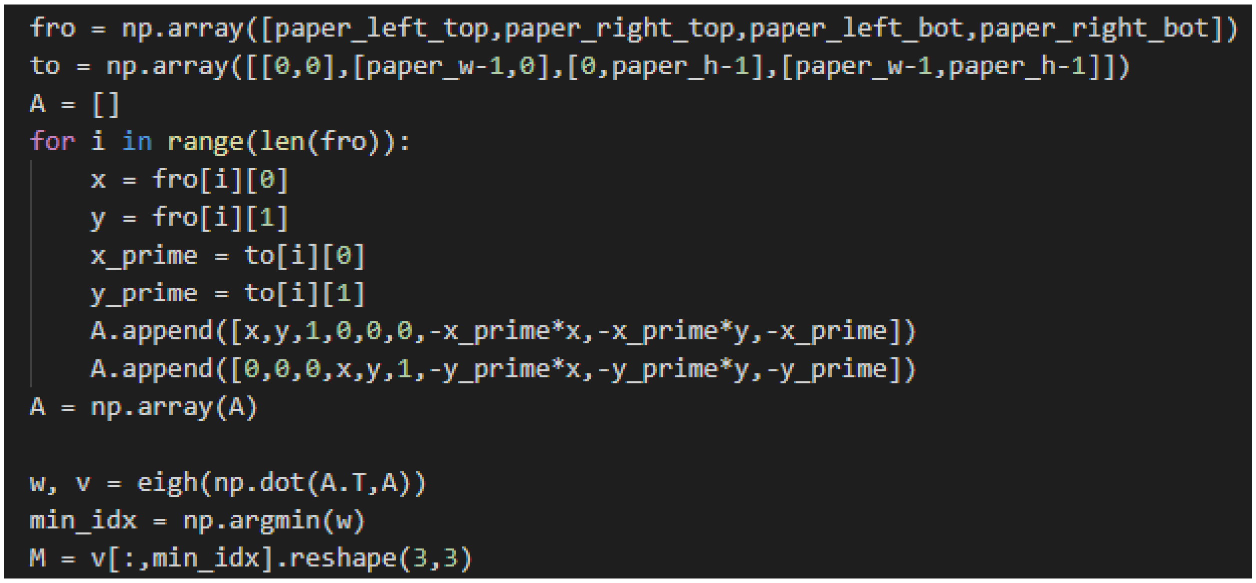
CSC420 Assignment 3
1002077726
Guanxiong Liu

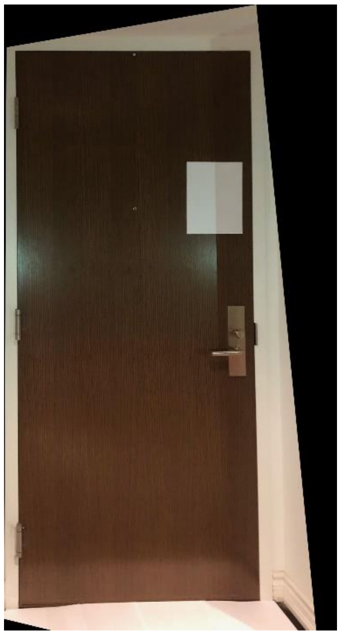
1. I transform the door along with letter sized paper to the scale of an actual letter sized paper. i.e. one pixel is 1 milimeter. Then I took 3 points(corners) of the door in the original photo apply with transformation matrix. After it is done, I can calculate the distances pairwise by using Euclidean distance and the distances I got are the actual width and height because the scale after transformation is the actual scale in millimeter. The width is 905.54mm and height is 2125.51mm.

   Derivation:



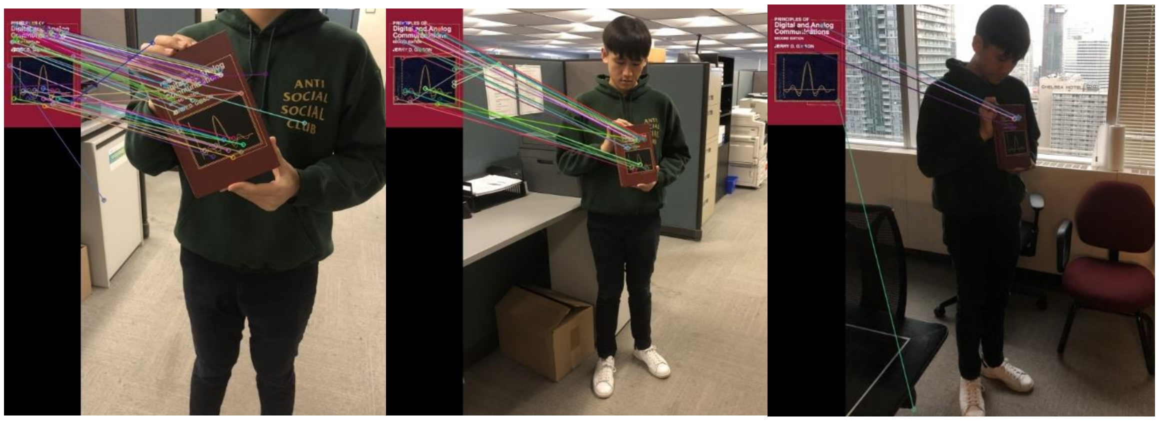
```
fro = np.array([paper_left_top,paper_right_top,paper_left_bot,paper_right_bot])
to = np.array([[0,0],[paper_w-1,0],[0,paper_h-1],[paper_w-1,paper_h-1]])
A = []
for i in range(len(fro)):
    x = fro[i][0]
    y = fro[i][1]
    x_prime = to[i][0]
    y_prime = to[i][1]
    A.append([x,y,1,0,0,0,-x_prime*x,-x_prime*y,-x_prime])
    A.append([0,0,0,x,y,1,-y_prime*x,-y_prime*y,-y_prime])
A = np.array(A)

w, v = eigh(np.dot(A.T,A))
min_idx = np.argmin(w)
M = v[:,min_idx].reshape(3,3)
```

   Result:



CSC420 Assignment 3
1002077726
Guanxiong Liu

2. A.

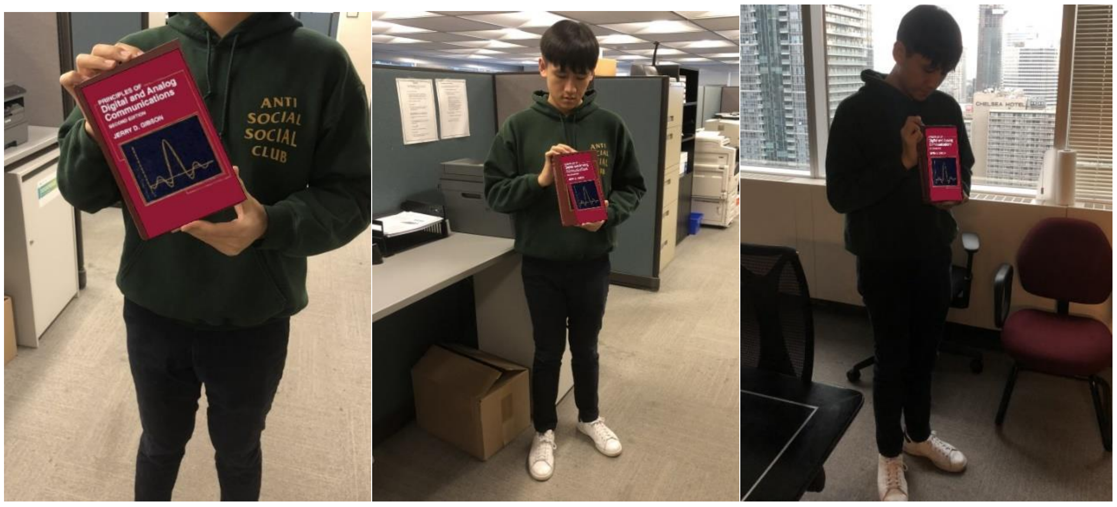


B. My visually estimates for image1,2 and 3 are 30%, 30%, 20% percent of outliers
The estimated minimum number of iterations is shown below:

	Image1	Image2	Image3
Affine	11	11	7
Homography	17	17	9

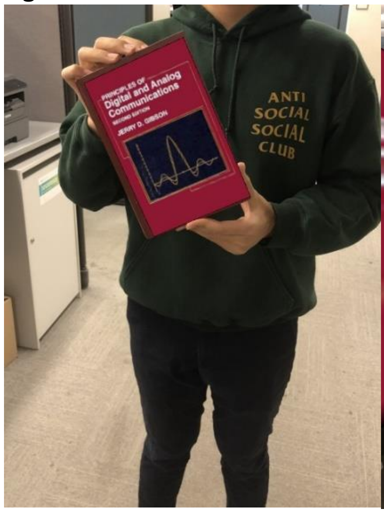
C and D. Use the above estimated minimum number of iterations to run the algorithm, the result is very bad.

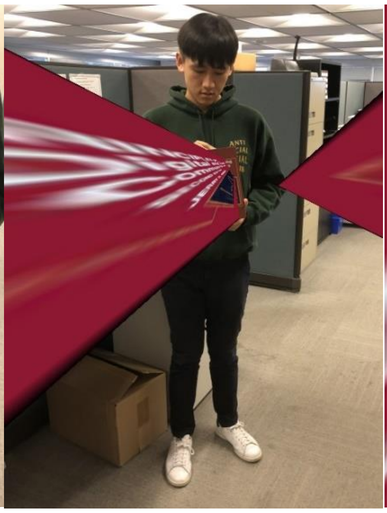
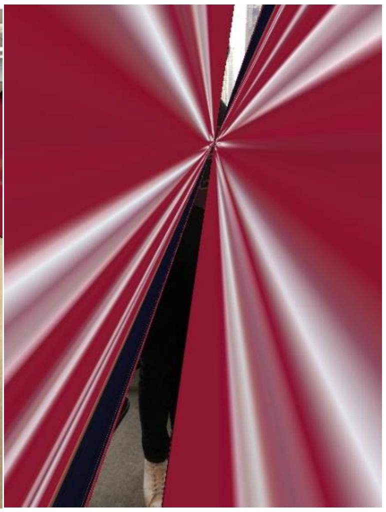
Affine:



Homography:

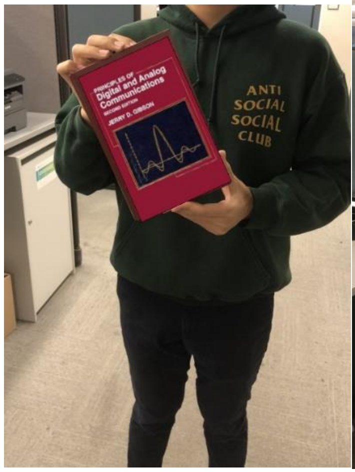
CSC420 Assignment 3
1002077726
Guanxiong Liu

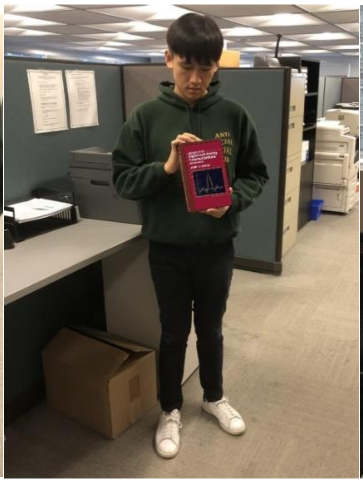
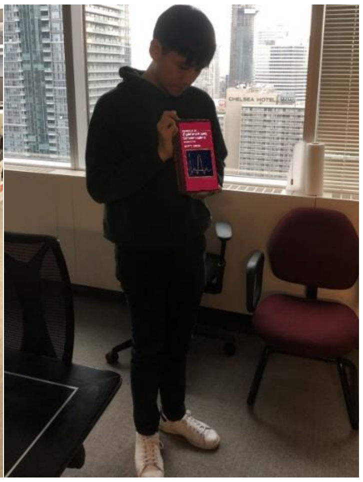


So, I manually set the percent of inliers to 50% for all three, this might be more accurate because visual might be deceiving. A lot of point should not be counted as inlier, but I accidentally counted them as inliers because they are parallel and seem to match the right place. However, when the matches are more crowded, even one-pixel scale matters a lot. Here are the results when I set all of them 50% inliers. ie. The actual inliers are way less than it appears.

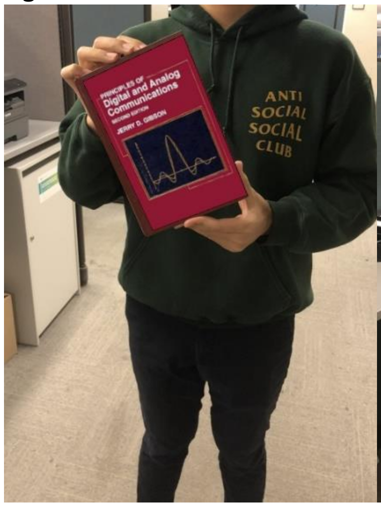
Affine:

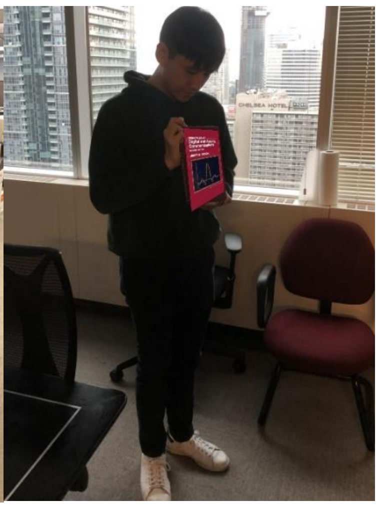


Homography:

CSC420 Assignment 3
1002077726
Guanxiong Liu

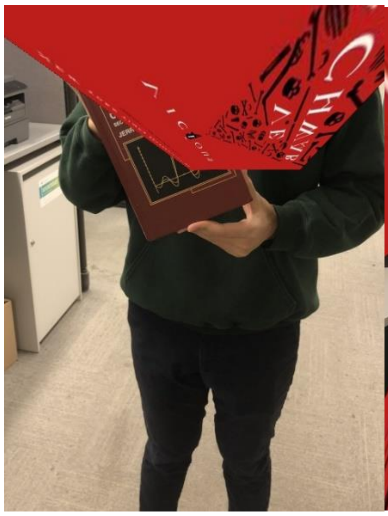
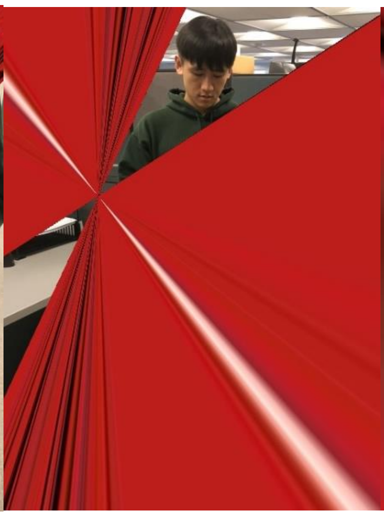
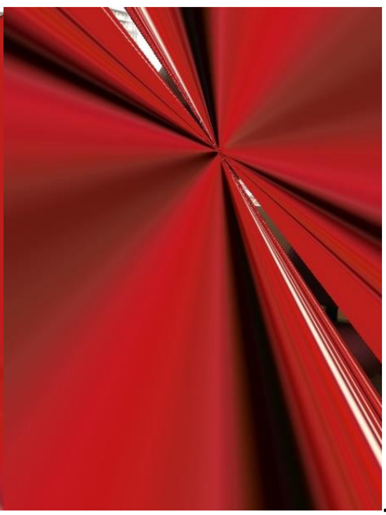




Conclusion for C and D:
RANSAC is more likely to success when you have a lot of data points and run enough iterations. The more data points, the more stable of the algorithm. Just like flip a coin 10 times, you may not get 50% of time of heads and 50% of time of tails, but when you increase number of trials to 100000, the chance would get close to 50:50. Enough iterations ensure we can find the best model among the trials. The difference between Affine and Homography is that Affine only requires 3 pairs of point to determine the transformation matrix but Homography requires 4 pairs. Generally, Homography would require more iterations than Affine to find the best matrix in the same condition (i.e. P=0.99, p=0.5)

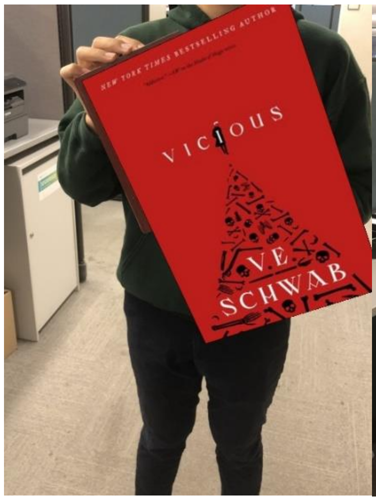
E. I use both previous best matrix and find homography using second book cover and the picture I am holding the book. Here are the results.

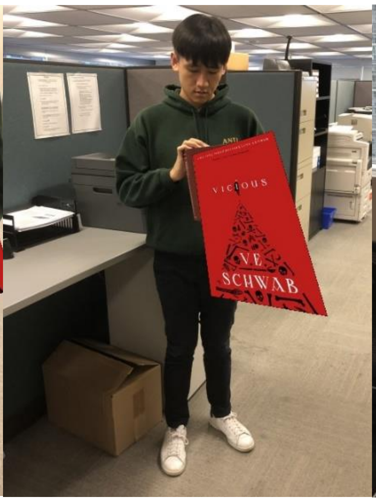
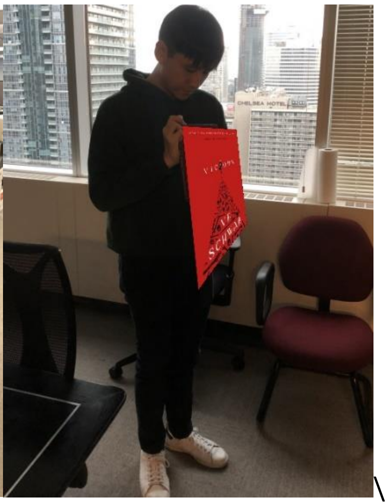
Using RANSAC homography:




.

CSC420 Assignment 3
1002077726
Guanxiong Liu

Using previous best matrix:





\

As we can see above, the first set of images make sense because there are no matches so RANSAC can only select the “best model” among non-matched points. The second set of images are recovered well just the size of the book cover is a bit off because the second book cover is not as large as the first book cover.

3. I use project: https://github.com/SymenYang/Vanish-Point-Detection to help me find the vanish points, I can certainly extend the image and do a visual localization, but for the sake of accuracy, I decide to use open source algorithm to determine three vanish point. The photo I shot is 3024x4032

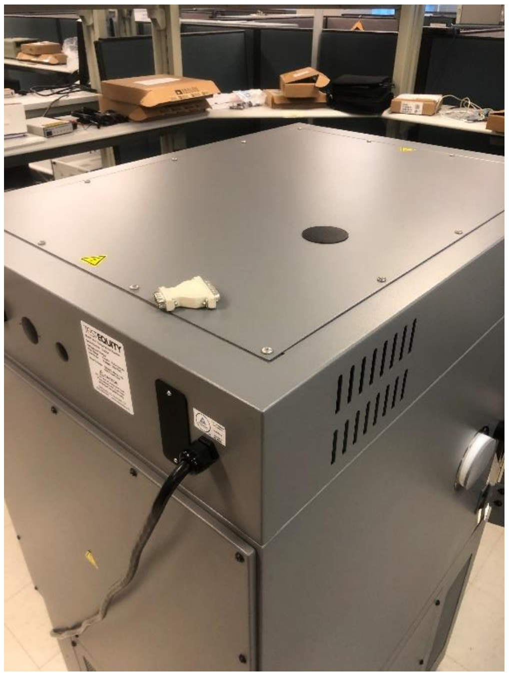
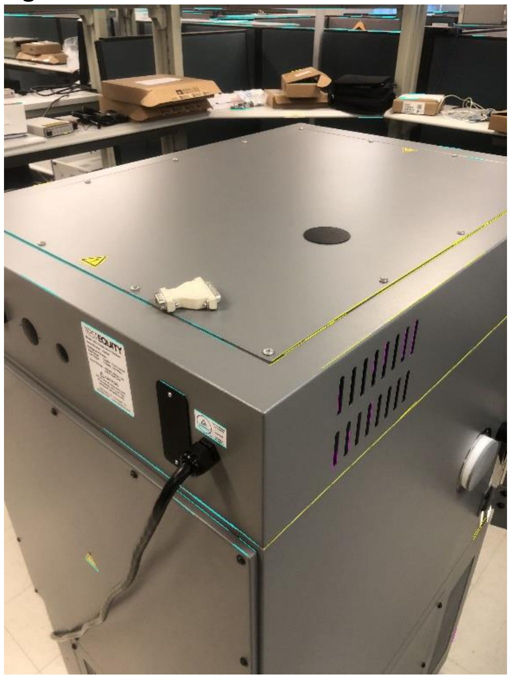


Photo I shot

CSC420 Assignment 3
1002077726
Guanxiong Liu



The output of the open source algorithm

I shot a photo and try to find 3 vanishing points by extending the parallel line in 3D but they will converge in 2D. The intercepts are the vanishing point. According to textbook *Multiple View Geometry in Computer Vision* (pag. 226), we have intrinsic matrix K:

$$\mathrm{K} = \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix}$$

Three vanish point are

$$\mathrm{V}_1 = \begin{bmatrix} -2495 \\ 142 \\ 1 \end{bmatrix} \; V_2 = \begin{bmatrix} 4846 \\ 62 \\ 1 \end{bmatrix} \; V_3 = \begin{bmatrix} 1849 \\ 5729 \\ 1 \end{bmatrix}$$

$$\mathrm{W} = \begin{bmatrix} w_1 & 0 & w_2 \\ 0 & w_1 & w_3 \\ w_2 & w_3 & w_4 \end{bmatrix}$$

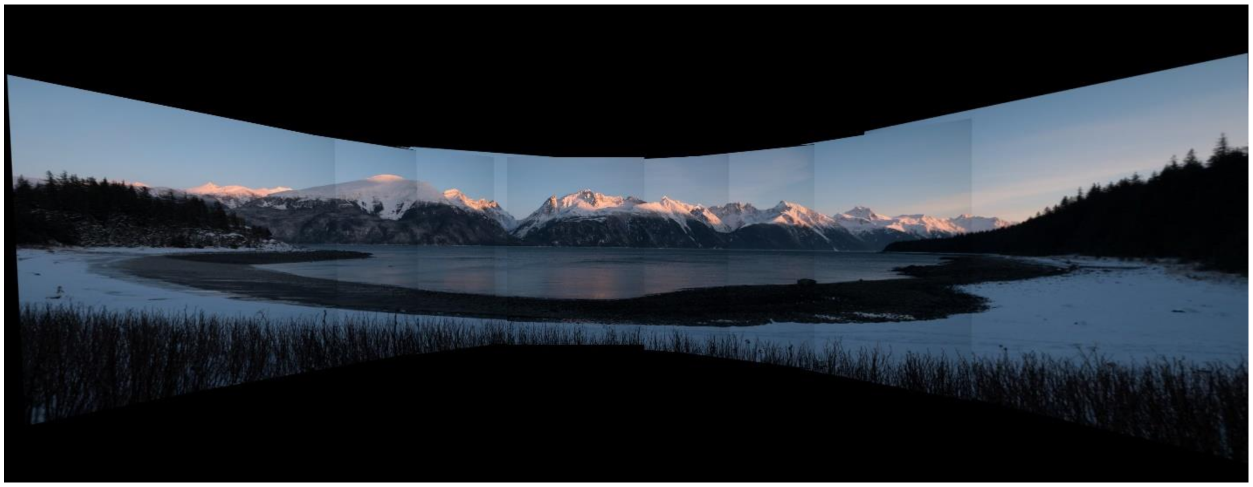
$$\mathrm{V}_1^T * w * V_2 = 0, \mathrm{V}_1^T * w * V_3 = 0, V_3^T * w * V_2 = 0$$

Each row can be computed as [xa*xb+ya*yb,xa*zb+xb*za,ya*zb+yb*za,za*zb] where $(\mathrm{a.b}) \in \{(1,2),(1,3),(3,2)\}$, then stack rows together. Then we just need to find the null space of the matrix, that is w and solved K by a Cholesky factorization as $\mathrm{w} = \left(\mathrm{KK}^{\mathrm{T}}\right)^{-1}$.

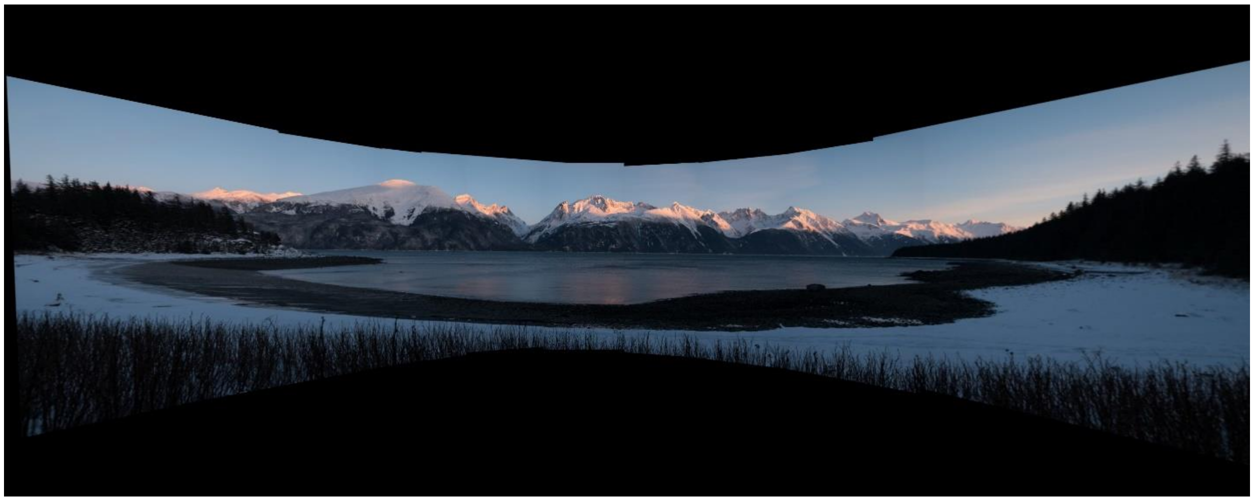
$$\text{Finally, we have } \mathrm{K} = \begin{bmatrix} 2773.716 & 0 & 1812.942 \\ 0 & 2773.716 & 2420.260 \\ 0 & 0 & 1 \end{bmatrix}$$

CSC420 Assignment 3
1002077726
Guanxiong Liu

4.



Without Blending.



With Blending