CSCE 643 Multi-View Geometry CV Project II

***Abstract*—In this project, we explore and investigate different approaches for estimation in 2D projective transformation. In total, we implemented 3 approaches for finding the homography transformation between two different images, Direct Linear Transformation (DLT), normalized DLT and DLT with Sampson error minimization. This paper mainly focuses on the mathematical foundations for those methodologies used in our implementation and presents the results through combining two images into one panorama to validate the correctness of our approach. We can see clearly that with the advancement of our approach, the sampson error is gradually reduced (though there are few differences we can notice between those image panorama results).**

I. Problem Formulation

As shown in the project requirement documents, the aim of all problems in our project is to find a homography that can transform one image (shown in Fig. 1 (a)) into the same space of another image (shown in Fig. 1 (b)). Clearly, to achieve that, we have some points in common (basically overlappings) between those images as information we need to compute the homography. Suppose we have two set of points $\mathbf{x}_i$ and $\mathbf{x}'_i$, which as mentioned are distributed in the overlapping portion of two images, the aim of this project is to use different approaches to compute the transformation homography $\mathbf{H}$ so that

$$\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i \tag{1}$$

in homogeneous space. Also be noticed that to acquire a better algorithm performance, those point pairs should be selected as accurate as possible and we should avoid choosing points that are colinear, coplanar, etc. Furthermore, in problem 3, we need to use sampson error and MLE methodologies to calculate the error of our homography and minimize the error to acquire better homography. The rest of this paper will be organized problem by problem.

II. Simple DLT Approach

A. Mathematical Foundation

In the simplest DLT approach, we first transform the homogeneous equation given in equation 1 as follows:

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0} \tag{2}$$

Suppose $\mathbf{H}$ can be decomposed into combination of vectors like:

$$\mathbf{H} = \begin{pmatrix} \mathbf{h_1} \\ \mathbf{h_2} \\ \mathbf{h_3} \end{pmatrix} \tag{3}$$

Then we can rewrite equation 2 into the following form:

$$\mathbf{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}_1\mathbf{X}_i \\ \mathbf{h}_2\mathbf{X}_i \\ \mathbf{h}_3\mathbf{X}_i \end{pmatrix} \tag{4}$$

Suppose the coordinates of the set of points we have are:

$$\begin{aligned} \mathbf{x}'_i &= (x'_i, y'_i, z'_i) \\ \mathbf{x}_i &= (x_i, y_i, z_i) \end{aligned} \tag{5}$$

Then we have:

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}_3\mathbf{x}_i - z'_i\mathbf{h}\mathbf{x}_i \\ z'_i\mathbf{h}_1\mathbf{x}_i - x'_i\mathbf{h}^3\mathbf{x}_i \\ x'_i\mathbf{h}_2\mathbf{x}_i - y'_i\mathbf{h}_1\mathbf{x}_i \end{pmatrix} \tag{6}$$

As we have $\mathbf{h}_i\mathbf{x}_i = \mathbf{x}_i^T\mathbf{h}_j^T$, we can further get the following simultaneous linear equations set:

$$\begin{bmatrix} \mathbf{0}^T & -z'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ z'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \\ -y'_i\mathbf{x}_i^T & x'_i\mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{pmatrix} = \mathbf{0} \tag{7}$$

Since we can obtain the 3rd row of the left matrix above using by combining first two rows up to scale, we can safely prune equation 7 to only preserve the linearly independent first two rows in that matrix and obtain:

$$\mathbf{A}_i\mathbf{h} = \mathbf{0} \tag{8}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}^T & -z'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ z'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \end{bmatrix}, \mathbf{h} = \begin{pmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{pmatrix} \tag{9}$$

As we know:

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{10}$$

Now we have come to the point that is very similar to what we have done before in Project 1. After dehomogenization, the equation 8 we get is actually a equation with 8 unknowns in $\mathbf{h}$ (8 DoFs) which we can solve exactly if we have 4 point pairs (every point pair provide 2 equations), to get the homography we can do as the followings:

- If we have exact 4 point correspondences in both images, we can solve equation 8 to get an exact solution, but as there might be some noises in both images, it could lead to very bad results.
- We can also solve this equation is we can find more point correspondence to get so-called overdetermined solution

(a) Original source image(the image we wanna transform)

(b) The original target image(target image space we use for panorama)

Fig. 1. Original images

through singular value decomposition (SVD), that way we should get more accurate results since we are provided with more information in the image.

B. Experiment

In this section we present the setup of our experiment throughout the process of solving all three problems in this project. If without further specification, all experiments we used for problem 1-3 follow the setup here, including the point choices.

1) Point Choice: As we have mentioned before, to acquire better and accurate homography transformation between two images, we should choose point pairs in both images that are not colinear. Following this principle, we choose points in two images as shown in Fig. 2. The point choices are marked using red and green X markers in two images respectively.

2) Panorama: Ideally after we solved $\mathbf{H}$, we can use simple MATLAB functions to apply the homography on the source image and rectify it to the same space of Fig. 1 (b), the result we get at this step is shown in Fig. 3. Obviously, we are still one step away from the panorama, so how do we combine the new image we get and Fig. 1 (b) together? In our approach, we used two MATLAB methods *imref2d* and *imfuse* to do the trick. In MATLAB, *imref2d* function can be used to construct a spatial ref object assuming the world coordinate system is co-aligned with the intrinsic coordinate system while *imfuse* function can leverage the reference info of two images and combine them together as one image. After applying those two methods on our result, we can get the final panorama shown in Fig. 7.

Intuitively in the final panorama, the portion of image that are marked green is the transformed image of Fig. 1 (a) while the magneta portion is purely from Fig. 1 (b). The portion that are marked as grey is of course the overlapping pixels from both images. As we can see, due to a good choice of point pairs and the high quality of the original photo, the result we got is already considerably accurate.

C. Summary

III. Normalized DLT

As we learn from textbook, the accuracy of result homography generated by pure DLT is relevant on the coordinate frame for expressing points and the best coordinate system for DLT is the normalized system (based on input points). Thus we can improve the result of DLT by first normalizing the input points into a new coordinate system, do DLT algorithm and then denormalize to get an actual homography that can take image to the target space. Through the normalization, we can deal with less well conditioned problems using DLT.

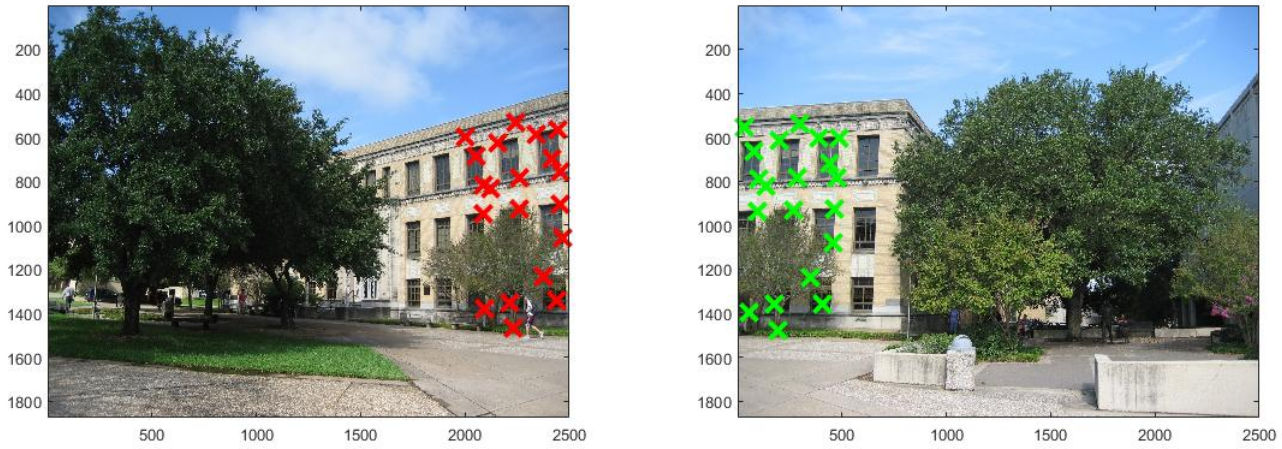
A. Mathmatical Foundation

The normalization process can be summarized as follows:

- Compute a similarity transformation $\mathbf{T}$, consisting of a translation and scaling that maps $\mathbf{x}_i$ to $\bar{\mathbf{x}}_i$ such that the centroid of all points $\bar{\mathbf{x}}_i$ is the coordinate origin $(0,0)^T$ and their average distance from the origin is $\sqrt{2}$.
- Compute a similar transformation $\mathbf{T}'$ using the same process to map $\mathbf{x}'_i$ to $\bar{\mathbf{x}}'_i$.
- Apply simple DLT in Problem 1 using the new corresponding point pairs $\bar{\mathbf{x}}_i \leftrightarrow \bar{\mathbf{x}}'_i$ to obtain the normalized homography $\bar{\mathbf{H}}$
- Use $\mathbf{H} = \mathbf{T}'^{-1}\bar{\mathbf{H}}\mathbf{T}$ to denormalize the homography from last step and get the actual homography.

Now let's do some math to further clarify the normalization process. Without loss of generality, suppose we wanna find the homography $\mathbf{T}$ for normalizing an image using a set of points $\mathbf{x}_i = (x_i, y_i, 1)^T$ in it, where:

$$T = \begin{pmatrix} s & 0 & t_x \\ 0 & s & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$



(a) Point choices in the first image

(b) Corresponding points in the second image

Fig. 2. Point pair choices between two original images

Fig. 3. The direct result we acquire after applying homography

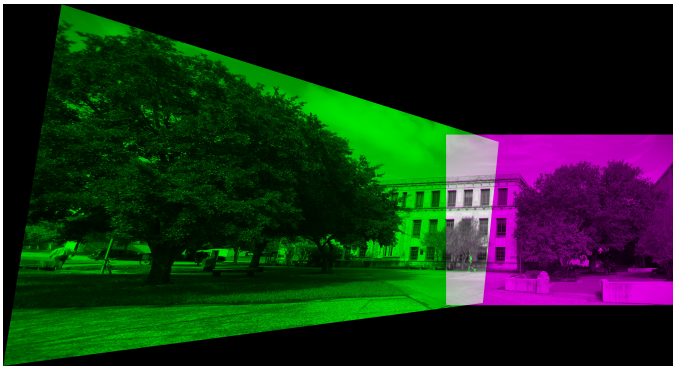


Fig. 4. Panorama formed by result of pure DLT

and it consists the combination of both the translation and scaling process that can take $\mathbf{x}_i$ to a new set of points $\bar{\mathbf{x}}_i = (x_i', y_i', 1)^T$ so that $\mathbf{x}_i' = \mathbf{T}\mathbf{x}_i$, that is to say:

$$\mathbf{x}_i' = \mathbf{T}\mathbf{x}_i = \begin{pmatrix} sx_i + t_x \\ sy_i + t_y \\ 1 \end{pmatrix} = \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} \tag{12}$$

Also as we mentioned, the centroid of the point set $\mathbf{x}_i'$ should be $(0, 0)$, which means:

$$\begin{pmatrix} \bar{x'} \\ \bar{y'} \\ 1 \end{pmatrix} = \begin{pmatrix} s\bar{x} + t_x \\ s\bar{y} + t_y \\ 1 \end{pmatrix} = (0, 0, 1)^T \tag{13}$$

where $\bar{x}, \bar{y}, \bar{x'}, \bar{y'}$ are the average of coordinates in both sets of points $\mathbf{x}_i$ and $\mathbf{y}_i$.

Additionally, we know that the average distance between origin and points in the normalized coordinate system is $\sqrt{2}$, therefore:

$$\begin{aligned} &\frac{1}{n}\sum_i^n \sqrt{{x_i'}^2 + {y_i'}^2} \\ =&\frac{s}{n}\sum_i^n \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2} = \sqrt{2} \end{aligned} \tag{14}$$

which means:

$$s = \frac{\sqrt{2}}{\frac{1}{n}\sum_i^n \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}} \tag{15}$$

Now, we can simply apply the point coordinates in certain image into equation 15 to solve s and then apply s into equation 13 to obtain both t_x and t_y, thereby the similar normalization transformation homography can be solved for both images we have. Say $\mathbf{T}$ and $\mathbf{T}'$ are two such homography for Fig. 1 (a) and Fig.1 (b) respectively, and we get two set of normalized coordinates in both images $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_i'$. By applying DLT on normalized point pairs, we obtain the transformation

homography between two normalized images to be $\hat{\mathbf{H}}$. If the actual homography we want is $\mathbf{H}$, then naturally we have:

$$\begin{aligned} \mathbf{x}_i' &= \mathbf{H}\mathbf{x}_i = T'^{-1}\hat{\mathbf{H}}\hat{\mathbf{x}}_i = \mathbf{T}'^{-1}\hat{\mathbf{H}}\mathbf{T}\mathbf{x}_i \\ \Rightarrow \mathbf{H} &= \mathbf{T}'^{-1}\hat{\mathbf{H}}\mathbf{T} \end{aligned} \tag{16}$$

So far, we have showed mathematically how we can normalize the image and use the normalized point pairs in DLT obtain the transformation homography between normalized image, as well as how to denormalize the homography back to normal space.

B. Experiment

As we mentioned in experiment 1, we use the same 20 point pairs as the input of our normalized DLT algorithm and applied the same methodology for imfusing the resulting images together. In this experiment, we still use the DLT approach for finding the homography, the difference here is that we first normalize the point correspondences and use normalized coordinates as the input of DLT. After doing DLT, we denormalize the homography we get and apply it for image synthesis. For brevity we ignored the direct result after applying homography we get from normalized DLT and jump into the Panorama section to show the match between our images:

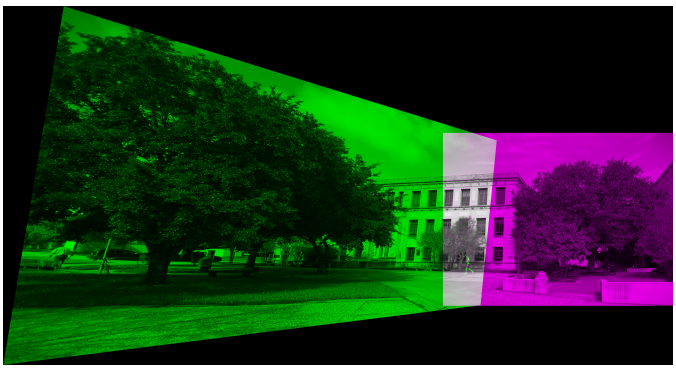


Fig. 5. Panorama formed by result of normalized DLT

Sadly, due to the high quality of original images and our good choice of point pairs, the pure DLT result is already too good that we cannot notice significant differences caused by applying normalization beforehand. We simply show the result here and leave the discovery of differences in the panorama to the discussion section.

C. Summary

IV. Gold Standard Algorithm with Sampson Error Function

Though we noticed pretty good results in both the pure DLT and normalized DLT experiment part, we want to emphasize that this might not be the actual case. As we can easily see from the image we used for experiment, the weather is good and clear, both overall and local lighting condition is promising thus noises in the image are limited. However, in reality, we don't usually have photos as good as these two for computation. When it comes to cloudy, rainy days, or even in the night, there will be a lot of noises in photos taken by cameras. Therefore, both DLT approaches presented above might be led to results with no sense due to noises, we definitely need to take measurement error into consideration in order to deal with varying realistic circumstances. In this section, we present and evaluate the Gold Standard Algorithm (aka MLE) with Sampson Error Function to further improve the robustness of our program.

A. Mathematical Foundation

Suppose for a given homography $\mathbf{H}$, without loss of generality we have points in different image spaces $\mathbf{x} = (x, y), \mathbf{x}' = (x', y')$ that satisfies $\mathbf{x}' \times \mathbf{H}\mathbf{x} = 0$. Naturally, this relationship defines an algebraic variety V_H in $\mathbb{R}^4$. So if we have a measurement $\mathbf{x}$ (basically a the measured point by a camera), the vector $\hat{\mathbf{x}}$ that minimizes the geometric error $\|\mathbf{x}-\hat{\mathbf{x}}\|^2$ is the closest point on V_H to $\mathbf{x}$. From the discussion in the above several sections, we also know that any point $\mathbf{x} = (x, y, x', y')^T$ that lies on V_H will also satisfy equation 8 ($\mathbf{Ah} = 0$). We can rewrite equation 8 as $C_H(\mathbf{x}) = 0$ to emphasize the actual dependency on $\mathbf{x}$ (once $\mathbf{h}$ is determined or given, $\mathbf{Ah}$ becomes a function of $\mathbf{x}$). Using Taylor expansion, we have:

$$C_H(\mathbf{x} + \delta_\mathbf{x}) = C_H(\mathbf{x}) + \frac{\partial C_H}{\partial \mathbf{x}}\delta_\mathbf{x} \tag{17}$$

Apparently, in our case, we have $\delta_\mathbf{x} = \hat{\mathbf{x}} - \mathbf{x}$ and we desire:

$$C_H(\hat{\mathbf{x}}) = 0 \Rightarrow C_H(\mathbf{x}) + (\frac{\partial C_H}{\partial \mathbf{x}})\delta\mathbf{x} = 0 \tag{18}$$

If we define J as the partial derivative matrix and ϵ to be the cost $C_H(\mathbf{x})$ associated with $\mathbf{x}$, we now turned our original problem into a well-formed minimization problem as:

- Find vector $\delta_\mathbf{x}$ that minimizes $\|\delta_\mathbf{x}\|$ subject to $J\delta_\mathbf{x} = -\epsilon$.

Using the standard Lagrange multiplier approach for such problem, we can finally get the Sampson error:

$$\|\delta_\mathbf{x}\|^2 = \delta_\mathbf{x}^T\delta_\mathbf{x} = \epsilon^T(JJ^T)^{-1}\epsilon \tag{19}$$

According to the textbook, when we apply Sampson approximation to geometric distance $d(\mathbf{x}, C)$ between a point $\mathbf{x}$ and conic C, we have $\epsilon = \mathbf{x}^T C\mathbf{x}$ and:

$$J = \begin{bmatrix} \frac{\partial(\mathbf{x}^T C\mathbf{x})}{\partial x} & \frac{\partial(\mathbf{x}^T C\mathbf{x})}{\partial y} \end{bmatrix} \tag{20}$$

We can now calculate Jacobian and ϵ based on point correspondences and plug them in equation 19 to obtain the sampson error.

B. Experiment

With the DLT approach similar to previous experiment, here we further use the output of DLT as the initial homography and the input to our Gold Standard Algorithm. In our implementation, we build a sampson error function for calculating the sampson error given certain homography (including jacob and epsilon calculation) and then we pass the sampson error function into MATLAB library function *lsqnonlin* which does

automatic nonlinear minimization. After the nonlinear library function stops, we got the minimized homography and applied it to combine our images, getting the result shown in Fig. 6.

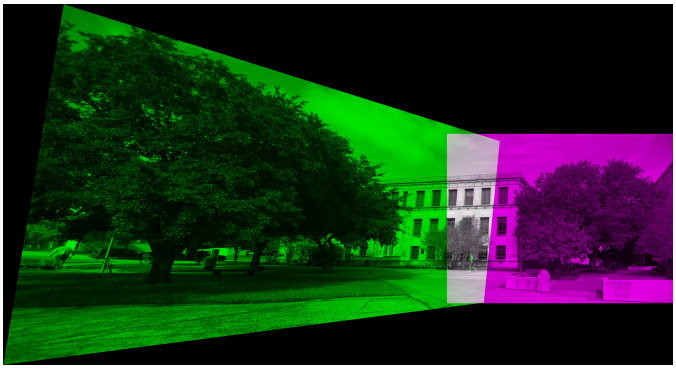


Fig. 6. Panorama formed by result of Gold Standard Algo with Sampson Error Function

Expectedly, the differences are negligible under human eye especially in such a low resolution, we will combine three results together in discussion for the sake of better comparison. However, we'd like to mention here that the sampson error of different approaches can be summarized as follows:

TABLE I
SAMPSON ERROR FOR DIFFERENT APPROACHES

pure DLT	normalized DLT	Gold Standard
5.7280	5.6887	4.5112

V. Discussion

Fig. 7. Combined result of three approaches (from section 2 to 4)

From the observation, we can find that the pure DLT result stretched the left hand image a little bit longer and in this case, all the object is longer than it appears in the real world. I inquiried and tested using other points set and found if we choose

more coplanar and colinear points, this phenomenon would become even significant than our current settings. The normalized DLT provide less stretched result, which looks very close to the world plane. And after Maximum likelihood estimation to reduce geometry error, the image looks even better, and I personally believe even the fidelity and clarity of the panorama seems to be enhanced. I would strongly suggests to use larger pictures like directly opening those 3 image results and quick switch between them, which is perhaps the only way for use to notice the little differences between them.

To wrap up, DLT can help us rectify an image to the space of another image, which can be used to create panorama, but the result produced by it is always worse than the approach of normalized DLT. The reason behind that is that there are unavoidable some noises in camera measurement, and without a normalization the magnitude of errors will be significantly higher. However, by manually identify and choosing point pairs that are accurate and choosing point pairs that are neither coplanar nor colinear, we can reduce the influence of noise in the process. Still, as we learnt from the class, we'd better use DLT to get a init better enough setting and then use other more advanced approach like MLE with sampson error function to further optimize and minimize the measurement error to get a better result in generating the panorama or rectify two images into the same space, or whenever we might need to do some error minimization. Additionally, please be noticed that we attached codes in the appendix for reference, to achieve different settings in different problems, the reader might have to follow instruction comments in code and uncomment or comment some code to get desired functionalities.

APPENDIX

A. Pure DLT

The $\mathbf{A}$ matrix constructed using 20 point pairs:

$$\begin{pmatrix}
0 & 0 & 0 & -2445.0 & -564.7 & -1.0 & 1.479 \cdot 10^6 & 3.415 \cdot 10^5 & 604.9 \\
2445.0 & 564.7 & 1.0 & 0 & 0 & 0 & -1.189 \cdot 10^6 & -2.745 \cdot 10^5 & -486.1 \\
0 & 0 & 0 & -2344.0 & -588.1 & -1.0 & 1.418 \cdot 10^6 & 3.557 \cdot 10^5 & 604.9 \\
2344.0 & 588.1 & 1.0 & 0 & 0 & 0 & -9.169 \cdot 10^5 & -2.301 \cdot 10^5 & -391.2 \\
0 & 0 & 0 & -2459.0 & -755.5 & -1.0 & 1.932 \cdot 10^6 & 5.936 \cdot 10^5 & 785.7 \\
2459.0 & 755.5 & 1.0 & 0 & 0 & 0 & -1.17 \cdot 10^6 & -3.596 \cdot 10^5 & -475.9 \\
0 & 0 & 0 & -2415.0 & -695.3 & -1.0 & 1.744 \cdot 10^6 & 5.02 \cdot 10^5 & 722.0 \\
2415.0 & 695.3 & 1.0 & 0 & 0 & 0 & -1.067 \cdot 10^6 & -3.073 \cdot 10^5 & -442.0 \\
0 & 0 & 0 & -2262.0 & -782.3 & -1.0 & 1.77 \cdot 10^6 & 6.12 \cdot 10^5 & 782.3 \\
2262.0 & 782.3 & 1.0 & 0 & 0 & 0 & -6.397 \cdot 10^5 & -2.212 \cdot 10^5 & -282.8 \\
0 & 0 & 0 & -2161.0 & -628.3 & -1.0 & 1.329 \cdot 10^6 & 3.863 \cdot 10^5 & 614.9 \\
2161.0 & 628.3 & 1.0 & 0 & 0 & 0 & -4.353 \cdot 10^5 & -1.266 \cdot 10^5 & -201.5 \\
0 & 0 & 0 & -2245.0 & -534.5 & -1.0 & 1.208 \cdot 10^6 & 2.875 \cdot 10^5 & 537.9 \\
2245.0 & 534.5 & 1.0 & 0 & 0 & 0 & -6.73 \cdot 10^5 & -1.602 \cdot 10^5 & -299.7 \\
0 & 0 & 0 & -2459.0 & -902.8 & -1.0 & 2.278 \cdot 10^6 & 8.363 \cdot 10^5 & 926.3 \\
2459.0 & 902.8 & 1.0 & 0 & 0 & 0 & -1.145 \cdot 10^6 & -4.205 \cdot 10^5 & -465.8 \\
0 & 0 & 0 & -2469.0 & -1060.0 & -1.0 & 2.667 \cdot 10^6 & 1.145 \cdot 10^6 & 1080.0 \\
2469.0 & 1060.0 & 1.0 & 0 & 0 & 0 & -1.133 \cdot 10^6 & -4.866 \cdot 10^5 & -459.0 \\
0 & 0 & 0 & -2438.0 & -1345.0 & -1.0 & 3.312 \cdot 10^6 & 1.827 \cdot 10^6 & 1358.0 \\
2438.0 & 1345.0 & 1.0 & 0 & 0 & 0 & -9.87 \cdot 10^5 & -5.443 \cdot 10^5 & -404.8 \\
0 & 0 & 0 & -2123.0 & -835.9 & -1.0 & 1.754 \cdot 10^6 & 6.903 \cdot 10^5 & 825.8 \\
2123.0 & 835.9 & 1.0 & 0 & 0 & 0 & -2.911 \cdot 10^5 & -1.146 \cdot 10^5 & -137.1 \\
0 & 0 & 0 & -2089.0 & -946.4 & -1.0 & 1.963 \cdot 10^6 & 8.893 \cdot 10^5 & 939.7 \\
2089.0 & 946.4 & 1.0 & 0 & 0 & 0 & -1.944 \cdot 10^5 & -88055.0 & -93.04 \\
0 & 0 & 0 & -2096.0 & -1382.0 & -1.0 & 2.931 \cdot 10^6 & 1.932 \cdot 10^6 & 1398.0 \\
2096.0 & 1382.0 & 1.0 & 0 & 0 & 0 & -1.098 \cdot 10^5 & -72377.0 & -52.38 \\
0 & 0 & 0 & -2215.0 & -1352.0 & -1.0 & 3.016 \cdot 10^6 & 1.84 \cdot 10^6 & 1362.0 \\
2215.0 & 1352.0 & 1.0 & 0 & 0 & 0 & -3.937 \cdot 10^5 & -2.402 \cdot 10^5 & -177.8 \\
0 & 0 & 0 & -2235.0 & -1469.0 & -1.0 & 3.305 \cdot 10^6 & 2.172 \cdot 10^6 & 1479.0 \\
2235.0 & 1469.0 & 1.0 & 0 & 0 & 0 & -4.352 \cdot 10^5 & -2.859 \cdot 10^5 & -194.7 \\
0 & 0 & 0 & -2052.0 & -688.6 & -1.0 & 1.365 \cdot 10^6 & 4.58 \cdot 10^5 & 665.1 \\
2052.0 & 688.6 & 1.0 & 0 & 0 & 0 & -1.562 \cdot 10^5 & -52400.0 & -76.1 \\
0 & 0 & 0 & -2086.0 & -815.8 & -1.0 & 1.653 \cdot 10^6 & 6.464 \cdot 10^5 & 792.4 \\
2086.0 & 815.8 & 1.0 & 0 & 0 & 0 & -2.082 \cdot 10^5 & -81433.0 & -99.82 \\
0 & 0 & 0 & -2262.0 & -929.6 & -1.0 & 2.111 \cdot 10^6 & 8.673 \cdot 10^5 & 933.0 \\
2262.0 & 929.6 & 1.0 & 0 & 0 & 0 & -6.091 \cdot 10^5 & -2.503 \cdot 10^5 & -269.2 \\
0 & 0 & 0 & -2001.0 & -598.2 & -1.0 & 1.117 \cdot 10^6 & 3.338 \cdot 10^5 & 558.0 \\
2001.0 & 598.2 & 1.0 & 0 & 0 & 0 & -64144.0 & -19177.0 & -32.05 \\
0 & 0 & 0 & -2377.0 & -1231.0 & -1.0 & 2.943 \cdot 10^6 & 1.524 \cdot 10^6 & 1238.0 \\
2377.0 & 1231.0 & 1.0 & 0 & 0 & 0 & -8.334 \cdot 10^5 & -4.315 \cdot 10^5 & -350.6
\end{pmatrix} \tag{21}$$

The homography we solved out using pure DLT:

$$\mathbf{H} = \begin{pmatrix} 2.214 & -0.2198 & -4239.0 \\ 0.6458 & 2.057 & -1435.0 \\ 0.0004856 & -2.811 \cdot 10^{-5} & 1.0 \end{pmatrix} \tag{22}$$

B. Normalized DLT

The homographies we used for normalization for both images are as follows:

$$\mathbf{H}_1 = \begin{pmatrix} 0.004667 & 0 & -10.56 \\ 0 & 0.004667 & -4.225 \\ 0 & 0 & 1.0 \end{pmatrix}, \mathbf{H}_2 = \begin{pmatrix} 0.004612 & 0 & -1.243 \\ 0 & 0.004612 & -4.199 \\ 0 & 0 & 1.0 \end{pmatrix} \tag{23}$$

where $\mathbf{H}_1$ is for Fig. 1 (a) and $\mathbf{H}_2$ for Fig. 1 (b).

The $\mathbf{A}$ matrix constructed using 20 normalized point pairs:

$$\begin{pmatrix}
0 & 0 & 0 & -0.8563 & 1.589 & -1.0 & -1.207 & 2.239 & -1.409 \\
0.8563 & -1.589 & 1.0 & 0 & 0 & 0 & -0.8552 & 1.587 & -0.9986 \\
0 & 0 & 0 & -0.3819 & 1.48 & -1.0 & -0.5382 & 2.085 & -1.409 \\
0.3819 & -1.48 & 1.0 & 0 & 0 & 0 & -0.2143 & 0.8303 & -0.5611 \\
0 & 0 & 0 & -0.9196 & 0.6985 & -1.0 & -0.529 & 0.4018 & -0.5752 \\
0.9196 & -0.6985 & 1.0 & 0 & 0 & 0 & -0.8752 & 0.6648 & -0.9518 \\
0 & 0 & 0 & -0.714 & 0.9798 & -1.0 & -0.6202 & 0.8511 & -0.8687 \\
0.714 & -0.9798 & 1.0 & 0 & 0 & 0 & -0.568 & 0.7794 & -0.7955 \\
0 & 0 & 0 & -0.002372 & 0.5735 & -1.0 & -0.001401 & 0.3388 & -0.5907 \\
0.002372 & -0.5735 & 1.0 & 0 & 0 & 0 & -0.0001446 & 0.03495 & -0.06095 \\
0 & 0 & 0 & 0.4721 & 1.292 & -1.0 & 0.6433 & 1.761 & -1.363 \\
-0.4721 & -1.292 & 1.0 & 0 & 0 & 0 & -0.1483 & -0.406 & 0.3141 \\
0 & 0 & 0 & 0.0767 & 1.73 & -1.0 & 0.1318 & 2.972 & -1.718 \\
-0.0767 & -1.73 & 1.0 & 0 & 0 & 0 & 0.01067 & 0.2406 & -0.1391 \\
0 & 0 & 0 & -0.9196 & 0.01094 & -1.0 & 0.06746 & -0.0008024 & 0.07335 \\
0.9196 & -0.01094 & 1.0 & 0 & 0 & 0 & -0.8321 & 0.009898 & -0.9049 \\
0 & 0 & 0 & -0.967 & -0.7235 & -1.0 & 0.7579 & 0.567 & 0.7837 \\
0.967 & 0.7235 & 1.0 & 0 & 0 & 0 & -0.8448 & -0.6321 & -0.8736 \\
0 & 0 & 0 & -0.8247 & -2.052 & -1.0 & 1.703 & 4.238 & 2.065 \\
0.8247 & 2.052 & 1.0 & 0 & 0 & 0 & -0.5143 & -1.279 & -0.6236 \\
0 & 0 & 0 & 0.646 & 0.3235 & -1.0 & 0.2519 & 0.1261 & -0.3899 \\
-0.646 & -0.3235 & 1.0 & 0 & 0 & 0 & -0.3948 & -0.1977 & 0.6111 \\
0 & 0 & 0 & 0.8041 & -0.1922 & -1.0 & -0.1087 & 0.02597 & 0.1351 \\
-0.8041 & 0.1922 & 1.0 & 0 & 0 & 0 & -0.6548 & 0.1565 & 0.8142 \\
0 & 0 & 0 & 0.7725 & -2.224 & -1.0 & -1.739 & 5.005 & 2.251 \\
-0.7725 & 2.224 & 1.0 & 0 & 0 & 0 & -0.7739 & 2.228 & 1.002 \\
0 & 0 & 0 & 0.219 & -2.083 & -1.0 & -0.4558 & 4.335 & 2.081 \\
-0.219 & 2.083 & 1.0 & 0 & 0 & 0 & -0.09276 & 0.8822 & 0.4235 \\
0 & 0 & 0 & 0.1241 & -2.63 & -1.0 & -0.3254 & 6.894 & 2.621 \\
-0.1241 & 2.63 & 1.0 & 0 & 0 & 0 & -0.04288 & 0.9083 & 0.3454 \\
0 & 0 & 0 & 0.9781 & 1.011 & -1.0 & 1.106 & 1.144 & -1.131 \\
-0.9781 & -1.011 & 1.0 & 0 & 0 & 0 & -0.8728 & -0.9022 & 0.8924 \\
0 & 0 & 0 & 0.82 & 0.4172 & -1.0 & 0.4464 & 0.2271 & -0.5444 \\
-0.82 & -0.4172 & 1.0 & 0 & 0 & 0 & -0.642 & -0.3267 & 0.783 \\
0 & 0 & 0 & -0.002372 & -0.1141 & -1.0 & 0.0002473 & 0.01189 & 0.1042 \\
0.002372 & 0.1141 & 1.0 & 0 & 0 & 0 & $3.707 \cdot 10^{-6}$ & 0.0001783 & 0.001563 \\
0 & 0 & 0 & 1.215 & 1.433 & -1.0 & 1.975 & 2.329 & -1.625 \\
-1.215 & -1.433 & 1.0 & 0 & 0 & 0 & -1.331 & -1.57 & 1.096 \\
0 & 0 & 0 & -0.5401 & -1.52 & -1.0 & 0.8152 & 2.295 & 1.51 \\
0.5401 & 1.52 & 1.0 & 0 & 0 & 0 & -0.2017 & -0.5679 & -0.3735
\end{pmatrix} \tag{24}$$

The normalized homography $\mathbf{H}_{norm}$ we obtained directly by applying DLT to normalized point pairs:

$$\mathbf{H}_{norm} = \begin{pmatrix} 0.004612 & 0 & -1.243 \\ 0 & 0.004612 & -4.199 \\ 0 & 0 & 1.0 \end{pmatrix} \tag{25}$$

The actual homography $\mathbf{H}_{denorm}$ (denormalized from $\mathbf{H}_{norm}$) for rectifying the first image to the space of second one:

$$\mathbf{H}_{denorm} = \begin{pmatrix} 0.9927 & -0.1012 & 0.02453 \\ 0.09716 & 0.9928 & 8.025 \cdot 10^{-5} \\ 0.04906 & -0.003037 & 1.0 \end{pmatrix} \tag{26}$$

C. Gold Standard Algo with Sampson Error Function

To begin with, we refer readers to equations 21, 22, 24, 25 for $\mathbf{A}$ matrix and initial homography $\mathbf{H}$ we used for gold standard minimization.

After applying the DLT on normalized resulting homography, we get

$$\mathbf{H}_{norm} = \begin{pmatrix} 0.9908 & -0.102 & 0.02453 \\ 0.1049 & 0.9977 & 8.43 \cdot 10^{-5} \\ 0.6086 & -1.557 & 0.9998 \end{pmatrix} \tag{27}$$

Furthermore, after denormalizing the above homography, we have:

$$\mathbf{H}_{denorm} = \begin{pmatrix} 1.768 & -2.062 & -1858.0 \\ 2.692 & -5.607 & -103.7 \\ 0.00284 & -0.007267 & 1.154 \end{pmatrix} \tag{28}$$

APPENDIX

A. Main Entry

```
clc;close all; clear all;
% read images
fig_1 = imread('pics/fig_1.jpg');
fig_2 = imread('pics/fig_2.jpg');
% load point coordinates from pre-defined
load('points.mat')
x1 = x1';
x2 = x2';
y1 = y1';
y2 = y2';
ptsSize1 = size(x1);
ptsCount1 = ptsSize1(1);
ptsSize2 = size(x2);
ptsCount2 = ptsSize2(1);
% notice we organize all the points here in the vector format (x, y, 1)
global X1;
global X2;
X1 = ones(ptsCount1, 3);
X2 = ones(ptsCount2, 3);

X1(:, 1) = x1(:, 1);
X1(:, 2) = y1(:, 1);

X2(:, 1) = x2(:, 1);
X2(:, 2) = y2(:, 1);
% get points from the first figure
%[x1, y1] = getpts(get(imshow('fig_1.jpg'),'Parent'));

% get points from the second figure
%[x2, y2] = getpts(get(imshow('fig_2.jpg'), 'Parent'));

% this is the pure dlt part
% compute A mat
%global A;
global A;
A_dlt = computeAMat(X1, X2);
% call function to get homography
% final homography from dlt without normalization
H_dlt = dltHomography(A_dlt);
% combine two imgs and show
img_dlt_combined = twoImgHomoCombine(H_dlt', fig_1, fig_2);
figure(1), image(img_dlt_combined);
% write images to file
%imwrite(img_dlt_combined, 'results/pure_dlt.jpg');

% here is the normalized dlt part
% normalizing similarity homography for both images
%global H_norm_1;
H_norm_1 = normSimHomo(X1);
%global H_norm_2;
H_norm_2 = normSimHomo(X2);
% apply normalization on our points
X1_norm = applyHomoPerPt(H_norm_1, X1);
```

```
X2_norm = applyHomoPerPt(H_norm_2, X2);
A_norm = computeAMat(X1_norm, X2_norm);
% do DLT to get the normalized DLT homography
H_dlt_norm = dltHomography(A_norm);
% final denormalized dlt homography
H_dlt_denorm = inv(H_norm_2) * H_dlt_norm * H_norm_1;
img_dlt_norm_combined = twoImgHomoCombine(H_dlt_denorm', fig_1, fig_2);
figure(2), image(img_dlt_norm_combined);
%imwrite(img_dlt_norm_combined, 'results/norm_dlt.jpg');

%% now we use DLT homo as the start point to do sampson error minimization
% global variable for jacobian
global E;
% epsilon
global C;
%A = A_dlt_norm;
%se_dlt_norm = sampson_error(H_dlt_denorm);
A = A_dlt;
se_dlt = sampson_error(H_dlt);
%% let's use minimization here
%% cast from mat to vector for adapting sample
% for normalized dlt
%[homo_min,resnorm] = lsqnonlin(@sampson_error,H_dlt_norm);
%homo_min_denorm = inv(H_norm_2) * homo_min * H_norm_1;
% for pure dlt minimization
%[homo_min,resnorm] = lsqnonlin(@sampson_error,H_dlt);
% for norm dlt
%homo_min = inv(H_norm_2) * homo_min * H_norm_1;

%se_min = sampson_error(homo_min);

%img_dlt_min = twoImgHomoCombine(homo_min', fig_1, fig_2);
%figure(3), image(img_dlt_min);
%imwrite(img_dlt_min, 'results/dlt_min.jpg');

function img_combined = twoImgHomoCombine(H, fig1, fig2)
    transform = projective2d(H);
    % get image reference info for fig_1 and transformed final figure 1
    im_ref1 = imref2d(size(fig1));
    % new reference of figure 1 after transformation
    [im_trans1, im_trans_ref1] = imwarp(fig1, im_ref1, transform);
    % image reference info for fig_2
    im_ref2 = imref2d(size(fig2));
    % overlap two images
    [img_combined, img_combined_ref] = imfuse(im_trans1, im_trans_ref1, fig2, im_ref2
        );
end

function A = computeAMat(x1, x2)
    % build matrix for selected points in the first figure
    ptsSize1 = size(x1);
    ptsCount1 = ptsSize1(1);
    ptsSize2 = size(x2);
    ptsCount2 = ptsSize2(1);

    A = zeros(ptsCount1 * 2, 9);
```

```
109     for i = 1:ptsCount1
110         A(2*i - 1, :) = [0 0 0 -x1(i, 1) -x1(i, 2) -1 (x2(i, 2) * x1(i, 1)) (x2(i, 2)
                * x1(i, 2)) x2(i, 2)];
111         A(2*i, :) = [x1(i, 1) x1(i, 2) 1 0 0 0 (-x2(i, 1) * x1(i, 1)) (-x2(i, 1) * x1
                (i, 2)) (-x2(i, 1))];
112     end
113 end
114 % function that accepts input (x, y) coordinates vector from both image and
115 % output the homography
116 function H_col = dltHomography(A)
117
118     % build A matrix
119     %A = zeros(ptsCount1 * 2, 9);
120     %for i = 1:ptsCount1
121     %    A(2*i - 1, :) = [0 0 0 -x1(i, 1) -x1(i, 2) -1 (x2(i, 2) * x1(i, 1)) (x2(i,
            2) * x1(i, 2)) x2(i, 2)];
122     %    A(2*i, :) = [x1(i, 1) x1(i, 2) 1 0 0 0 (-x2(i, 1) * x1(i, 1)) (-x2(i, 1) *
            x1(i, 2)) (-x2(i, 1))];
123     %end
124
125     % singular value decomposition
126     [U,S,V] = svd(A);
127     V = V/V(9, 9);
128     H_col = [V(1, 9), V(2, 9), V(3, 9);
129         V(4, 9), V(5, 9), V(6, 9);
130         V(7, 9), V(8, 9), V(9, 9);];
131 end
132
133 % function for finding normalizing similarity homography
134 % accepting x and y vector for a series of points, output the homography
135 function H_norm =  normSimHomo(X)
136     % compute the avg value of both x and y coordinates
137     % TODO: whether mean function can be applied to multi row vector like
138     % we have here
139     avg_x = mean(X(:, 1));
140     avg_y = mean(X(:, 2));
141
142     % acquire number of points we have
143     ptsSize = size(X);
144     n = ptsSize(1);
145
146     % calculate the sum of distance of every points to the centroid
147     dist_sum = 0;
148     for i = 1:n
149         dist_sum = dist_sum + sqrt((X(i, 1) - avg_x)^2 + (X(i, 2) - avg_y)^2);
150     end
151     avg_dist = dist_sum / n;
152
153     % calculate diagonal element s in similarity homography
154     s = sqrt(2) / avg_dist;
155     t_x = -s * avg_x;
156     t_y = -s * avg_y;
157
158     H_norm = [s 0 t_x;
159         0 s t_y;
160         0 0 1];
```

```
161 end
162
163 % function for applying homography point wise
164 % by default we assume X=(x, y, 1) as a row
165 function X_new = applyHomoPerPt(H, X)
166     X_new = (H * (X'))';
167 end
168
169 % useless data
170 %x1 = [2282.1985645933014; 2484.0526315789471; 2477.3241626794256;
    2472.8385167464112; ...
171 %       2463.8672248803828; 2452.6531100478469; 2452.6531100478469;
    2401.0681818181815; ...
172 %       2300.1411483253587; 2450.4102870813394; 2340.5119617224877;
    2439.1961722488036; ...
173 %       2340.5119617224877; 2748.705741626794];
174
175 %x2 = [249.99999999999997; 447.99999999999989; 450.99999999999994;
    456.99999999999994; ...
176 %       465.99999999999994; 465.99999999999994; 465.99999999999994;
    414.99999999999994; ...
177 %       319.00000000000006; 483.99999999999994; 384.99999999999989;
    480.99999999999994; ...
178 %       384.99999999999989; 1264];
179
180 %y1 = [1385.4431818181815; 1389.928827751196; 1223.9599282296649; 1111.818779904306;
    ...
181 %       900.99342105263133; 856.13696172248785; 822.49461722488013;
    788.85227272727252; ...
182 %       804.55203349282283; 560.08433014354046; 582.51255980861242;
    501.77093301435383; ...
183 %       524.19916267942563; 562.32715311004767];
184
185 %y2 = [1400; 1400; 1238; 1127; 926.00000000000011; 887.00000000000011; 851; ...
186 %       809; 815; 599.00000000000023; 605.00000000000023; 539.00000000000023; ...
187 %       548.00000000000023; 458.00000000000023];
188
189 % this function evaluates the sampson error
190
191 function E_sum = sampson_error( homo_mat)
192 global A;
193 global X1;
194 global X2;
195 global E;
196 global C;
197     ptsSize1 = size(X1);
198     ptsCount1 = ptsSize1(1);
199     ptsSize2 = size(X2);
200     ptsCount2 = ptsSize2(1);
201
202     n = ptsCount1;
203
204     homo = mat_to_vector(homo_mat);
205     % residual cost
206     e = A * homo';
207     J = jacob(homo_mat, X1, X2);
```

```
208     E_sum = 0;
209     for k = 1 : n
210      % actual Jacobian
211      E = [J(k, 1 : 4)' J(k, 5 : 8)'];
212      % actual JJ^T
213      B = inv(E * E');
214      % epsilon
215      C = [ e(2 * k - 1,1)  e(2 * k,1)];
216      F = sqrt(C * B * C');
217      E_sum = E_sum + F;
218     end
219 end
220
221 function h = mat_to_vector(homo)
222     h = [homo(1, 1) homo(1, 2) homo(1, 3) homo(2, 1) homo(2, 2) homo(2, 3) homo(3, 1)
            homo(3, 2) homo(3, 3)];
223 end
224
225 % this function evaluates the jacobian matrix for CH(X).
226 function J = jacob(homo, X1, X2)
227
228 x1_size = (size(X1));
229 x1_pts = x1_size(1);
230 x2_size = (size(X2));
231 x2_pts = x2_size(1);
232
233 if(x1_pts ~= x2_pts)
234     error('select same amount of points in two figures!');
235 end
236
237 for i = 1 : x1_pts
238     a = (-1) * homo(2, 1) + homo(3, 1) * X1(i, 2);
239     b = (-1) * homo(2, 2) + homo(3, 2) * X2(i, 2);
240     c = 0;
241     d = homo(3,1) * X1(i, 1) + homo(3,2) * X1(i, 2) + homo(3,3);
242     e = homo(1,1) - homo(3,1) * X1(i, 1);
243     f = homo(1,2) - homo(3,2) * X2(i, 1);
244     g = (-1) * homo(3,1) * X1(i, 1) - homo(3,2) * X1(i, 2) - homo(3,3);
245     gg = 0;
246
247     m = [ a b c d];
248     n = [ e f g gg ];
249     p = [ m n ];
250     J(i,:,:) = p';
251 end
252
253 end
```