16-720 : Computer Vision: Homework 2 (Spring 2020) - Augmented Reality with Planar Homographies

Shih-Chuan Wang
Andrew id: shihchuw

February 22, 2020

1 Homographies

1.1 Planar Homographies as a Warp

Q1.1 Homography

x_1 and x_2 are two projected points of x_π, which lies on plane π. Each of them has corresponding projection matrix P_1 and P_2.

$$x_1 = P_1 x_\pi = \begin{bmatrix} f_1 & 0 & 0 & 0 \\ 0 & f_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 & r_4 & r_7 & p \\ r_2 & r_5 & r_8 & q \\ r_3 & r_6 & r_9 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

$$x_2 = P_2 x_\pi = \begin{bmatrix} f_2 & 0 & 0 & 0 \\ 0 & f_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_a & r_d & r_g & p \\ r_b & r_e & r_h & q \\ r_c & r_f & r_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Since π is a plane, the above equations can be reduced to $x_1 = H_1 x_\pi$ and $x_2 = H_2 x_\pi$.
Then we can inverse H_2 to get the following equation, $x_1 = H_1 H_2^{-1} x_2$.
$H_1 H_2^{-1}$ is the **H** we are trying to prove.

1.2 The Direct Linear Transform

Q1.2 Correspondences

1. **h** has 8 degrees of freedom.
2. 4 point pairs are required to solve **h**.
3.
Let $\mathbf{x_1^i} = \begin{bmatrix} x_1^i & y_1^i & 1 \end{bmatrix}^\top$ be the projected 2D point on the first camera and $\mathbf{x_2^i} = \begin{bmatrix} x_2^i & y_2^i & 1 \end{bmatrix}^\top$ be the projected 2D point on the second camera. For these two given points, we can write the above in a matrix form as follows.

$$\begin{bmatrix} x_2^i \\ y_2^i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1^i \\ y_1^i \\ 1 \end{bmatrix} \tag{3}$$

From the matrix above, we can derive the following relations between $\mathbf{x_1^i}$and $\mathbf{x_2^i}$. For inhomogeneous coordinates,

$$x_2^i = \frac{h_{11}x_1^i + h_{12}y_1^i + h_{13}}{h_{31}x_1^i + h_{32}y_1^i + h_{33}} \tag{4}$$

$$= \frac{h_{21}x_1^i + h_{22}y_1^i + h_{23}}{h_{31}x_1^i + h_{32}y_1^i + h_{33}} \tag{5}$$

Then we can rearrange the above equations to the followings,

$$x_2^i h_{31} x_1^i + x_2^i h_{32} y_1^i + x_2^i h_{33} - h_{11}x_1^i - h_{12}y_1^i - h_{13} = 0 \tag{6}$$
$$y_2^i h_{31} x_1^i + y_2^i h_{32} y_1^i + y_2^i h_{33} - h_{21}x_1^i - h_{22}y_1^i - h_{23} = 0 \tag{7}$$

Written in matrix form $\mathbf{A_i h} = 0$

$$\begin{bmatrix} -x_1^i & -y_1^i & -1 & 0 & 0 & 0 & x_1^i x_2^i & y_1^i x_2^i & x_2^i \\ 0 & 0 & 0 & -x_1^i & -y_1^i & -1 & x_1^i y_2^i & y_1^i y_2^i & y_2^i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \tag{8}$$

4. The trivial solution for **h** is all 0. The matrix **A** is not full rank, otherwise the only solution for **h** will be all 0. Therefore, matrix **A** has zero eigen value, which corresponding eigen vector is the solution of **h**.

1.3 Theory Questions

Q1.3 Homography under rotation

For camera 1, $\mathbf{x_1} = \mathbf{K_1}[\mathbf{I} \quad \mathbf{0}]\mathbf{X}$ and for camera2, $\mathbf{x_2} = \mathbf{K_2}[\mathbf{R} \quad \mathbf{0}]\mathbf{X}$. Since the last column of extrinsic matrix is all zero, we can simplify the equations to, $\mathbf{x_1} = \mathbf{H_1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ and $\mathbf{x_2} = \mathbf{H_2} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$. With the about equations we can calculate the homography **H** that satisfies $\mathbf{x_1} = \mathbf{H}\mathbf{x_2}$, which $\mathbf{x_1} = \mathbf{H_1}\mathbf{H_2^{-1}}\mathbf{x_2}$.

Q1.4 Understanding homographies under rotation

For pose 1, $\mathbf{x_1} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} = \mathbf{K}\tilde{\mathbf{X}}, \tilde{\mathbf{X}} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$, and for pose 2, $\mathbf{x_2} = \mathbf{K}[\mathbf{R} \quad \mathbf{0}]\mathbf{X} = \mathbf{K}\mathbf{R}\tilde{\mathbf{X}}$. Using the equation derived in Q1.3, we can get the the following equation,

$$x_1 = K(KR)^{-1}x_2 = KR^{-1}K^{-1}x2, \quad H = KR^{-1}K^{-1} \tag{9}$$

If we multiply two homography H, we get

$$H^2 = KR^{-1}K^{-1}KR^{-1}K^{-1} = KR^{-2}K^{-1} \tag{10}$$

Since R^{-2} means applying a rotation twice, H^2 is the homography correspoding to a rotation of 2θ.

Q1.5 Limitations of the planar homography

Since we assume the 3D points int the scene lie on the same plane, we can not map any arbitrary scene image to another viewpoint.

Q1.6 Behavior of lines under perspective projections

Using the equation

$$x = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} X \tag{11}$$

Set

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

and

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} \begin{bmatrix} X/2 \\ Y/2 \\ Z/2 \end{bmatrix} = \begin{bmatrix} m_1^T X/2 \\ m_2^T Y/2 \\ m_3^T Z/2 \end{bmatrix} = \begin{bmatrix} u/2 \\ v/2 \\ w/2 \end{bmatrix} \tag{14}$$

The mid point in the 3D world is also on the line both in the image, which means the lines are preserved.

2 Computing Planar Homographies

2.1 Feature Detection and Matching

Q2.1.1 FAST Detector

Fast detector determine the corner by considering a circle of 16 pixels around the point. If the number of the pixels, which intensities are larger than some threshold, is larger than N, the point will be considered as corner. As for Harris corner, it looks at the gradients in the image patch. FAST is faster than Harris corner, since it doesn't need to compute gradients, but to compare pixel intensity.

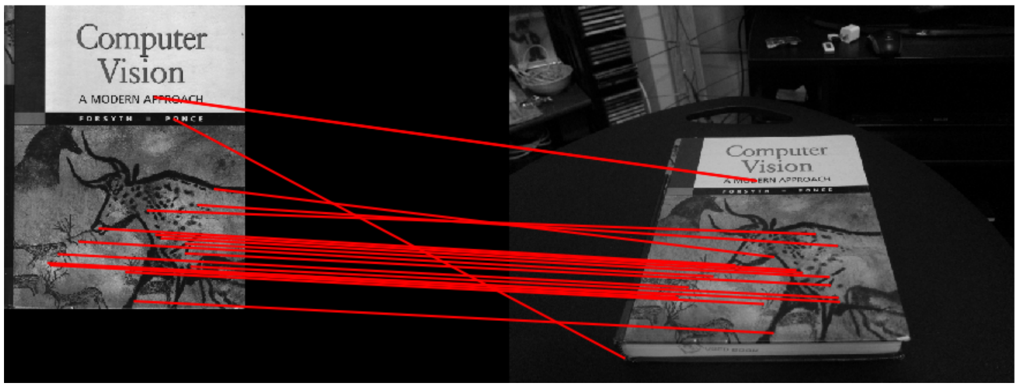
Q2.1.2 BRIEF Descriptor

BRIEF descriptor randomly choose (x,y) location pairs in the image patch and create a binary feature vector by comparing them to the pixel density. While the filterbank method we used is to stack the filter responses and extract a vector out of it. We can use Guassian filer bank as a descriptor.

Q2.1.3 Matching Methods

Hamming distance: While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different. Given two binary strings by BRIEF, we can use hamming distance as a metric to quantify how match two binary strings are.
Nearnest Neighbor: The nearnest neighbor will find the closet feature match using a distance function.

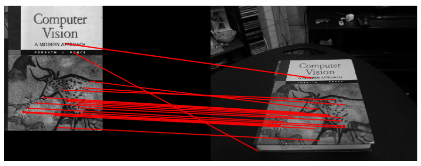
Hamming distance is more robust to outliers. Since if we are using conventional Euclidean distance, a few really big outliers can greatly bias our estimation.

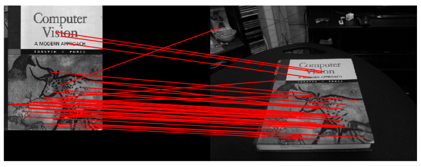


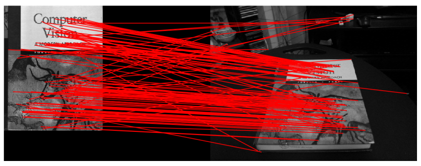
Q2.1.4 Feature Matching

Q2.1.5 Feature Matching Parameter Tuning

Sigma is the threshold used in FAST to decide whether the pixels on the circle are brighter, darker or similar. With bigger sigma, more points will be considered as corner. Ratio is the maximum ratio of distances between first and second closest descriptor in the second set of descriptors. This threshold is useful to filter ambiguous matches between the two descriptor sets. Bigger the ratio, more descriptors are considered matched.

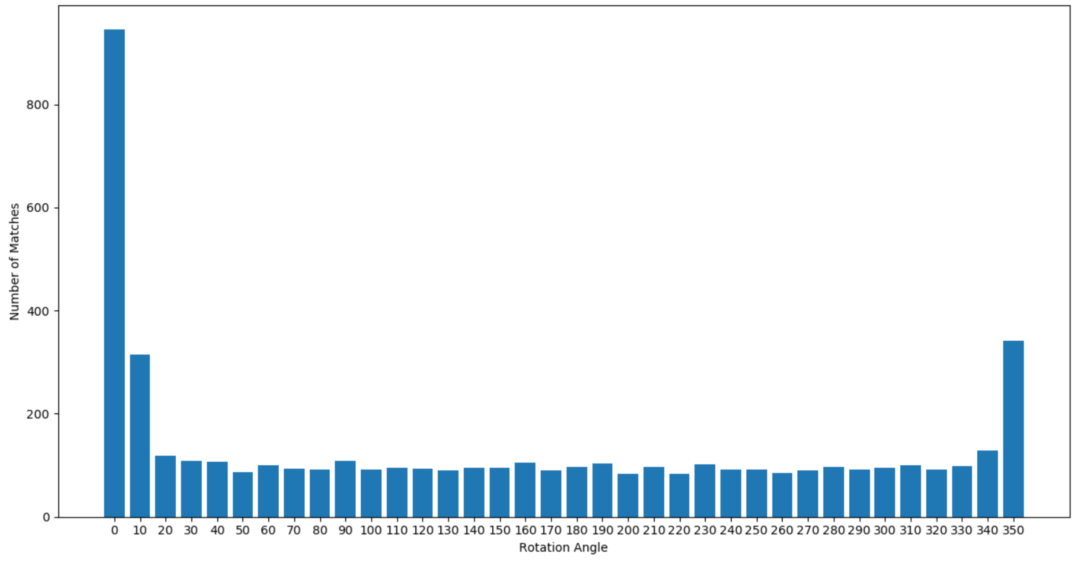

sigma = 0.15
ratio = 0.7

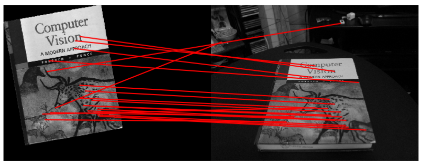

sigma = 0.12
ratio = 0.7

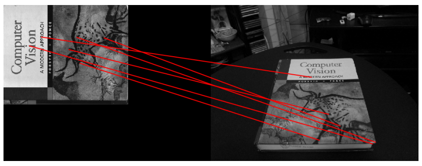

sigma = 0.15
ratio = 0.9

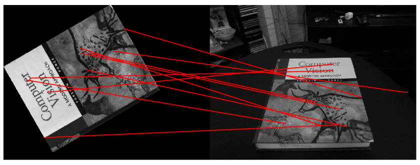
Q2.1.6 BRIEF and Rotations

The histogram and three matching results at different orientations are down below.




rotation = 10 degree


rotation = 90 degree


rotation = 120 degree

2.2 Homography Computation

Q2.2.1 Computing the Homography

The function **computeH** is implemented.

2.3 Homography Normalization

Q2.2.2 Homography with normalization

The function **computeH_norm** is implemented.

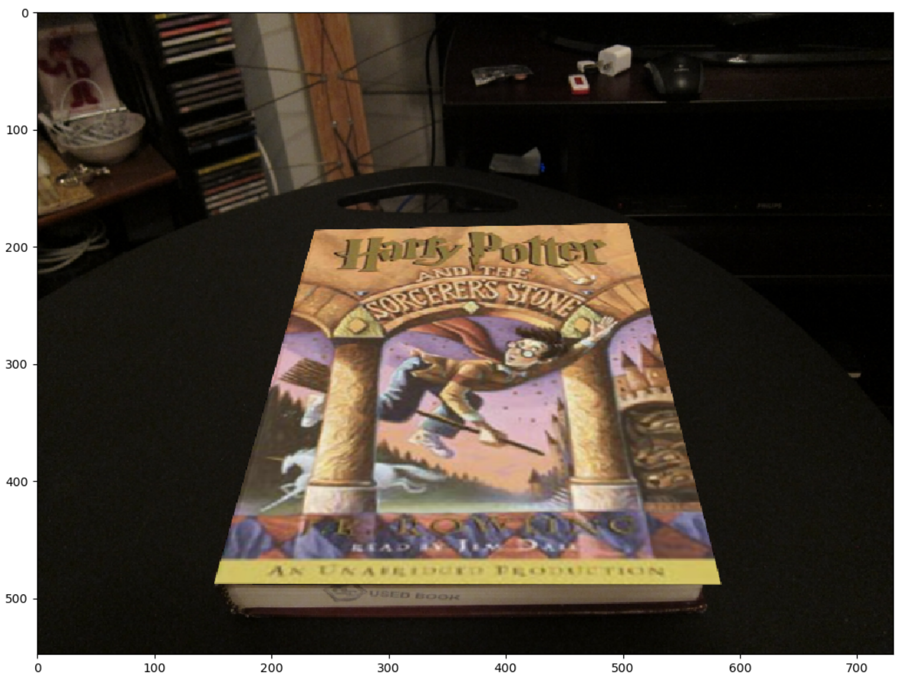
2.4 RANSAC

Q2.2.3 Implement RANSAC for computing a homography

The function **computeH_ransac** is implemented.

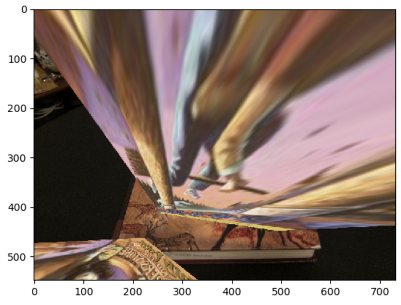
2.5 Automated Homography Estimation and Warping

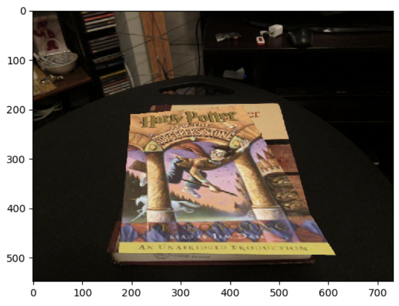
Q2.2.4 Putting it together



Q2.2.5 RANSAC Parameter Tuning

Parameter **max_iters** determines how many iterations we need to run RANSAC , and **inlier_tol** determines how far away from a model should a point be considered as inlier. Both two parameters are rather insensitive in our case. Performance starts to degrade when **max_iters** goes down to 6, and **inlier_tol** goes up to 30. The result images are down below.


inlier_tol = 2
max_iters = 6


inlier_tol = 30
max_iters = 500

3 Creating your Augmented Reality application

Q3.1 Incorporating video

The vedio result is in the /result folder.