ECSE 4961 Project 3

Aidan Westphal

November 2023

Purpose

The purpose of this project was to implement various aspects of 3D Reconstruction, specifically camera calibration, calculating the Fundamental Matrix, Rectification, and 3D Reconstruction with known camera parameters. This project uses a variety of tools to accomplish these, primarily MATLAB and C++ along with some hand calculations for distances at the end.

Theory

Camera Calibration

This section will gloss over several important concepts and values one can calculate in regard to 3D Reconstruction, along with some derivation/explanation of their origin. Please note that the concept of 3D Reconstruction is quite broad, as there are many different ways to accomplish it which vary depending on what information we're provided.

In this specific instant of 3D reconstruction, we are provided a set of 2D/3D calibration points followed by a set of matching points on a face to reconstruct. We are provided with no information about the camera. As such, our first task is to calibrate said cameras.

We note that our list of points is greater than 6 (we can uniquely solve) so we can repeat the same point calibration method as executed in Project 2. I will not fully rederive the equations but note they come from the full perspective equation and that each set of points provides two equations. We get the following homogenous matrix equation:

$$\begin{bmatrix} \vec{P_i} & 1 & 0 & 0 & 0 & 0 & -c_iX_i & -c_iY_i & -c_iZ_i & -c_i \\ 0 & 0 & 0 & 0 & \vec{P_i} & 1 & -r_iX_i & -r_iY_i & -r_iZ_i & -r_i \end{bmatrix} \begin{bmatrix} \vec{p_1} \\ p_{14} \\ \vec{p_2} \\ p_{24} \\ \vec{p_3} \\ p_{34} \end{bmatrix} = \vec{0}$$

We note that, for each point, we add the two specified equations. We take the least squares solution by calculating the SVD of matrix A and taking the vector corresponding to its smallest singular value. From here we recover

the projection matrix and can solve for its components through the following relationships:

$$\vec{r_3} = \vec{p_3}$$
$$t_z = p_{34}$$
$$c_0 = \vec{p_1}\vec{p_3}^T$$
$$r_0 = \vec{p_2}\vec{p_3}^T$$
$$s_x f = \sqrt{\vec{p_1}\vec{p_1}^T - c_0^2}$$
$$s_y f = \sqrt{\vec{p_2}\vec{p_2}^T - r_0^2}$$
$$t_x = (p_{14} - c_0 t_z)/s_x f$$
$$t_y = (p_{24} - r_0 t_z)/s_y f$$
$$\vec{r_1} = (\vec{p_1} - c_0\vec{r_3})/s_x f$$
$$\vec{r_2} = (\vec{p_2} - c_0\vec{r_3})/s_y f$$

As such we recover a set of parameters (W, R, T) for either camera.

Important Values
Now that the cameras are calibrated, we can start deriving some of these important values and discuss 3D Reconstruction theory.

The main goal with 3D Reconstruction involves obtaining what we already have: individual camera parameters and orientations. We were able to obtain these because a calibration set was provided, but what if it wasn't? How can we approximate these values?

The following is a short derivation of the Fundamental and Essential matrices. Oftentimes, especially when presented with only two images without any 3D or camera information at all, there is little we can calculate about the cameras. However, the Fundamental Matrix is one of these important values we can calculate as it describes the mapping between the image points on either plane. We begin by noting $P_L = RP_R + T$, where P_L, P_R are points in the left and right camera frame respectively, related by R, T. We can rearrange to $P_R = R^T(P_L - T)$. We then note that the point that we are reconstructing lies on a backprojection plane formed between itself and ei-

ther camera point. We note that the cross product $(T \times P_L)$ describes this normal and that its inner product with any other vector on the plane, like $(P_L - T)$, equals zero. We then recall $(P_L - T) = RP_R$ by our first equation leading to the following:

$$(T \times P_L)^T RP_R = 0$$

We note that we can describe the inner product w.r.t T by a skew matrix, thus $(T \times P_L) = SP_L$ with S being the following:

$$S = \begin{bmatrix} 0 & -T_Z & T_Y \\ T_Z & 0 & -T_X \\ -T_Y & T_X & 0 \end{bmatrix}$$

Thus, we are left with $P_L^T S^T R P_R = P_L^T E P_R = 0$ and $E = S^T R$ is the Essential Matrix. Note E is rank 2 by means of the cross product. Note how E describes a mapping between the camera frames. This equation is known as the Essential Equation.

We now bring in full perspective. Note $\lambda_L U_L = W_L P_L$ and $\lambda_R U_R = W_R P_R$ where U_L, U_R are in the row-column frame. We substitute these into the essential equation, yielding $U_L^T F U_R = 0$ for $F = W_L^{-T} E W_R^{-1}$ being the Fundamental Matrix and this equation is the Fundamental Equation.

F has a lot of applications. Primarily, it can be used to describe the location of epipoles and epipolar lines. Specifically, $F^T U_L$ and FU_R yield the epipolar line constraints $\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T$ for the left and right images respectively. Similarly, the epipoles for the left and right images are found by $F^T e_L = 0$ and $Fe_R = 0$. The Fundamental Matrix also encodes all information about the camera parameters which can be attempted to be approximated. We can calculate this matrix either from known camera parameters or using an algorithm such as the 8 point method.

The last piece of information we need is the orientation between the two camera frames. Say we have determined the relative orientations of either camera via calibration, we need to describe points in the right camera frame

w.r.t the left camera frame. Using the orientations of each camera frame to the object frame we have the following derivation:

$$P_l = R_L P + T_L,\ P_r = R_R P + T_R \implies P = R_R^T(P_r - T_R)$$

Substituting we get:

$$P_l = R_L R_R^T(P_r - T_R) + T_L = R_L R_R^T P_r - R_L R_R^T T_R + T_L$$

Thus we get the relative orientation R,T as:

$$R = R_L R_R^T,\ T = -RT_R + T_L$$

We can use these to calculate F.

Rectification

Rectification is the process of rotating the image frames of each camera such that they are coplanar and parallel to the baseline. The main use of this algorithm is for feature matching, as the epipoles of either camera are at infinity (baseline is parallel to the image frames implies that the backprojection line between each camera (baseline) never intersects the image frame), implying matching feature points will exist on roughly (theoretically the same) pixel row for either image, reducing our search time and complexity to simply searching a row of pixels.

To perform rectification, we need to know the camera parameters of each camera along with the relative orientation between each camera (R,T as calculated above). We follow the following steps:

1. Construct a rotation matrix R'_L which rotates the left image frame such that it's parallel to the baseline. First note that the X_L axis maps onto T_L so we set our first row to $\hat{T_L}$, the unit vector of T_L. We set our second axis to an arbitrary vector which is orthonormal to the first row, in this project I chose the normalized form of $\begin{bmatrix} t_y & -t_x & 0 \end{bmatrix}$. We then choose the third row of the rotation matrix to be the cross product between the first two rows.
2. We note the following relationship between points in the rotated frame c'_L, r'_L can be expressed in terms of their old c_L, r_L points by

backward mapping:

$$\lambda \begin{bmatrix} c_L \\ r_L \\ 1 \end{bmatrix} = W'_L R_L^{T\prime} W_L^{-1} \begin{bmatrix} c'_L \\ r'_L \\ 1 \end{bmatrix}$$

3. We repeat the same procedure and note that $R'_R = R'_L R$. In the above equation, replace R'_L with R'_R.

We note that rectification is not a unique construction. In this process, we have two main choices that vary our output: the choice for the second row in the rotation matrix and a scale factor d for fs_x, fs_y in either W' matrix. Depending on how one constructs the rotation matrix, more or less of the image will be visible in the final output. Similarly, the scaling of d will have a "focus" impact on the image. Sometimes rectification will hide important details (i.e. they are cut off) and thus scaling d downwards allows more of these regions to fit into our image.

3D Reconstruction

We note that we know our camera parameters and the relative orientation between each camera, thus we can perform a full reconstruction. While it's impossible to know where the original object frame was, we can let our object frame coincide with the left camera. Then, we have the following relationships by full perspective projection:

$$\lambda_L \begin{bmatrix} c_L \\ r_L \\ 1 \end{bmatrix} = W_L \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix}$$

$$\lambda_R \begin{bmatrix} c_R \\ r_R \\ 1 \end{bmatrix} = W_R \begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = W_R (R^T \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} - R^T T)$$

Note how a point $P_L = RP_R + T$ so we backsolve for the point P_R, giving the above equation. We note we have 5 parameters and 6 equations so we solve. We use a least squares method such as with camera calibration, solving for a vector $V = \begin{bmatrix} \lambda_L & \lambda_R & X_L & Y_L & Z_L \end{bmatrix}$. We describe the above equations as a homogenous system as follows:

$$\lambda_L \begin{bmatrix} c_L \\ r_L \\ 1 \end{bmatrix} - W_L \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} = 0$$

$$\lambda_R \begin{bmatrix} c_R \\ r_R \\ 1 \end{bmatrix} - A \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} + AT = 0 \text{ , where A} = W_R R^T$$

We write the following as a matrix equation equal to zero (first-order least squares solution). Note: A_n is the nth row of A: a vector.

$$\begin{bmatrix} c_L & 0 & -(fs_x)_L & 0 & -c_0 \\ r_L & 0 & 0 & -(fs_y)_L & -r_0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & c_R & -A_{11} & -A_{12} & -A_{13} \\ 0 & r_R & -A_{21} & -A_{22} & -A_{23} \\ 0 & 1 & -A_{31} & -A_{32} & -A_{33} \end{bmatrix} \begin{bmatrix} \lambda_L \\ \lambda_R \\ X_L \\ Y_L \\ Z_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ A_1 T \\ A_2 T \\ A_3 T \end{bmatrix}$$

We note the least squares solution to the above equation is given by: $V = (P^T P)^{-1} P^T b$ where P is the 6x5 matrix and b is the solution vector (zeros and the translation vector). Since A has full column rank (our set of equations are not degenerate), we can assure $(A^T A)^{-1}$ exists and thus perform the above calculation.

Summary

Using my MATLAB script from Project 2, I calibrated the left and right cameras, yielding the following:

Left Camera:

$$W = \begin{bmatrix} 1635 & 0 & 1055 \\ 0 & 1600 & 753 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} .5131 & -.8582 & -.0131 \\ -.1366 & -.0620 & -.9887 \\ -.8477 & -.5091 & .1490 \end{bmatrix} \quad T = \begin{bmatrix} -9.902 \\ 29 \\ -122.6 \end{bmatrix}$$

Right Camera:

$$W = \begin{bmatrix} 1656 & 0 & 1007 \\ 0 & 1637 & 719 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} .8565 & -.5160 & -.0121 \\ -.1289 & -.1884 & -.9736 \\ -.5001 & -.8354 & .2279 \end{bmatrix} \quad T = \begin{bmatrix} -20.9 \\ 16.5 \\ -122.4 \end{bmatrix}$$

We note that the W matrices are approximately equivalent, which makes sense. This implies that the images were taken by the same camera. As such,

variance is limited to the change in pose (orientation). We now calculate the relative orientation of the right camera w.r.t the left camera:

$$R = \begin{bmatrix} .8825 & .1084 & .4574 \\ -.0730 & .9919 & -.1052 \\ -.4651 & .0601 & .8832 \end{bmatrix}, T = \begin{bmatrix} 62.72 \\ -1.762 \\ -25.25 \end{bmatrix}$$

Calculating the Fundamental Matrix:

$$F = \begin{bmatrix} 3.778E-7 & -9.32E-6 & .0089 \\ -2.598E-6 & 2.483E-6 & .0427 \\ .0034 & -.0301 & -10.16 \end{bmatrix}$$

We now switch from MATLAB to C++ to implement the rectification and 3D reconstruction. I have provided four pairs of images, specifically the pre/post of the rectification with and without epipoles drawn for the facial points.

No Epipolar Lines:

(a) Left Image - Pre

(b) Right Image - Pre

(a) Left Image - Post (b) Right Image - Post

With Epipolar Lines:

(a) Left Image - Pre (b) Right Image - Pre

(a) Left Image - Post (b) Right Image - Post

I believe the labeling of the left and right images was mismatched, but note that the math works regardless. We also note that the epipolar lines successfully transform to parallel lines and approximately coincide. To alleviate small differences in W_L, W_R, I chose $W' = 0.5(W'_L + W'_R)$ and scaled it with a $d = 0.8$ to better fit the facial features of the left image.

I used MATLAB for the 3D reconstruction segment because of its scatter3 feature but I should note I could have just as easily implemented this in C++ as my matrix class supports all common matrix operations including inverses. Anyways, the algorithm produced the following 3D scatterplot:

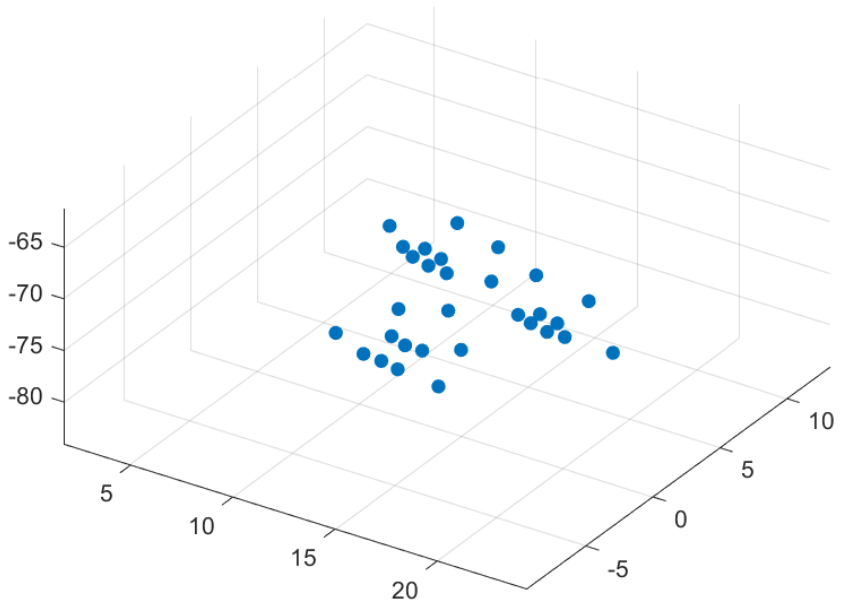

Figure 5: A 3D Reconstructed Face. One can clearly make out the eyes and eyebrows.

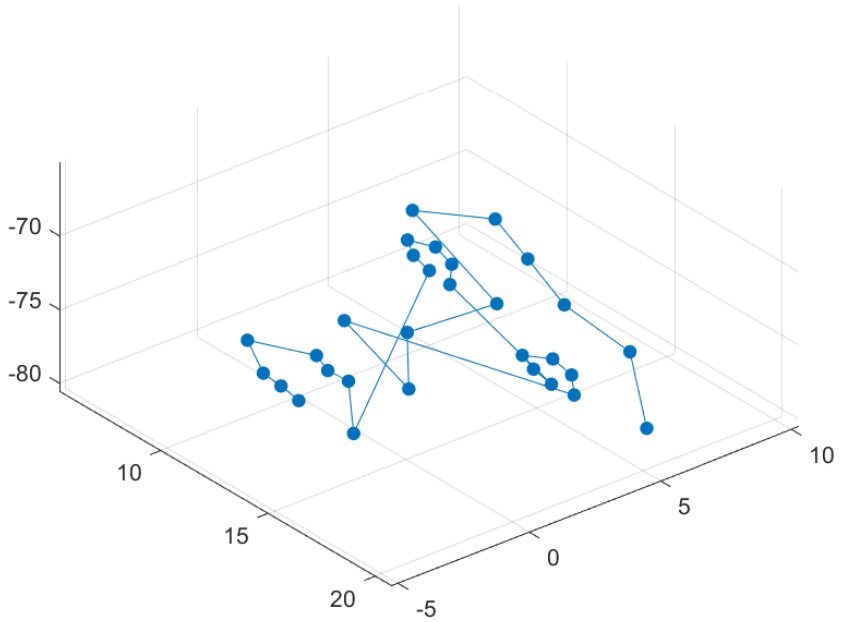

Figure 6: The same scatterplot but with a line superimposed over it.

By clicking on the points of the face, we can approximate measurements of the person's eyes. The following are some points of some important facial structures:

Left Eye: (8.75784 5.18063 −74.5751) and (11.0916 4.8921 −75.4093), width = 2.495

Right Eye: (14.7405 4.67439 −76.9468) and (16.8777 4.90312 −77.9812), width = 2.385

Mouth: (10.2741 −2.1639 −74.9645) and (15.4057 −2.30706 −76.8153), width = 5.457

Summary

In the end, I am satisfied with my work on this project. I have successfully completed all tasks required by this project: including camera calibration, calculating F, rectification, and 3D reconstruction. Furthermore, my results are consistent with the expected results. The intrinsic camera parameters were approximately equivalent which implies the same camera took each image. The Fundamental Matrix along with the other calculated results like R,T produced correct results further in the project. Specifically, the Fundamental Matrix produced the correct epipoles and R,T produced the correct rectification. Furthermore, the rectification produced parallel epipolar lines which approximately coincide with each other, as expected. Finally, I properly reconstructed a 3D face at the end.

As for error and troubles, there will always be error in a project like this, which I tried to mitigate at points (such as choosing a W' as the individual W'_R, W'_L matrices were slightly different). As for troubles, most of my time went to fixing a bug in my C++ rectification algorithm which resulting from me drawing faulty epipolar lines, such that they would diverge after the transformation. I chose to use C++ in the first place because I enjoy working with the language and already have a strong Matrix class that I had built up over this class along with image conversions from the first HW. I will admit though that MATLAB is easier to use, especially when I have to create high-level diagrams or find least squares solutions to homogenous systems (it's hard to calculate a null vector algorithmically, or at least out of my scope and likely the scope of this class). Regardless, I enjoyed whatever challenges I gave myself and found this project very interesting. I would say that my main source of improvement would be to either convert the entire program into MATLAB or, more favorably, begin using a linear algebra

library for C++ and move my entire project to C++. While it would be harder, I would find it more fun.