16-720 Computer Vision:

Homework 2 (Fall 2020)

Augmented Reality with Planar Homographies

Shayeree Sarkar

Q4.1)

$$x_1 = \Pi_1 P_1 x_\pi$$

$$P_1^{-1} x_1 = \Pi_1 x_\pi \quad \text{---} (1)$$

$$x_2 = \Pi_2 P_2 x_\pi \quad \text{---} (2)$$

Replacing x_π from eq^n in (2)

$$\Rightarrow x_2 = \left(\frac{\Pi_2 P_2 P_1^{-1}}{\Pi_1}\right) x_1$$

$$\Rightarrow x_2 = P_2 P_1^{-1} \left(\frac{\Pi_2}{\Pi_1}\right) x_1$$

$$\Rightarrow P_1 P_2^{-1} x_2 = x_1 \left(\frac{\Pi_2}{\Pi_1}\right)$$

$$\Rightarrow H x_2 = x_1 \left(\frac{\Pi_2}{\Pi_1}\right)$$

$$\Rightarrow \boxed{x_1 \equiv H x_2} \rightarrow \text{Hence Proved}$$

1.2 (1) Then column vector 'h' will have 8 degrees of freedom, since scale doesn't matter

1.2 (2) A minimum of '4' point pairs are required to solve 'h', since we obtain 8 homogeneous eqns with 8 unknowns for 'h'

1.2 (3) Now for estimating homography :

$$\begin{pmatrix} u^i \\ v^i \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x^i \\ y^i \\ 1 \end{pmatrix}$$

u

$$\text{Then, } u^i = \frac{h_{11}x^i + h_{12}y^i + h_{13}}{h_{31}x^i + h_{32}y^i + h_{33}} \quad \text{— (1)}$$

$$\text{Then, } v^i = \frac{h_{21}x^i + h_{22}y^i + h_{23}}{h_{31}x^i + h_{32}y^i + h_{33}} \quad \text{— (2)}$$

From eqn (1) :

$$h_{11}x^i + h_{12}y^i + h_{13} - h_{31}x^i u^i - h_{32}y^i u^i - h_{33}u^i = 0 \quad \text{— (3)}$$

From Eqn (2) :

$$h_{31}x^i v^i + h_{32}y^i v^i + h_{33}v^i - h_{21}x^i - h_{22}y^i - h_{23} = 0$$

$$\Rightarrow h_{21}x^i + h_{22}y^i + h_{23} - h_{31}x^i v^i - h_{32}y^i v^i - h_{33}v^i = 0 \quad \text{— (4)}$$

Taking eqⁿˢ (3) & (4) :
we split to find :

$$A_i = \begin{pmatrix} x^i & y^i & 1 & 0 & 0 & 0 & -x^i u^i & -y^i u^i & -u^i \\ 0 & 0 & 0 & x^i & y^i & 1 & -x^i v^i & -y^i u^i & -v^i \end{pmatrix}$$

↳ Hence Derived

& $h = \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix}$

1.2 (4) For $Ah = 0$

The trivial solution will be simply

$\rightarrow h = 0$

Also, A $\Rightarrow$ not a full rank matrix. Since only 4 pts are required to calculate 'h', A is (8×9) matrix

Here the 8 colums out of 9 are linearly independent, due to which the 8 out of 9 eigen vectors will be linearly dependent and one of 9 eigen values will be 0.

The eigen vector corresponding to 0 eigen value will map to $Ah = 0$

1.8

$$x_1 = K_1[I \quad 0]X$$

$$x_2 = K_2[R \quad 0]X$$

$$x_1 = K_1[I \quad 0]X = K_1[I \quad 0]\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = K_1\begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Hence X, can be written as:

$$X = \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} K_1^{-1}x_1 \\ 1 \end{bmatrix}$$

Now for pt x_2:

$$x_2 = K_2[R \quad 0]X = K_2[R \quad 0]\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = K_2[R \quad 0]\begin{bmatrix} K_1^{-1}x_1 \\ 1 \end{bmatrix}$$

$$\Rightarrow x_2 = K_2 R K_1^{-1} x_1$$

$$\therefore \boxed{H = K_2 R K_1^{-1}} \rightarrow \text{Hence Proved}$$

1.9

We know,

$$H = KRK^{-1}$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H^2 = KR(\theta)K^{-1}\, KR(\theta)K^{-1}$$

$$\Rightarrow H^2 = K \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} K^{-1}$$

$$\Rightarrow H^2 = K \begin{bmatrix} \cos^2\theta - \sin^2\theta & -2\sin\theta\cos\theta & 0 \\ 2\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} K^{-1}$$

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

$$\Rightarrow H^2 = KR(2\theta)K^{-1}$$

(1.5) Planar homography works only between an arbitrary (image) to another viewpoint, iff the scene/image is planar, which is not the case in an actual scenario.

Most importantly between subregion of the two images, there exists different homographies corresponding to the viewpt. on subregions of same planar.

(1.6) If we have 3 points → A, B, C &
we consider their corresponding projections in 2D space:

$$a = PA,\ b = PB,\ c = PC$$

Now in order to constraints the three points, A, B, C on the same line, we can use a linear fn, such that:

$$C = f(A, B)$$

Eg $C = A + t(B - A)$

Now if we project the above expression

$$c = PC = P * f(A, B) \rightarrow \text{in 2D space}$$

$\Rightarrow$ $c = PA + t(PB - PA) \approx c = a + (b-a)t$

Hence we can verify algebraically that in this case, projection preserves lines

2.1.1 :

A Fast Detector uses pixel density, to detect corners and requires a ring of contrasting pixels for detection. If there aren't such contrasting pixels then the detector fails in corner detection. On the other hand, the Harris Corner Detector is a corner detection operator that uses a window around each pixel and uses sum of squared difference of the pixel values when the window is shifted by a small amount in any direction. It takes differential of the corner score into account with reference to direction directly and hence is more accurate in its detection.

2.1.2 :

Filter banks as opposed to descriptors provide global image representation, which leads to loss in spatial information which why they are not a good image feature discriminator.
Coming to the BRIEF descriptors, it takes a smoothed image patch and selects a set location pairs in a unique way after which a pixel intensity comparisons is done on those location pairs and based on the intensity values a bit-string vector is created for the image locations. The feature descriptions are encoded into a series of numbers that differentiate one feature from another, and this information is transformation invariant
The GIST filter bank can be used a global descriptor for holistic representation of the scene

2.1.3 :

The nearest neighbors are defined as the keypoints with minimum Euclidean distance from the given descriptor vector. The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. The k-NN classifier takes the k nearest points and assigning the sign of the majority. We measure the performance of the descriptors using Nearest Neighbor (NN) correctness. We refer to this measure as recognition rate ρ. It simply computes the fraction of descriptors that are NN in feature space while belonging to the same real-world point, and hence being a true match

Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different.
BRIEF descriptor takes a smoothed image patch and selects a set location pairs in a unique way. Then a pixel intensity comparisons are done on these location pairs and based on the intensity values a bitstring vector is created for the image locations. These binary strings are used to match features using the Hamming distance, which basically just applies a XOR gate and bit count, helping speed up the detection process.

Hence we can conclude that for BRIEF descriptors as opposed to calculating sum of the squared distance of pixel density from the nearest neighbours, that is the conventional Euclidean distance, hamming distance makes more sense as a distance metric for BRIEF descriptors.

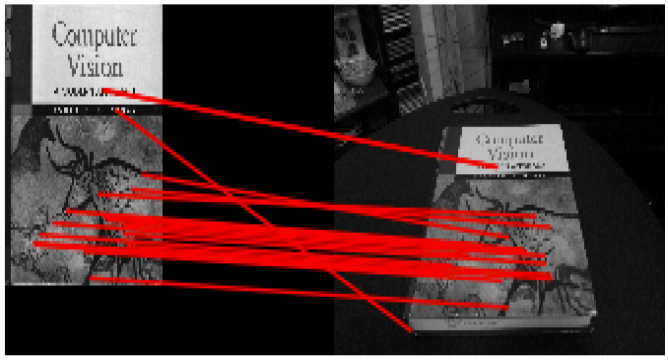
2.1.4 :

This is the result we get on doing Feature matching between cv_cover and cv_desk using the Vanilla Parameters

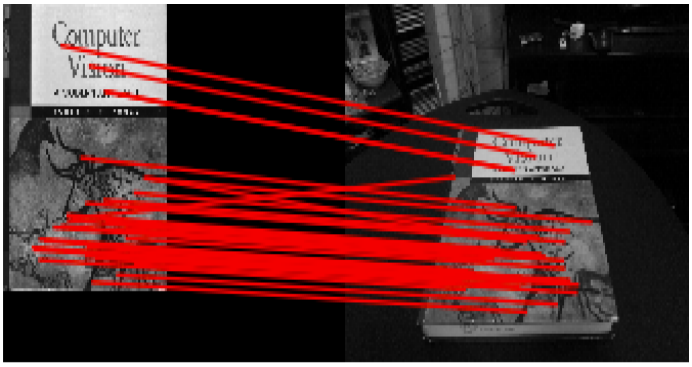
2.1.5 Ablation Study :

The following experiments were conducted wrt to varied values for sigma and ratio.

Sigma : 0.15, Ratio: 0.7



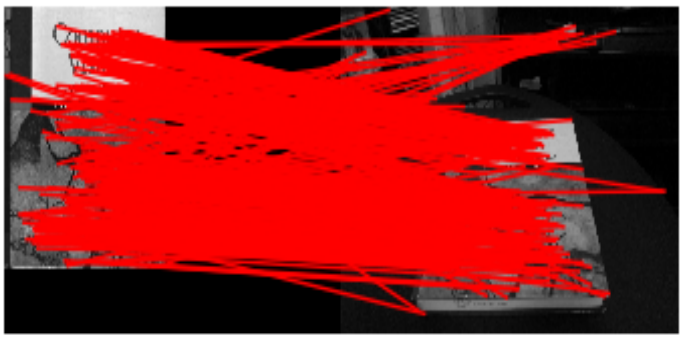
Sigma : 0.125, Ratio: 0.7



Sigma : 0.15, Ratio: 0.7

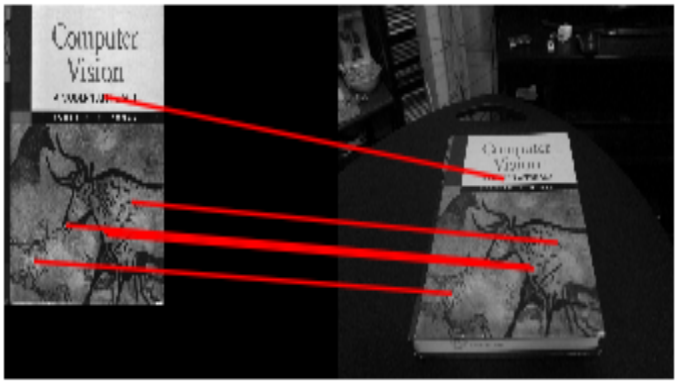
Sigma : 0.15, Ratio: 1

Sigma : 0.125, Ratio: 1



Sigma : 0.1 Ratio: 1

Sigma : 0.15, Ratio: 0.6



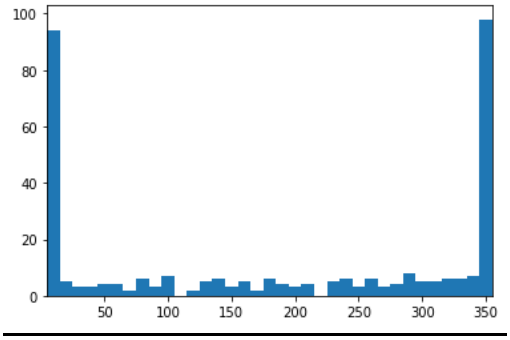
Sigma : 0.125, Ratio: 0.6

Sigma : 0.1, Ratio: 0.6

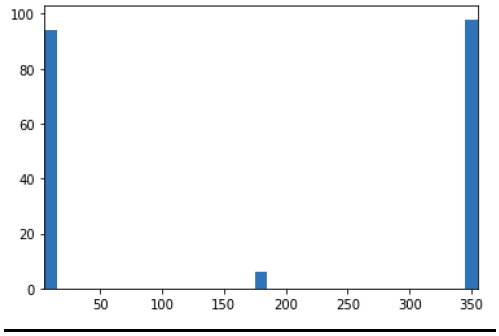
From the experiments above we can conclude that when the Ratio increases, even if sigma varies, that is increases or decreases, the no of matches between the images increases greatly. However given when the ratio isn't significantly large as in 0.6-0.7, then we notice that decreasing the value of sigma increases the no of matches. So we can conclude that ratio acts as a threshold of difference between the points, such that when ratio is high, points which are a mismatch are also considered as a match, but when the ratio is low , the only similar points are matched, which should be the case.

2.1.6:

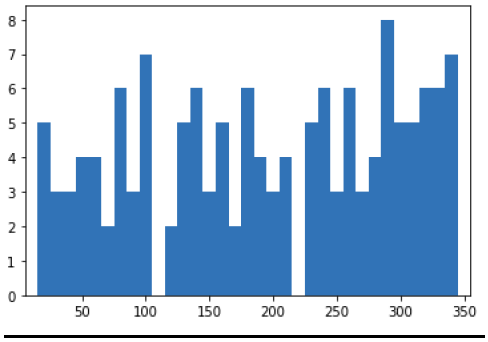
We get the following histogram on rotating the image where X axis notes the angles and Y axis denotes the no of matches



The above histogram is when the images is rotated from 0 to 360 with increments of 10 degrees in every rotation. We notice that with rotation the no of matches decreases drastically. Only when the image is rotated slightly , that is +/- 10 degrees the performance is relatively better then those with large degrees of rotation. Since BRIEF Features are extracted around the keypoints using the same pattern of point pairs selection, rotation causes decrease in no of matched points greatly



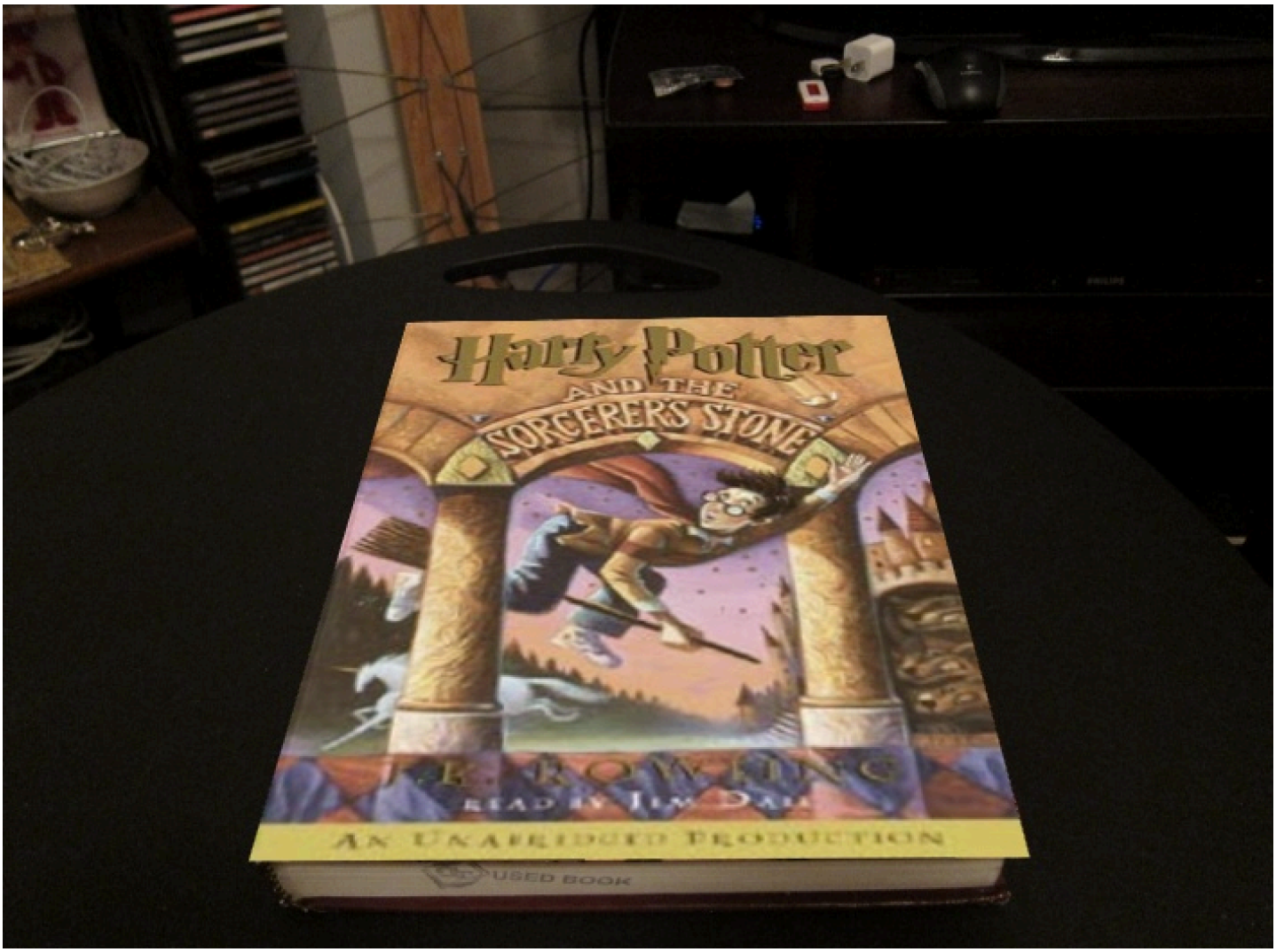
The above histogram denotes the angles 0,360 and 180 degrees where the maximum no of matches are present. Angles 0 and 360 are in the exact orientation as the original image, hence that is the case. However we also notice that when the image is inverted the BRIEF corner detector is able to hardly detect the matching points between the images.



When we don't consider the 0 and 360 degrees of rotation, we notice that the maximum no of matches is less than even 8, which reinstates that since BRIEF Features are extracted around the keypoints using the same pattern of point pairs selection, rotation causes decrease in no of matched points greatly

2.2.4

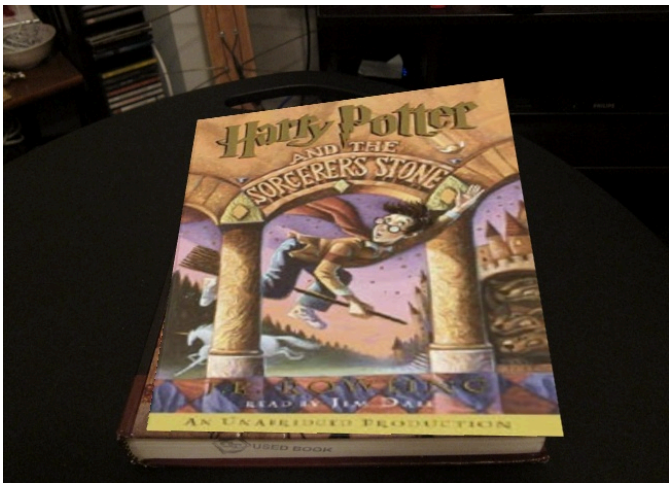
After running the HarryPotterize.py , this image given is derived :



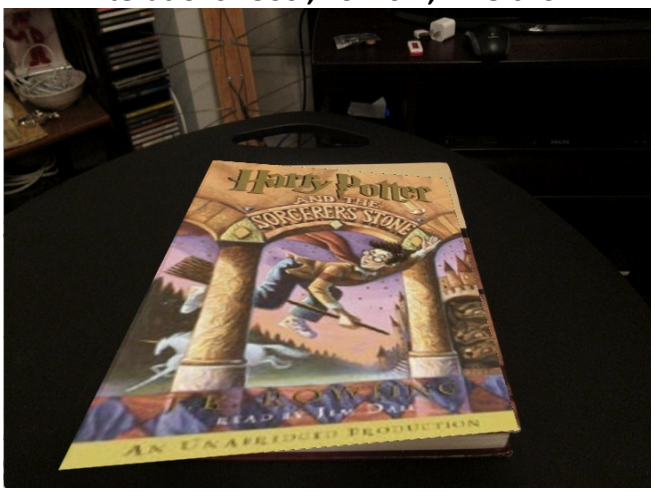
2.2.5 : Abalation Study:

The following experiments were conducted wrt to varied values for iterations and inlier tolerance values:

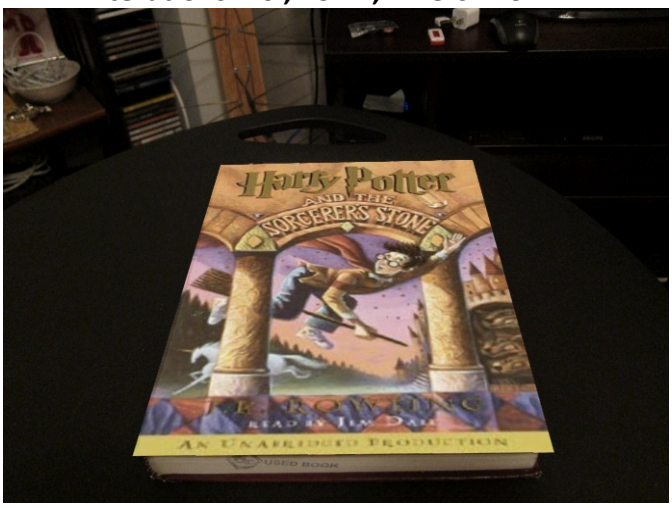
Iterations :100 , Tol: 0.1, Inliers=5



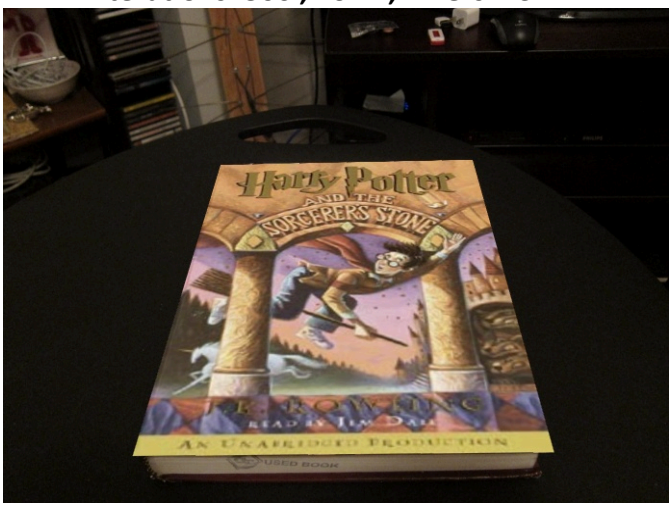
Iterations : 500 , Tol : 0.2, Inliers=6



Iterations :10 , Tol: 2, Inliers =13



Iterations :500 , Tol: 2, Inliers=13



If the tolerance is high, large amount of points are used to calculate the homography and hence fewer no of iterations will be required for doing the same.
However if the tolerance is very small, even a large no of iterations can't help in calculating the homography, there is no change in the no of inliers detected.

Extra Credit:

4.1X : On using SIFT Corner Detectors and open CV Functions for calculating the Best Homography through Ransac, in ar_ec.py, I was able to achieve a speed of 32 FPS for the video file

Citation:

1. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html
2. https://www.google.com/search?q=hamming+distance&rlz=1C5CHFA_enUS889US890&oq=hammi&aqs=chrome.0.69i59l2j0l2j69i57j46j0j69i60.1060j1j4&sourceid=chrome&ie=UTF-8
3. https://www.uio.no/studier/emner/matnat/its/TEK5030/v19/lect/lecture_4_2_feature_matching.pdf
4. https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj2/html/alieberman3/