Computer Vision Homework Assignment 3
By Luka Eerens

Homographies (30 Points)

Q1.1
Prove that there exists an H that satisfies Equation 1 given two 3 x 4 camera projection matrices M1 and M2 corresponding to cameras C1, C2 and a plane $\prod$. Do not produce an actual algebraic expression for H. All we are asking for is a proof of the existence of H.
Note: A degenerate case may happen when the plane $\prod$ contains both cameras' centers, in which case there are infinite choices of H satisfying Equation 1. You can ignore this case in your answer.

Answer:

Equation 1, in this report refers to $p \equiv Hq$ where to p is a 3D point projected in 2D from the perspective of one camera, and q from the perspective of another camera.

Now H is a 3x3 matrix that converts p into q, both of which are 3x1 matrices, therefore there are 9 elements in H.

Now the homogeneous transformation matrix H happens to be invariant to scale, and because of this property, the last element in this matrix, (bottom right) has a value of 1 when this H is normalized.

Since this last element is 1 in normalized form, there are only 8 elements, which we need to solve for, which require 8 equations.

Now in the case of a 3x4 camera projection Matrix, we are able to solve 8 equations (4 p and q values, each of which require 2 equations to solve = 8 equations in total), which means that we can solve for H, provided that its determinant is not equal to 0.

Q1.2

Prove that there exists a homography H that satisfies x_1 = Hx_2, given two cameras separated by a pure rotation. That is, for camera 1, $x_1 = K_1[I\ 0]\boldsymbol{X}$ and for camera 2, $x_2 = K_2[R\ 0]\boldsymbol{X}$.

Note that K_1 and $K_2$are the 3x3 intrinsic matrices of the two cameras and are different. I is 3 x 3 identity matrix, 0 is a 3x1 zero vector and $\boldsymbol{X}$ is a point in 3D space. R is the 3 x 3 rotation matrix of the camera.

Answer:

So let's breakdown how the perspectives can be expressed for the SAME point from the 2 different cameras, situated at different perspectives from the point in question.

From the equations that have been provided above for each camera:

$$x_1 = K_1[I\ 0]\boldsymbol{X}\ \ and\ \ x_2 = K_2[R\ 0]\boldsymbol{X}$$

Where:

$\boldsymbol{X}$ represents the ground truth position in the real world of the points. It is a [4x1] dimensions vector.

I and R represent the identity and rotation matrices respectively. Both: [3x3] matrices

$x_1\ and\ x_2$ represent the transformation (influenced by the perspective) of ground truth position from the perspectives of cameras 1 and 2 respectively. Both: [3x1] vectors.

$K_1\ and\ K_2$ represent the intrinsic matrices of each camera respectively. Both: [3x3] matrices.

0 is a vector filled with zeros. It is a [3x1] dimensions vector.

With that out of the way, we can now proceed with the proof. Start of with the equation of the first camera:

$$x_1 = K_1[I\ 0]\boldsymbol{X} \qquad \textbf{(1)}$$

In **(1)**, we can express $K_1[I\ 0]$ as a single letter variable which will simplify the equation for upcoming algebraic processes. Let us call this variable V.

$$\mathrm{V} = K_1[I\ 0]$$

$$\therefore x_1 = V\boldsymbol{X} \qquad \textbf{(2)}$$

Now if we want to express $\boldsymbol{X}$ in terms of x_1 then a bit of linear algebra is needed: So using equation **(2)** we compute the transpose of V (V^T) on both sides:

$$V^T x_1 = V^T V\boldsymbol{X}$$

Next we isolate $\boldsymbol{X}$ by itself, by multiplying booth sides by the inverse of that matrix multiplication between $V^T V$:

$$(V^T V)^{-1} V^T x_1 = (V^T V)^{-1} V^T V \boldsymbol{X}$$

And of course $(V^T V)^{-1} V^T V = I$, the identify matrix, and any matrix times the identity matrix is equal to the matrix. Therefore we obtain:

$$(V^T V)^{-1} V^T x_1 = \boldsymbol{X} \qquad (\mathbf{3})$$

Now that we have an expression for $\boldsymbol{X}$ in terms of x_1, and since $\boldsymbol{X}$ is a term found in the equation for both cameras, we are able to create an equation where both x_1 and x_2 exist. So plug in (**3**) into the equation for the second camera:

$$x_2 = K_2[R\ 0]\ (V^T V)^{-1} V^T x_1$$

Now let's re-introduce $K_1[I\ 0]$ back into the equation, replacing V everywhere and we get:

$$x_2 = K_2[R\ 0]\ (\,(\,K_1[I\ 0]\,)^T\ K_1[I\ 0]\,)^{-1}\ (K_1[I\ 0])^T x_1$$

Now $(\,(\,K_1[I\ 0]\,)^T\ K_1[I\ 0]\,)^{-1}$ can be expanded as: $(\,(\,K_1[I\ 0]\,)^T\,)^{-1}\ (\,K_1[I\ 0]\,)^{-1}$

So we get:

$$x_2 = K_2[R\ 0](\,(\,K_1[I\ 0]\,)^T\,)^{-1}\ (\,K_1[I\ 0]\,)^{-1}\ (K_1[I\ 0])^T x_1$$

We notice here that $(\,(\,K_1[I\ 0]\,)^T\,)^{-1}$ and $(K_1[I\ 0])^T$ combine to form the identity matrix, and therefore we end up getting:

$$x_2 = K_2[R\ 0]\ (\,K_1[I\ 0]\,)^{-1}\ x_1$$

We now have this shamefully ugly matrix before x_1 which need to be bought back to the other side, in order to satisfy the proof of this question that there exists a homography H that satisfies $x_1 = \mathrm{H}x_2$. So:

$$x_1 = K_1[I\ 0]\ (\,K_2[R\ 0]\,)^{-1}\ x_2$$

Where:

$$H = K_1[I\ 0]\ (\,K_2[R\ 0]\,)^{-1}$$

Q1.3

Let x_1 be a set of points in an image and x_2 be the set of corresponding points in an image taken by another camera. Suppose there exists a homography H such that:

$$x_1^i \equiv \boldsymbol{H} x_2^i \ \ (i \in \{1 \ldots N\})$$

where $x_1^i = \begin{bmatrix} x_1^i & y_1^i & 1 \end{bmatrix}^T$ are in homogenous coordinates, $x_1^i \in x_1$ and $\boldsymbol{H}$ is a 3x3 matrix. For each point pair, this relation can be rewritten as:

$$A_i \boldsymbol{h} = 0$$

where $\boldsymbol{h}$ is a column vector reshaped form $\boldsymbol{H}$, and A_i is a matrix with elements derived from the points x_1^i and x_2^i. This can help calculate $\boldsymbol{H}$ from the given point correspondences.

1. How many degrees of freedom does $\boldsymbol{h}$ have?
2. How many point pairs are required to solve $\boldsymbol{h}$?
3. Derive A_i

Answer:

Let us first rewrite $x_1^i \equiv \boldsymbol{H} x_2^i \ \ (i \in \{1 \ldots N\})$ based on $x_1^i = \begin{bmatrix} x_1^i & y_1^i & 1 \end{bmatrix}^T$:

$$\begin{bmatrix} x_1^i \\ y_1^i \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{h_1} & \boldsymbol{h_2} & \boldsymbol{h_3} \\ \boldsymbol{h_4} & \boldsymbol{h_5} & \boldsymbol{h_6} \\ \boldsymbol{h_7} & \boldsymbol{h_8} & \boldsymbol{h_9} \end{bmatrix} \begin{bmatrix} x_2^i \\ y_2^i \\ 1 \end{bmatrix}$$

Now as mentioned before, this relation can be rewritten in $A_i \boldsymbol{h} = 0$ form. If we follow the steps taken during the lecture, we end up getting:

$$A_i \boldsymbol{h} = 0$$

$$\begin{bmatrix} -x_1^i & -y_1^i & -1 & 0 & 0 & 0 & x_2^i x_1^i & y_2^i x_1^i & x_1^i \\ 0 & 0 & 0 & -x_1^i & -y_1^i & -1 & x_2^i y_1^i & y_2^i y_1^i & y_1^i \end{bmatrix} * \begin{bmatrix} \boldsymbol{h_1} \\ \boldsymbol{h_2} \\ \boldsymbol{h_3} \\ \boldsymbol{h_4} \\ \boldsymbol{h_5} \\ \boldsymbol{h_6} \\ \boldsymbol{h_7} \\ \boldsymbol{h_8} \\ \boldsymbol{h_9} \end{bmatrix} = 0$$

1. How many degrees of freedom does $\boldsymbol{h}$ have?

From this equation we can see that there are 9 elements in $\boldsymbol{h}$**.** However $\boldsymbol{h}$ when normalised has the last element that is a 1. Therefore the number of degrees of freedom is 9-1 = 8.

2. How many point pairs are required to solve for $\boldsymbol{h}$?

There are 8 unknowns (as the 9^{th} element is 1), as such we have 4 different pairs of points, each of which require 2 equations to solve (the pairs). Therefore we need 4 different pairs of points to solve for $\boldsymbol{h}$**.**

3. Derive Ai:

$$\begin{bmatrix} -x_1^i & -y_1^i & -1 & 0 & 0 & 0 & x_2^i x_1^i & y_2^i x_1^i & x_1^i \\ 0 & 0 & 0 & -x_1^i & -y_1^i & -1 & x_2^i y_1^i & y_2^i y_1^i & y_1^i \end{bmatrix}$$

Q1.4

Suppose that a camera is rotating about its center **C**, keeping the intrinsic parameters **K** constant. Let $\boldsymbol{H}$ be the homography that maps the view from one camera orientation to the view at a second orientation. Let θ be the angle of rotation between the two. Show that $\boldsymbol{H}^2$ is the homography corresponding to a rotation of 2θ. Please limit your answer within a couple of lines. A lengthy proof indicates that you're doing something too complicated (or wrong).

Answer:

The $\boldsymbol{H}$ that maps rotational change in orientation from one camera to the other can be expressed as:

$$\boldsymbol{H} = \begin{bmatrix} cos\ (\theta) & -sin\ (\theta) & 0 \\ sin\ (\theta) & cos\ (\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where $\boldsymbol{H}^2$ can be expressed as:

$$\boldsymbol{H^2} = \begin{bmatrix} cos\ (\theta) & -sin\ (\theta) & 0 \\ sin\ (\theta) & cos\ (\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} cos\ (\theta) & -sin\ (\theta) & 0 \\ sin\ (\theta) & cos\ (\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This is equal to:

$$\boldsymbol{H^2} = \begin{bmatrix} cos\ (2\theta) & -sin\ (2\theta) & 0 \\ sin\ (2\theta) & cos(2\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This shows a rotation of 2θ, therefore it proves that $\boldsymbol{H}^2$ is the homography corresponding to a rotation of 2θ.

Q1.5

Why is the planar homography not completely sufficient to map any arbitrary scene image to another viewpoint? State your answer concisely in one or two sentences.

Answer:

An arbitrary scene image will be in 3D, and this is a problem because planar homography requires the image points to all belong to the same planar surface.

If dealing with this 3D arbitrary scene, the perspective and location of the camera will change, and therefore they won't exactly be looking at the same planar surface, which means it won't be valid.

Q1.6

We stated in class that perspective projection preserves lines (a line in 3D is projected to a line in 2D). Verify algebraically that this is the case, i.e., verify that the projection **P** in x = **PX** preserves lines. *Hint: consider how to parameterize a line and show that the same thing holds after perspective projection*

Answer:

Consider this parametric expression for a line (subscript 1), as well as for the transformed line (subscript 2):

$$l_1{}^T \cdot \mathbf{X} = 0 \quad \text{and} \quad l_2{}^T \cdot x = 0$$

Knowing that project coordinates can be represented by:

$$\mathrm{x} = \mathbf{PX}$$

We know that the equations mentioned earlier can be expressed as:

$$l_2{}^T \cdot \mathbf{PX} = 0$$

Now, we can unify **P** and l_2 by taking the transpose of **P** to get:

$$l_2{}^T \mathbf{P^T} \cdot \mathbf{X} = 0$$

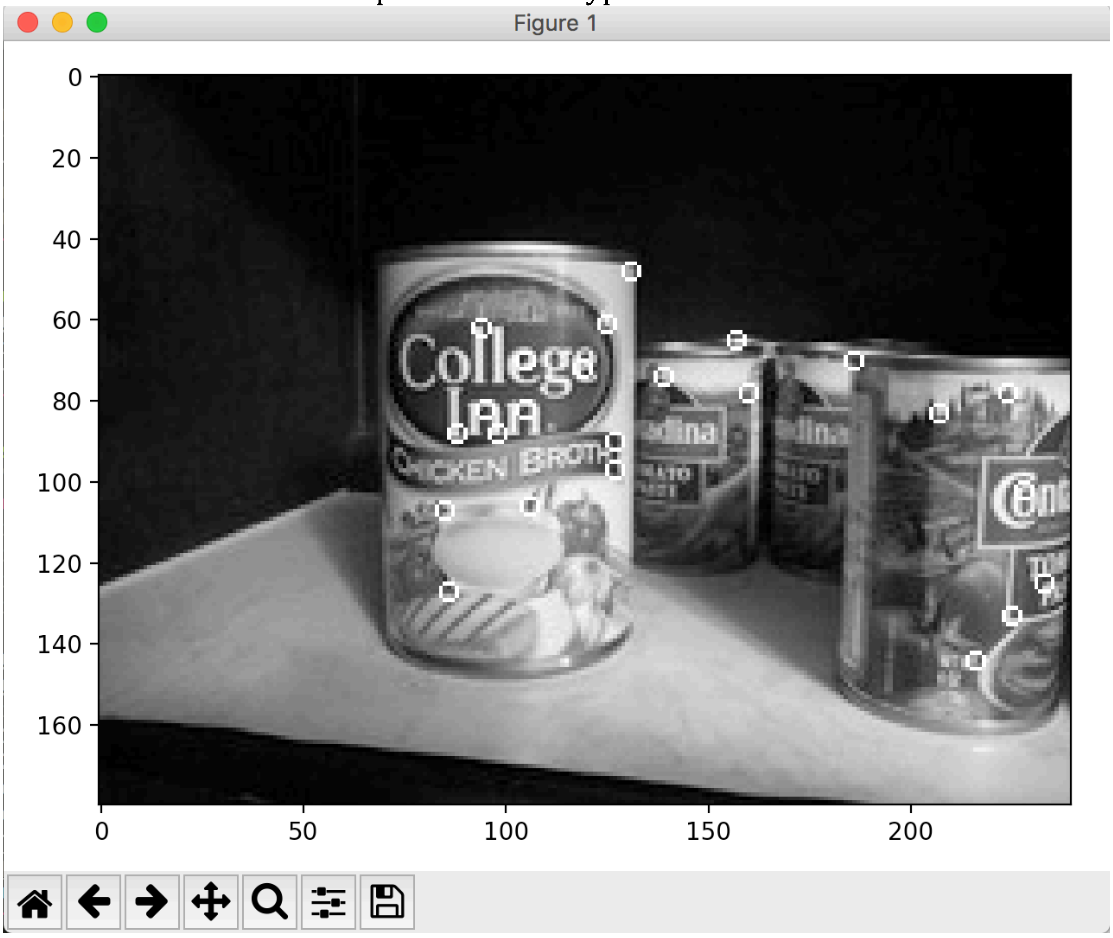
$$(l_2 P)^T \cdot \mathbf{X} = 0$$

Looking back at $l_1{}^T \cdot \mathbf{X} = 0$ we see that the projected lines are preserved as:

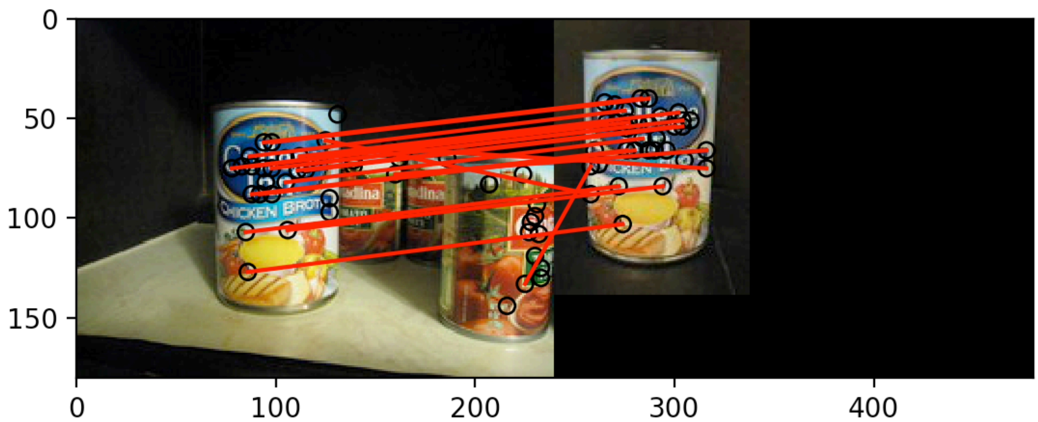
$$l_1^T = (l_2 P)^T$$

Q2.4

So here is first of all the output from the keypoint detector:



As far as plot matching, here is what I obtain when I set the ratio to 0.8:

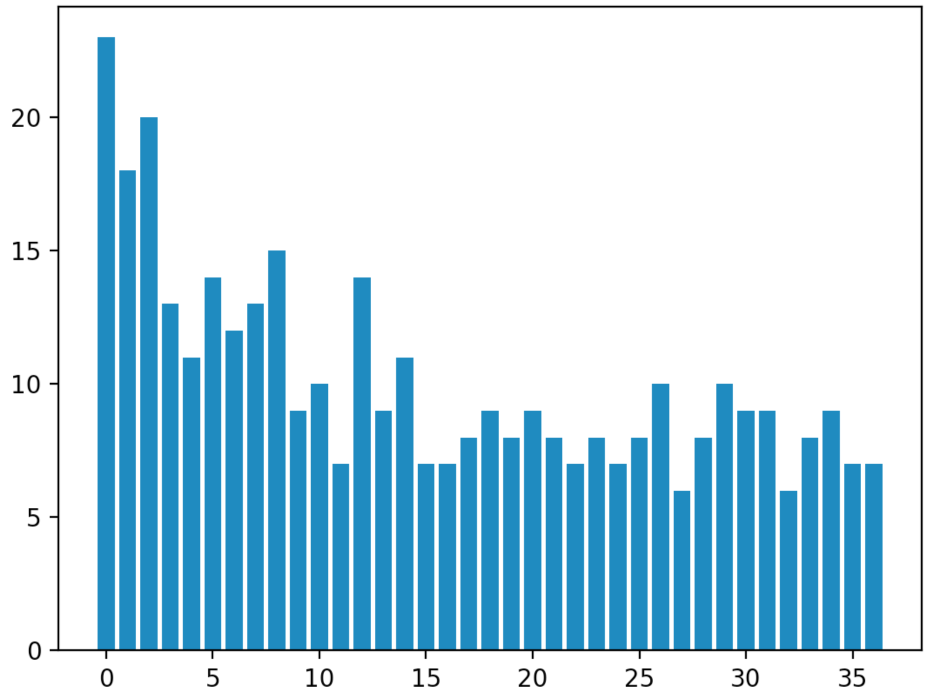


As I seem to lower this ratio, it seems that more features become matched which are not suppose to match. Increasing the ratio, decreases the number of matched features but these matches seem to more coherent.

Q2.5

Answer:

This is is the histogram for matches by each angle increment of 10 degrees. A note is that the specification of what function should be called isn't updated in Q2.py but in run_q2.py. It is there in the function argument that you declare whether to use the function from Q2.5 or Q2.6. Apologize for the inconvenience.

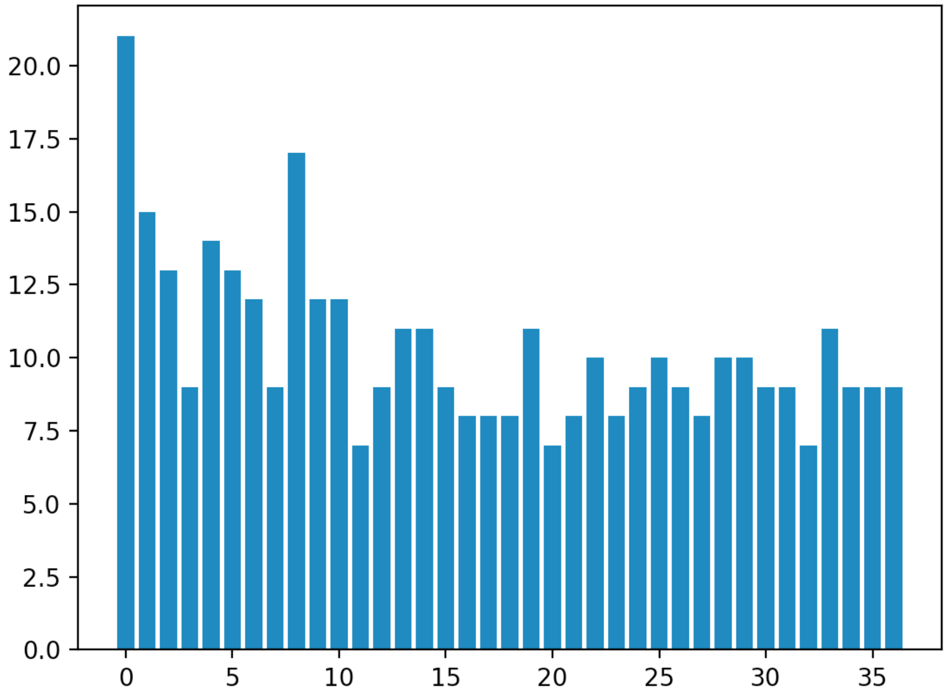


The reason why there is a downwards trend in matches as you rotate is because of several reasons, one is that the features become distorted past a certain matching threshold with each rotational transformation. You thus have the features that the system focuses on, becoming harder to detect and whose spatial relationship is changing with each rotation, such that in some cases features that would stand alone would then merge into the angled blur of other features nearby.

Q2.6

Answer:

Here is how the algorithm performed with the same histogram output:



This algorithm works better than the previous one as is visible from this graph.

For this I implemented SVD, and then matrix multiplied the coordinates of the location of the features of interest with the rotation matrix that was generated.

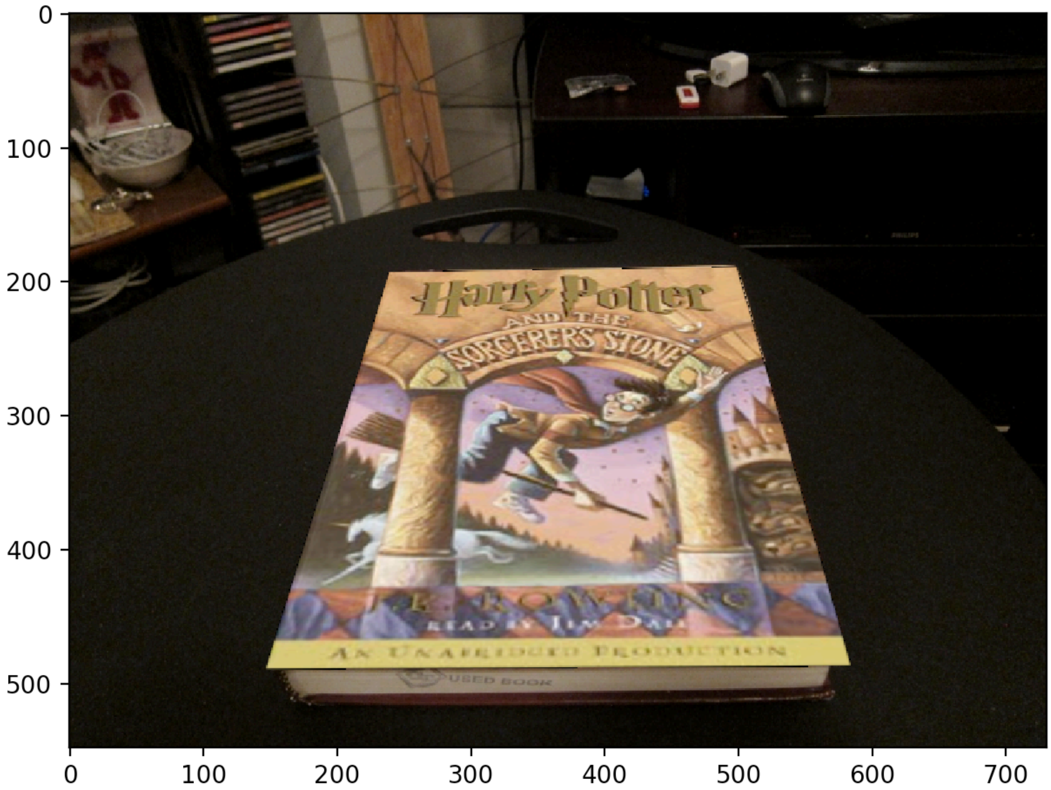
Q3.4

Answer:

The textbook was hypoterized correctly with the final H matrix being:

```
[[ 7.30718109e-01 -3.45030073e-01  2.39016886e+02]
 [-1.53690001e-02  2.22856402e-01  1.92992658e+02]
 [-2.27056649e-05 -9.20981805e-04  1.00000000e+00]]
```

With the following result:



Q4

This question was attempted, but I was not able to produce anything that I could display.

Q5

I did not try this question.