of these epipoles are conversed regardless of the location of $\mathbf{X}$. The point $\mathbf{x}$ must lie on the epipolar line $\mathbf{l}$'. 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 $\mathbf{C}'$ in the perspective of $\mathbf{C}$. However, the epipole can be found by checking where multiple epipolar lines converge.

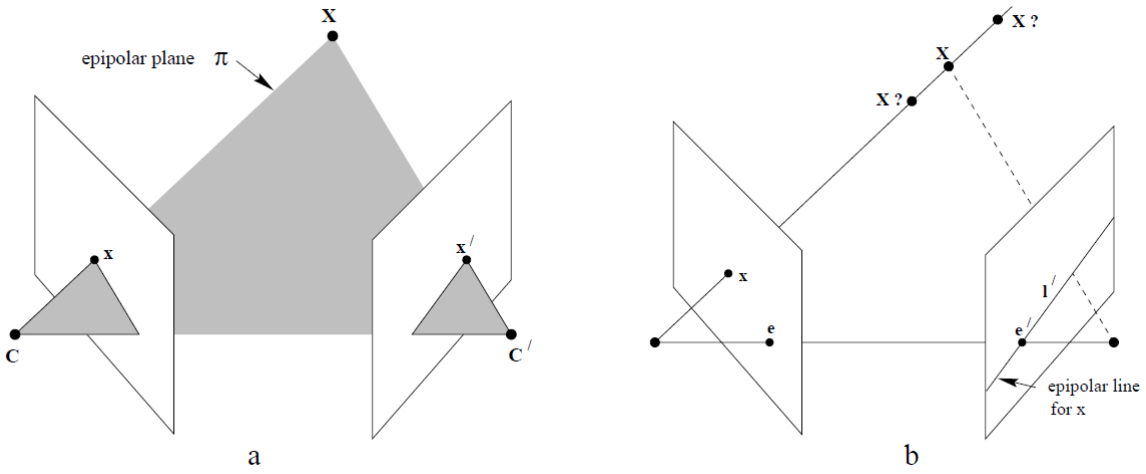

Figure 3: Point correspondence geometry. (a) The two cameras are indicated by their centres $\mathbf{C}$ and $\mathbf{C}'$ image planes. The camera centres, 3-space point $\mathbf{X}$, and its images $\mathbf{x}$ and $\mathbf{x}'$ lie in a common plane π. (b) This ray is imaged as a line $\mathbf{l}'$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

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 $\mathbf{x}'$ corresponding to $\mathbf{x}$ is on the epipolar line $\mathbf{l}'$ then the following must hold true

$$x^T l = 0$$

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

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

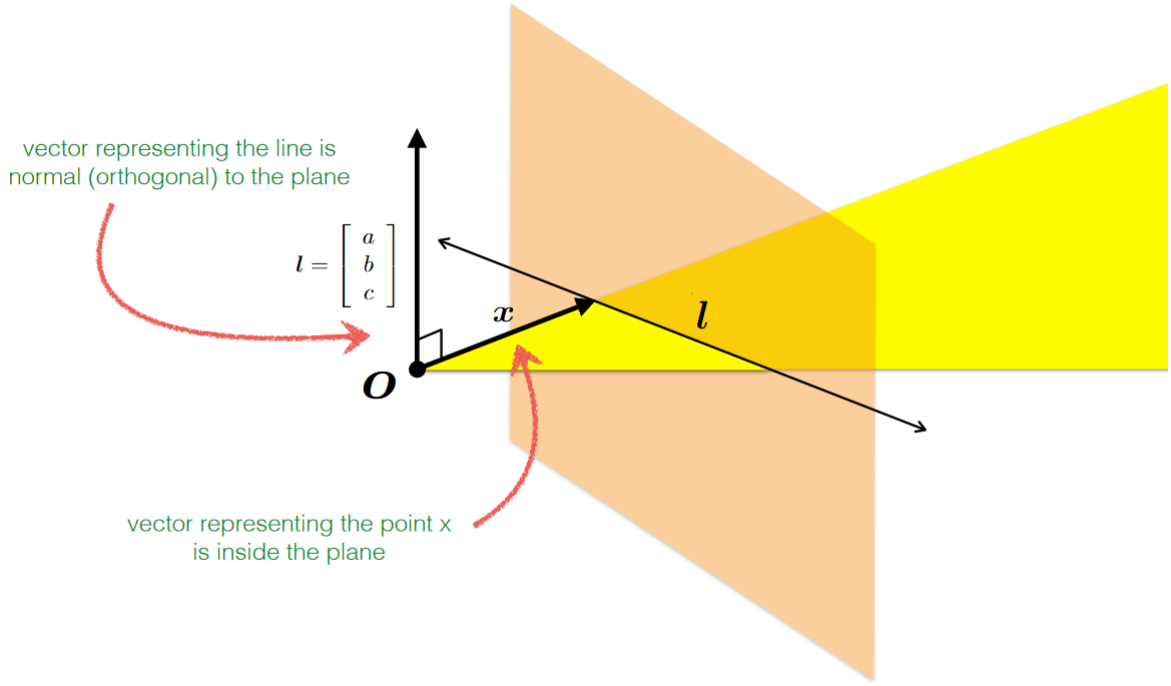

Figure 4: Orthogonality between $\mathbf{x}$ and $\mathbf{l}$ are shown. The orange plane is the image plane for the camera and the yellow plane represents π, the epipolar plane.

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 correspon-

dences 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. The assumption for the essential matrix is that cameras are calibrated. When calculating the fundamental matrix, we made no assumptions about the focal length or principle axis of the camera.

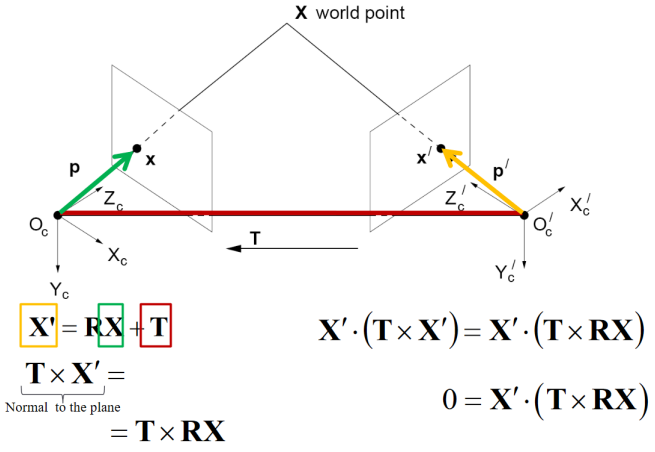

Figure 5: In order to get the Essential matrix...

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 $\langle x, y\rangle = \langle \mathbf{U}x, \mathbf{U}y\rangle \quad \forall\, x, y \in \Re^m$. 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

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

$$1v_1 + 1v_2 = 0$$
$$v_1 = -v_2$$

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.