CSC420 a3

Guanchen Zhang

October 2018

1 Part A

The picture of the door in an oblique view is shown below :

To estimate the width and height of the door from the oblique, we need to calculate the homography matrix by using the pixel locations of the four corners of the paper in oblique view (x_p, y_p) (where subscript p stands for paper)

and those in parallel view (x'_p, y'_p) (in this case A4 paper with 210*297mm).After having homography matrix H, we are able to estimate the actual height and width of the door by plugging the the pixel locations of the four corners of the door to the equations. Since we have 8 unknowns, we need at least 4 matches (8 rows in matrix A)to solve the equation. The h matrix is the eigenvector of $A^T A$ with smallest eigenvalue.

$$\begin{bmatrix} x_{p1} & y_{p1} & 1 & 0 & 0 & 0 & -x'_{p1}x_{p1} & -x'_{p1}y_{p1} & -x'_{p1} \\ 0 & 0 & 0 & x_{p1} & y_{p1} & 1 & -y'_{p1}x_{p1} & -y'_{p1}y_{p1} & -y'_{p1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{p4} & y_{p4} & 1 & 0 & 0 & 0 & -x'_{p4}x_{p4} & -x'_{p4}y_{p4} & -x'_{p4} \\ 0 & 0 & 0 & x_{p4} & y_{p4} & 1 & -y'_{p4}x_{p4} & -y'_{p4}y_{p4} & -y'_{p4} \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{221} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Since the projective mapping between 2 projection planes are the same, matrix h can be shared between the oblique view in which door and paper is a quadrilateral and parallel view in which door and paper are perfect rectangle. Therefore, having the pixel locations of four corners of door (x_d, y_d), we can get the real size of the door by using the equations below for each point:

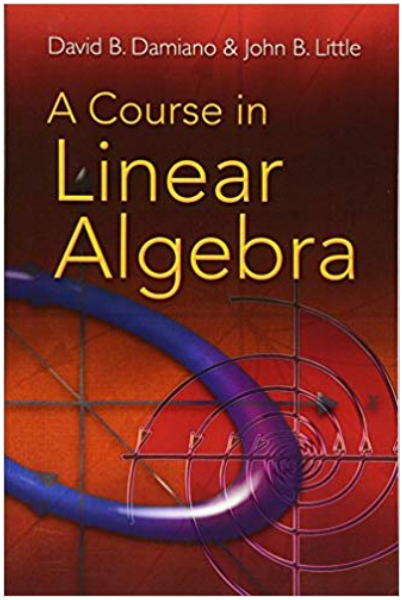
$$x'_{di} = \frac{h_{00}x_{di} + h_{01}y_{di} + h_{02}}{h_{20}x_{di} + h_{21}y_{di} + h_{22}}$$

$$y'_i = \frac{h_{10}x_{di} + h_{11}y_{di} + h_{12}}{h_{20}x_{di} + h_{21}y_{di} + h_{22}}$$

The estimated height of the door is $1185.77115496mm \approx 1.18m$ and width is $558.510067723mm \approx 0.56m$. The estimated result is very close to the actual height of $1.1m$ and width of $0.59m$. The discrepancy may come from the fact that I estimated the pixel locations in the oblique picture by hand so that the location may not be accurate.

2 Part B

Note: to avoid a large assignment file, I resize every image to 640*480 therefore it is hard for SIFT to detect keypoints and find matches due to abundance of information in each pixel. In general, all the methods work better when I used uncompressed/resized images as input.
The input images are listed below:



Book cover

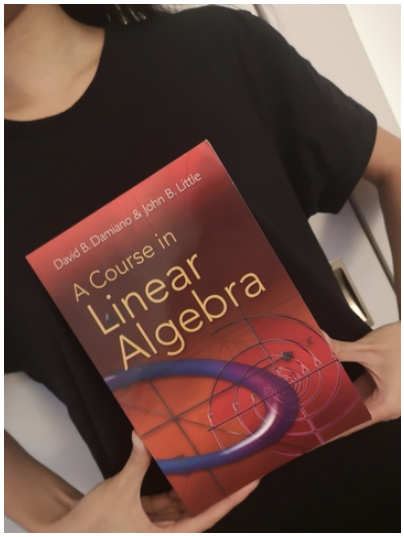


image 1

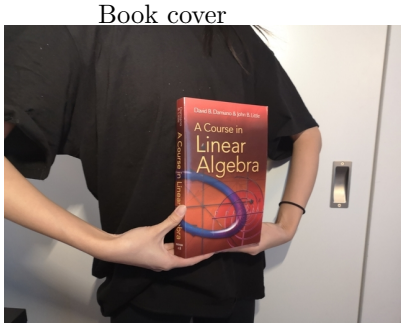
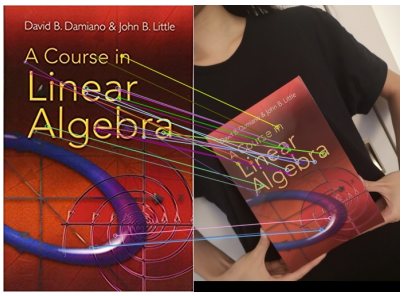


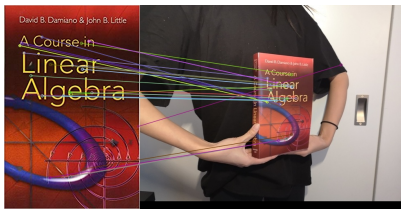
image 2

image 3

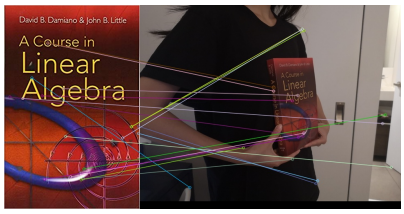
a.



sift from image 1



sift from image 2



sift from image 3

b.

Note: The point matches between each image and book cover are along with the Question 2(d), so the percentage of outliers mostly reflects the output image from Question 2(d)

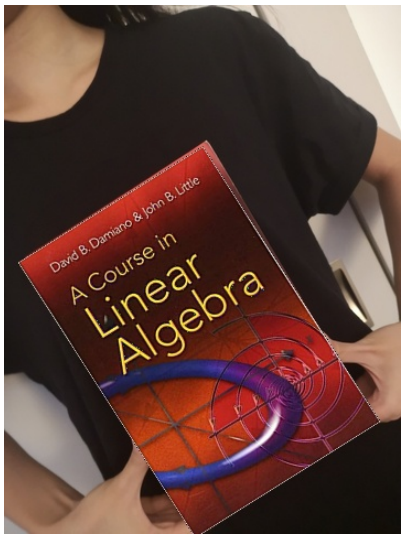
For affine, we need at least 3 matches so k=3.

number of mathces	number of outlier	percentage of outliers	number of iterations
21	1	0.05	3
22	4	0.18	6
18	16	0.89	3458

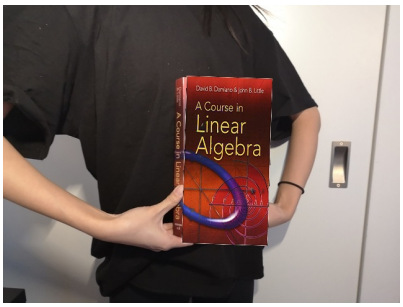
For projective transformation, we need at least 4 matches so k=4.

number of mathces	number of outlier	percentage of outliers	number of iterations
21	1	0.05	3
22	4	0.18	8
18	16	0.89	31452

c.



affine from image 1

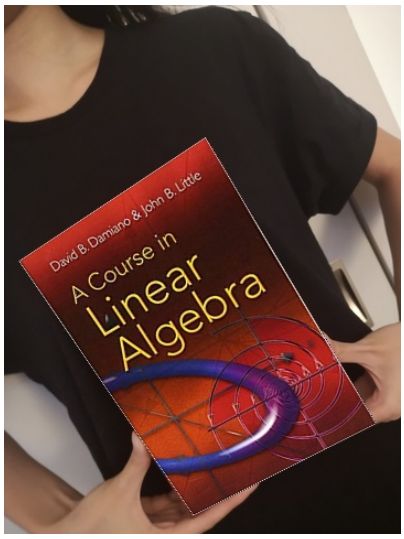


affine from image 2

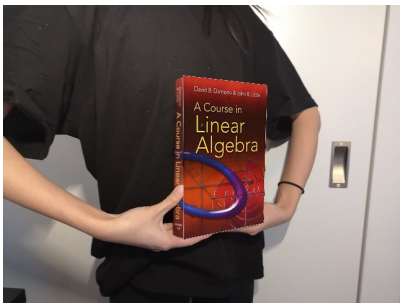
affine from image 3

The method works well with img1 that has only in-plane rotation and fails with img2 and even worse with img3. An affine transformation is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation). Therefore, The reason that img2 failed is that the outcome is Parallelogram. Besides, the reason the img3 failed even badly is that the input image is compressed too much and lack of matches between bookCover and img3. Moreover, the matching algorithm generally works better when the background is simpler. In this case, The img3.jpg has more complex background thus includes more potential keypoints.

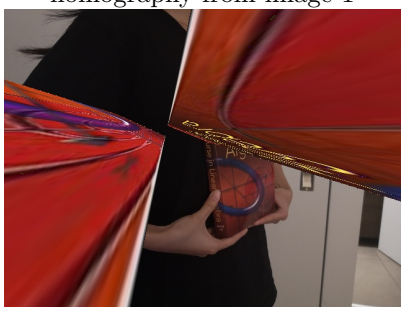
d.



homography from image 1



homography from image 2

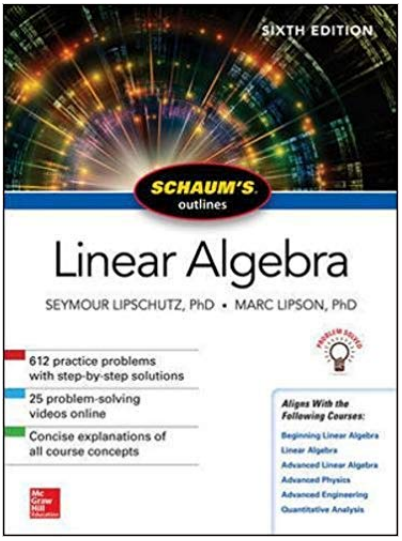


homography from image 3

The methods works very well in most cases(img1 and img2) because projective transformation only preserves straight line rather parallelism. The methods does not work so well with img3 but I suppose the major issue is due to the compressed image and lack of matches. To support my hypothesis, I did another run with a uncompressed image and the outcome is worse than the outcome for img2 but better than the one I attached in my report.

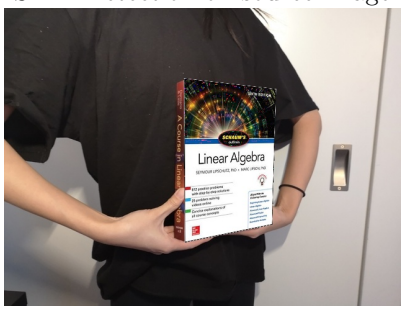
A homography, is a matrix that maps a given set of points in one image to the corresponding set of points in another image. An affine transform generates a matrix to transform the image with respect to the entire image. It does not consider certain points as in the case of homography. Hence in affine transformation the parallelism of lines is always preserved.

e.

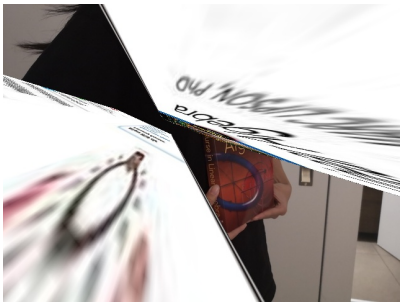


SIFT Detection on source image

SIFT Detection on reference image



SIFT Detection on source image



SIFT Detection on reference image

We calculated the projective transformation matrix already by using the book.jpg hence the relationship between reference image (book.jpg and anotherBook.jpg) and oblique view image, so we can directly applied the homography matrix to the anotherBook.jpg to get a oblique view of anotherBook. Then we can just copy the outcome to the original im1.jpg,im2.jpg and im3.jpg.

If book.jpg is not given, we can only get the projective transformation matrix by comparing im1.jpg for example to anotherBook.jpg to find matches. In most cases, we are unable to match 2 books with different cover becasue most of points will be outliers.

3 Part C

Suppose that it is known that the camera has zero skew and that pixels are square, I recover the camera intrinsic K of my phone with resolution of 500*500 (after resize) from three orthogonal vanishing points

To acquire a picture with 3 vanishing points, the picture must be taken from a view with 3 different sets of parallel 3D lines, because all lines with the same 3D direction intersect at the same vanishing point.
The image of the absolute conic is the conic $w = (KK^T)^{-1}$
The conic w is a device for representing orthogonality in an image.
The vanishing points v_1, v_2 of 2 perpendicular world lines satisfy $v_1^T w v_2 = 0$. THis means that the vanishing points are conjugate with respect to w. Assuming zero skew and square pixels, we have the constraint that $w_{12} = w_{21} = 0$ and $w_{11} = w_{22}$

The plan is:
(1) take a picture with three sets of parallel lines in the scence with each set having direction orthogonal to the others
(2) Warp the image into a larger image frame so that we can find the vanish point by drawing intersection of parallel lines. Find all three vanishing point and get their location respectively
(3) In the case of square pixels w has the form.

$$\text{w=} \begin{bmatrix} w_1 & 0 & w_2 \\ 0 & w_1 & w_3 \\ w_2 & w_3 & w_4 \end{bmatrix}$$

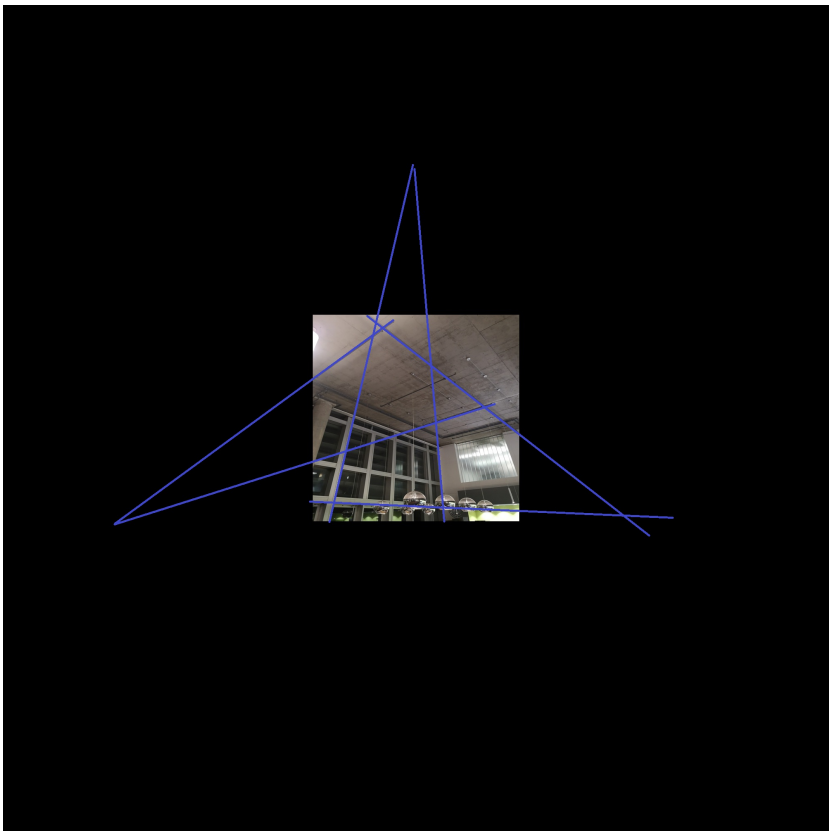
(4) Each pair of vanishing points v_i, v_j generates an equation $v_i^T w v_j = 0$. The constraints from the three pairs of vanishing points are satcked together to form an equation Aw=0 where

$$\text{A=} \begin{bmatrix} x_1x_2 + y_1y_2 & x_1z_2 + z_1x_2 & x_1y_2 + y_1x_2 & z_1z_2 \\ x_1x_3 + y_1y_3 & x_1z_3 + z_1x_3 & x_1y_3 + y_1x_3 & z_1z_3 \\ x_3x_2 + y_3y_2 & x_3z_2 + z_3x_2 & x_3y_2 + y_3x_2 & z_3z_2 \end{bmatrix}$$

(5) The matrix K is obtained from $w = (KK^T)^{-1}$ by Cholesky factorization of w, followed by inversion

$$\text{K=} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 428.10026913 & 0. & 246.99620234 \\ 0. & 428.10026913 & 68.26644387 \\ 0. & 0. & 1. \end{bmatrix}$$

Therefore, the focus length is 428.1 and principle point locates at (247.0,68.3)



4 Part D