ME8135 — Assignment 4 Solution

Student: Arash Basirat Tabrizi
Submitted to: Dr. Sajad Saeedi
Due: June 16, 2023

Introduction:

In the previous assignments we explored a variety of estimation techniques. In many of our estimation techniques, we made the assumption that the noise corrupting the inputs or the measurements was zero-mean Gaussian. In reality, our inputs and/or measurements may also be corrupted with unknown biases. If we do not account for these, our estimate will also be biased.

Despite our best efforts to negate the effects of biases and find proper ***correspondences*** —pairing of a measurement with a part of a model— something deleterious can always happen to our measurements so that we are stuck with a datum that is highly improbable according to our noise model; we label this an ***outlier*** measurement. If we do not properly detect and remove outliers, many of our estimation techniques will fail, often catastrophically.

In the next section of this report, on outlier detection and rejection, will discuss a method to help deal with this type of problem.

RANSAC Technique for Handling Outliers:

Random sample consensus (RANSAC) is an iterative method to fit a parameterized model to a set of observed data containing outliers. Outliers are measurements that do not fit a model, while ***inliers*** do fit. RANSAC is a **probabilistic algorithm** in the sense that its ability to find a reasonable answer can only be guaranteed to occur with a certain probability that improves with more time spent in the search.

example data set with inliers and outliers   RANSAC finds line with most inliers
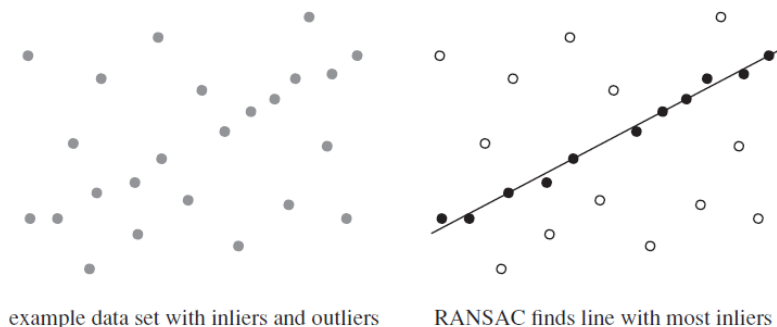
Figure 1: The RANSAC approach is to classify the data as either an inlier or an outlier and then only use the inliers in the line fit (Fig 5.6. in textbook).

Figure 1 provides a classic line fitting example in the presence of outliers. In its basic version, the algorithm requires the following steps for fitting a line to a set of 2D points (the sample set):

1. Select a random sample of the minimum required size to fit the model.

2. Calculate a model from this sample set using the points that we have already randomly sampled.

3. Investigate how much of our entire set of 2D points agrees with this model up to a tolerance δ. Compute the total number of inliers.

4. Repeat steps 1-3 until model with the most inliers over all samples is found.

1

1. Problem Statement:

Given source and destination 2D points, we wish to compute a homography that maps the source points to the destination. A minimum of 10 inliers needed to be detected by the RANSAC algorithm. The reprojection error of the inliers need to be less than 0.005.

Deliverables: Report the normalized homography transformation and provide the Python code. Additionally, provide a scatter plot, showing inliers (marked by o) and outliers (marked by x), where inliers of source and destination are connected via a line.

Solution Formulation:

This problem gives us the opportunity to investigate how RANSAC can be useful. Our goal is to fit the homographic transformation H given a set of pairs of corresponding source and destination points (correspondences) in presence of outliers. Before we introduce our refined RANSAC algorithm, let us further explore homography.

Given our sample set is a set of points in 2D, we initially visualize the sample set as follows:
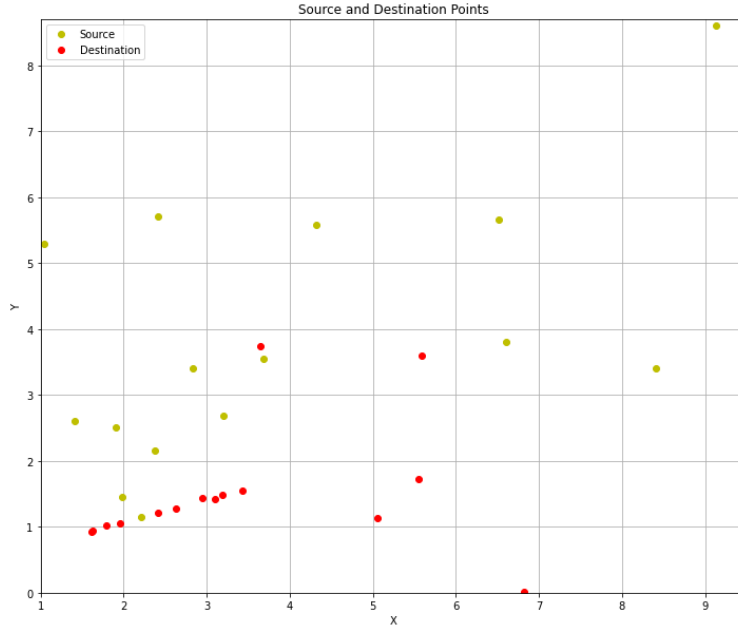

Figure 2: Visualizing the problem's sample set.

A **_homography_** maps a source point $(x_{\text{source}_1}, y_{\text{source}_1})$ to a destination point $(x_{\text{dest}_1}, y_{\text{dest}_1})$. In mathematical terms, we have the following:

$$\begin{bmatrix} ax_{\text{dest}_1} \\ ay_{\text{dest}_1} \\ a \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_{\text{source}_1} \\ y_{\text{source}_1} \\ 1 \end{bmatrix} \tag{1}$$

The homography matrix H is a 3x3 matrix with 9 parameters in total. Equation 1 can be rewritten as linear system in the form $\mathbf{A} \cdot \mathbf{h} = \mathbf{0}$, where $\mathbf{A}$ is a matrix constructed from the known source-destination point correspondences. The column vector $\mathbf{h}$ contains the unknown parameters of the homography matrix (refer to 2).

$$
\begin{bmatrix}
x_{\text{source}_1} & y_{\text{source}_1} & 1 & 0 & 0 & 0 & -x_{\text{source}_1}x_{\text{dest}_1} & -y_{\text{source}_1}x_{\text{dest}_1} & -x_{\text{dest}_1} \\
0 & 0 & 0 & x_{\text{source}_1} & y_{\text{source}_1} & 1 & -x_{\text{source}_1}y_{\text{dest}_1} & -y_{\text{source}_1}y_{\text{dest}_1} & -y_{\text{dest}_1}
\end{bmatrix}
\begin{bmatrix}
h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22}
\end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2)
$$

Now if we have more than one source-destination pairs, the linear system simply becomes:

$$
\begin{bmatrix}
x_{\text{source}_1} & y_{\text{source}_1} & 1 & 0 & 0 & 0 & -x_{\text{source}_1}x_{\text{dest}_1} & -y_{\text{source}_1}x_{\text{dest}_1} & -x_{\text{dest}_1} \\
0 & 0 & 0 & x_{\text{source}_1} & y_{\text{source}_1} & 1 & -x_{\text{source}_1}y_{\text{dest}_1} & -y_{\text{source}_1}y_{\text{dest}_1} & -y_{\text{dest}_1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_{\text{source}_n} & y_{\text{source}_n} & 1 & 0 & 0 & 0 & -x_{\text{source}_n}x_{\text{dest}_n} & -y_{\text{source}_n}x_{\text{dest}_n} & -x_{\text{dest}_n} \\
0 & 0 & 0 & x_{\text{source}_n} & y_{\text{source}_n} & 1 & -x_{\text{source}_n}y_{\text{dest}_n} & -y_{\text{source}_n}y_{\text{dest}_n} & -y_{\text{dest}_n}
\end{bmatrix}
\begin{bmatrix}
h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22}
\end{bmatrix}
=
\begin{bmatrix} 0_1 \\ \vdots \\ 0_{2n} \end{bmatrix}
$$
$$(3)$$

In equation 3, the subscript n represents the number of source-destination pairs, which is 15 in our problem.

To solve for the homography parameters **Direct Linear Transformation (DLT)** algorithm is the most common approach:

1. For each correspondence create a 2x9 matrix A_i

2. Concatenate into single $2n$x9 matrix **A**

3. Compute Singular Value Decomposition (SVD) of $A = U\Sigma V^{\text{T}}$

4. Store singular vector of the smallest singular value $\mathbf{h} = v_{\hat{i}}$

5. Reshape to get **H** (3x3)

Now we wish to refine our RANSAC algorithm to estimate homography using RANSAC. The RANSAC loop algorithm becomes:

1. Get four point correspondences (randomly).

2. Compute **H** using DLT.

3. Calculate re-projection error.

4. Count inliers.

5. Keep **H** if largest number of inliers.

We note that even though we have a maximum of 15 correspondences available to us, only 4 suffice for the algorithm to estimate the homography transformation. The outlined algorithm is implemented in the *compute_homography()* function definition of our solution code found here: *GitHub Repo*/A4/A4.py

Lastly, in our Python implementation, the re-projection error parameter plays a significant role. Essentially, the re-projection error represents how well the estimated transformation (homography) aligns the source points to their corresponding destination points. It measures the difference between the transformed source point (after applying the estimated transformation, Figure 3) and the actual destination point.

In the code, in *compute_homography()* function definition, after estimating the transformation using RANSAC, the algorithm applies the transformation to the source points to obtain the projected (transformed) points. These projected points are then compared to the corresponding destination points. For each point correspondence, the re-projection error is calculated as the distance (typically Euclidean distance) between the projected point and the actual destination point. If this error is below the specified threshold, the point correspondence is considered an inlier, indicating a good fit to the estimated transformation. If the error exceeds the threshold, the point correspondence is considered an outlier, suggesting a poor fit to the estimated transformation.

Results and Observations:

The output of our Python code reports the normalized homography transformation to be:

$$\mathbf{H} = \begin{bmatrix} 0.71296407 & 0.42888655 & 0.13003295 \\ 0.2851198 & 0.1715983 & 0.42905969 \\ 0.15673523 & 0.01447761 & 1 \end{bmatrix} \tag{4}$$

The the transformed source points after applying the estimated transformation can be visualized in the following figure:
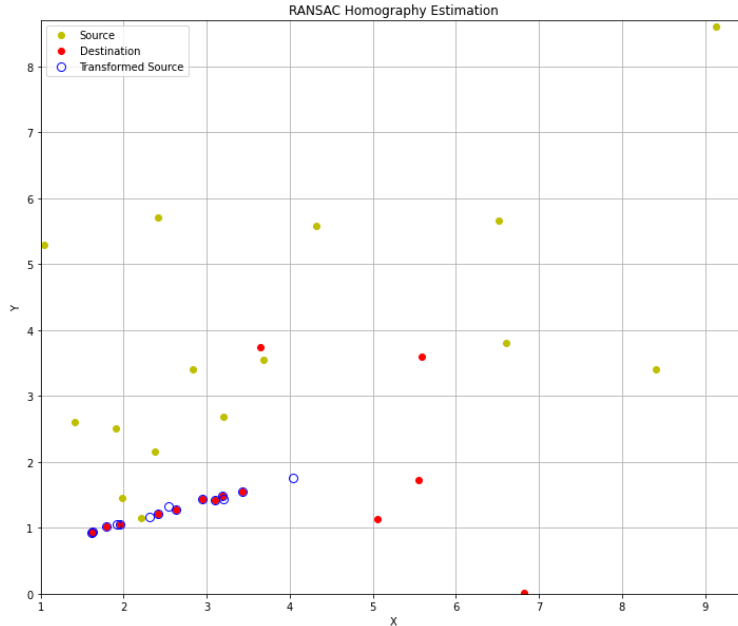

Figure 3: Plot of transformed source points.

In Figure 3 some of the blue circles appear to overlap but in total 15 transformed points were generated (corresponding to the 15 source points).

Lastly, the main plot and deliverable of the problem statement is presented in Figure 4.
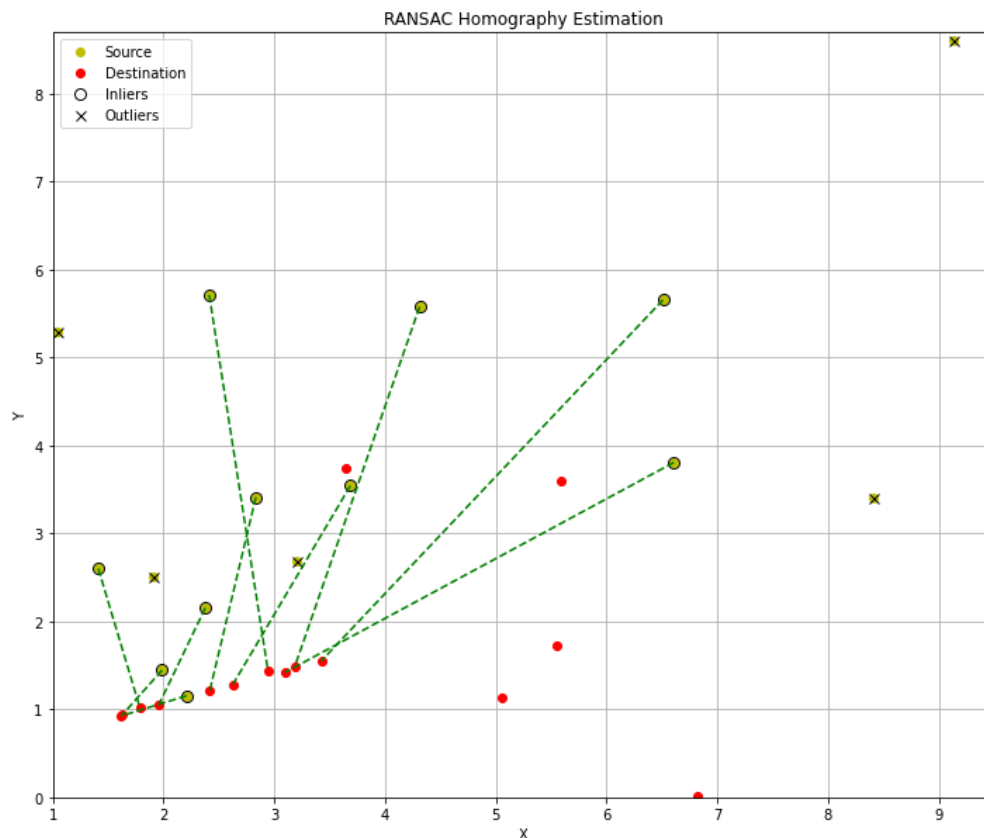
Figure 4: RANSAC homography estimation visualized with scatter plot.

Let us discuss the dashed green lines connecting each inlier source point to its corresponding destination point in Figure 4. These lines represent the mapping or alignment between the inliers of the source points and their corresponding destination points. By visually observing these lines, we get a better understanding of how the homography transformation maps the source points to the destination points. It helps visualize the relationship and correspondence between the source and destination points, specifically focusing on the inliers that have a good alignment.