CS 83: Computer Vision

Augmented Reality with Planar Homographies

Amittai Siavava

02/04/2024

Abstract

This project implements a step-by-step AR application.

Credit Statement

I discussed ideas with:

1. Ivy (Aiwei) Zhang
2. Angelic McPherson

However, the code and writeup are entirely my own, with reference to class notes especially on convolutions and hough transforms.

Contents

1. Theory Questions

Let $\mathbf{x}_1$ be a set of points in an image and $\mathbf{x}_2$ be the set of corresponding points in an image taken by another camera. Suppose there exists a homography $\mathbf{H}$ such that:

$$\mathbf{x}_1^i \equiv \mathbf{H}\mathbf{x}_2^i \qquad (i \in \{1, 2 \ldots, N\})$$

where $\mathbf{x}_1^i = \begin{bmatrix} \mathbf{x}_1^i & \mathbf{y}_1^i & 1 \end{bmatrix}^T$ are in homogeneous coordinates, $\mathbf{x}_1^i \in \mathbf{x}_1$ and $\mathbf{H}$ is a 3×3 matrix. For each point pair, this relation can be rewritten as

$$\mathbf{A}_i \mathbf{h} = 0$$

where $\mathbf{h}$ is a column vector reshaped from $\mathbf{H}$, and $\mathbf{A}_i$ is a matrix with elements derived from the points $\mathbf{x}_1^i$ and $\mathbf{x}_2^i$. This can help calculate $\mathbf{H}$ from the given point correspondences.

Problem 1.

How many degrees of freedom does $\mathbf{h}$ have?

$\mathbf{h}$ has 8 degrees of freedom.

Problem 2.

How many points *pair-wise* are needed to solve $\mathbf{h}$?

Since we get two equations from each point pair (one for each coordinate), we need 4 point pairs to solve $\mathbf{h}$.

Problem 3.

Derive $\mathbf{A}_i$.

Given $\mathbf{x}_1^i = \begin{bmatrix} x_1^i & y_1^i & 1 \end{bmatrix}^T$, and $\mathbf{x}_2^i = \begin{bmatrix} x_2^i & y_2^i & 1 \end{bmatrix}^T$, we have:

$$\mathbf{x}_1^i \equiv \mathbf{H}\mathbf{x}_2^i$$

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

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

Through rearranging the terms, we can write the equaton as:

$$\underbrace{\begin{bmatrix} x_2^i & y_2^i & 1 & 0 & 0 & 0 & -x_1^i x_2^i & -x_1^i y_2^i & -x_1^i \\ 0 & 0 & 0 & x_2^i & y_2^i & 1 & -y_1^i x_2^i & -y_1^i y_2^i & -y_1^i \end{bmatrix}}_{\mathbf{A}_i} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\mathbf{0}}$$

Problem 4.

When solving $\mathbf{Ah} = 0$, in essence you are trying to find the $\mathbf{h}$ that exists in the *null space* of $\mathbf{A}$. What that means is that there would be some non-trivial solution for $\mathbf{h}$ such that that product $\mathbf{Ah}$ turns out to be 0.

(i) What will be a trivial solution for $\mathbf{h}$?

The vector $\mathbf{h} = \mathbf{0}$ will be a trivial solution. While technically a solution to the system $\mathbf{Ah} = 0$, it is considered "trivial" because it does not provide any useful information about the homography.

(ii) Is the matrix $\mathbf{A}$ full rank? Why, or why not?

No, the matrix cannot be full-rank. This is because the null space of $\mathbf{A}$ is not $\{\mathbf{0}\}$. Essentially, there exists a non-trivial solution to the equation $\mathbf{Ah} = 0$, and a non-trivial solution implies that the matrix $\mathbf{A}$ is not full rank.

(iii) What impact will it have on the singular values?

If $\mathbf{A}$ is not full rank, then one of its singular values will be 0.

(iv) What impact will it have on the singular vectors?

The singular vector corresponding to the singular value 0 will be a non-trivial solution to the equation $\mathbf{Ah} = 0$, or, essentially, the components of the homography vector $\mathbf{h}$.

5. Computing Planar Homographies

Problem 1.

FAST Detector

How is the FAST detector different from the Harris corner detector that you've seen in the lectures? (You will probably need to look up the FAST detector online.) Can you comment on its computational performance vis-à-vis the Harris corner detector?

The FAST detector is different from the Harris corner detector in that it is designed to be faster. It does this by using a simple intensity comparison between pixels in a circle around the candidate corner. The Harris corner detector on the other hand, uses a more complex measure of corner-ness that involves the eigenvalues of the structure tensor. The FAST detector is faster than the Harris corner detector because it uses a simple intensity comparison, as opposed to the more complex eigenvalue computation used by the Harris corner detector. *However, FAST is less accurate and can miss corners that Harris would have detected.*

Problem 2.

BRIEF Descriptor

How is the BRIEF descriptor different from the filterbanks you've seen in the lectures? Could you use any one of those filter banks as a descriptor?

BRIEF differs from other feature descriptors such as SIFT and MOPS because it computes binary strings that represent the image region around a feature point. This is different from filter banks which use a set of filters to compute a feature vector. BRIEF compares directly the intensities of pixels in the image region around a feature point, and then uses the results to generate a binary string. On the upside, BRIEF is faster than other feature descriptors. On the downside, it is less robust to changes and noise in the image, and is not rotationally invariant.

Problem 3.

Matching Methods

The BRIEF descriptor belongs to a category called binary descriptors. In such descriptors the image region corresponding to the detected feature point is represented as a binary string of 1s and 0s. A commonly used metric used for such descriptors is called the *Hamming distance*. Please search online to learn about Hamming distance6 and Nearest Neighbor, and describe how they can be used to match interest points with BRIEF descriptors.

What benefits does the Hamming distance distance have over a more conventional Euclidean distance measure in our setting?

Hamming distance is a measure of the number of bits that differ between two binary strings, usually of equal length. Nearest neighbors finds the closest matching descriptor to an image patch given a set of reference patches and their descriptors. It may be useful to find the "closest" descriptor as an approximation for a given image patch's descriptor.

In the context of BRIEF descriptors, the Hamming distance is better suited to matching interest points than the Euclidean distance since we are dealing with a binary descriptor. We also want to match similar descriptors generated by the same world point despite changes in perspective or lighting. For this purpose, the Hamming distance is more appropriate than the Euclidean distance because it is more robust to changes in the image. Hamming distance can also be computed faster than the Euclidean since they can be simplified to bitwise or boolean operations, while the Euclidean distance requires more complicated real-number arithmetic.

Nearest neighbor search can also be useful to approximate the best descriptor.

Problem 4.

Feature Matching

Implement a function `matches, locs1, locs2 = matchPics(I1, I2)`

Use the provided function `plotMatches` to visualize your matched points amd include the resulting image in your writeup.

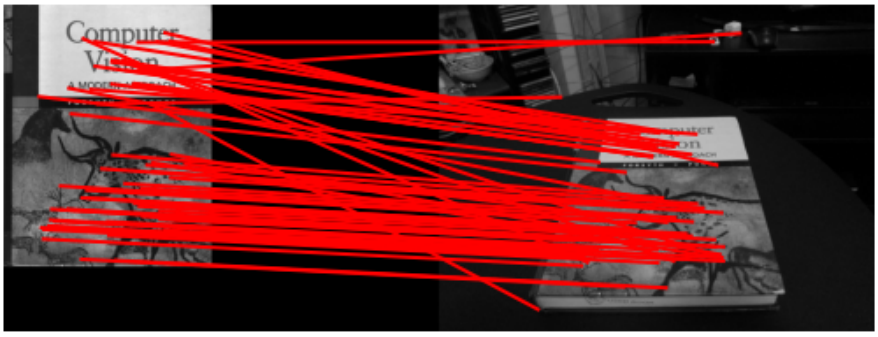

FIGURE 1. Matched points

Problem 5.

BRIEF and Rotations

Let's investigate how BRIEF works with rotations. Write a script `briefRotTest.py` that:

- Takes the `cvcover.jpg` and matches it to itself rotated [Hint: use `scipy.ndimage.rotate`] in increments of 10 degrees.
- Stores a histogram of the count of matches for each orientation.
- Plots the histogram using `matplotlib.pyplot.bar`

Include visualizations of the feature matching results at three different orientations. Explain why you think the BRIEF descriptor behaves this way.

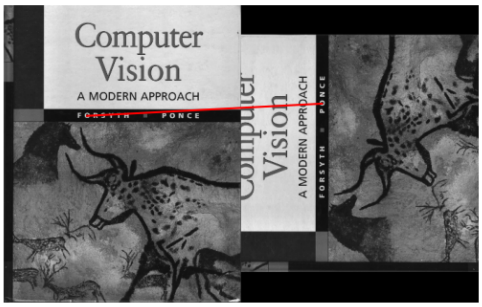

FIGURE 2. rotation = 90

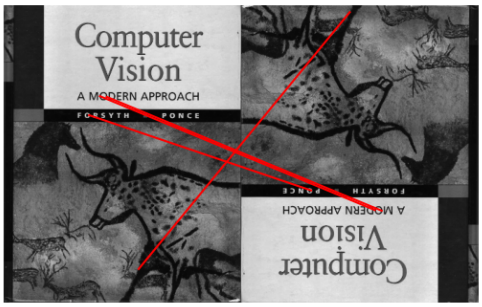

FIGURE 3. rotation = 180

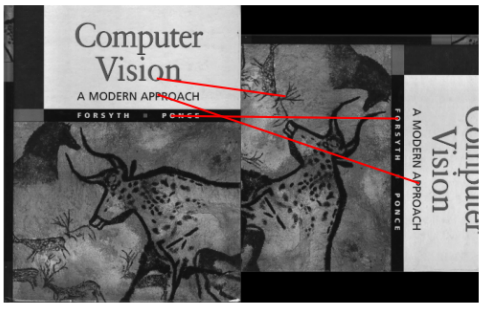

FIGURE 4. rotation = 270

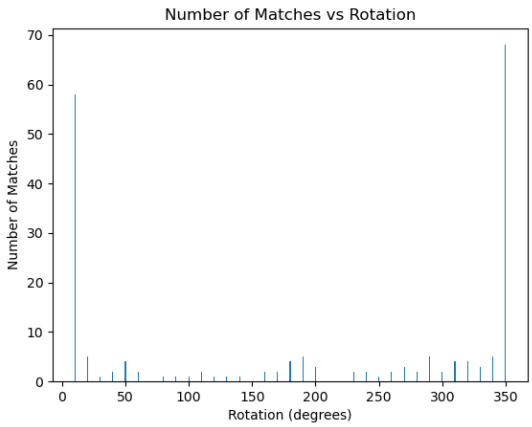

FIGURE 5. Match counts

We see the highest matches the more the two images are aligned (rotations closer to 0 or 360 degrees). This is to be expected, since the two images have a lot of similar points when less rotated.

Problem 9.

Putting it Together

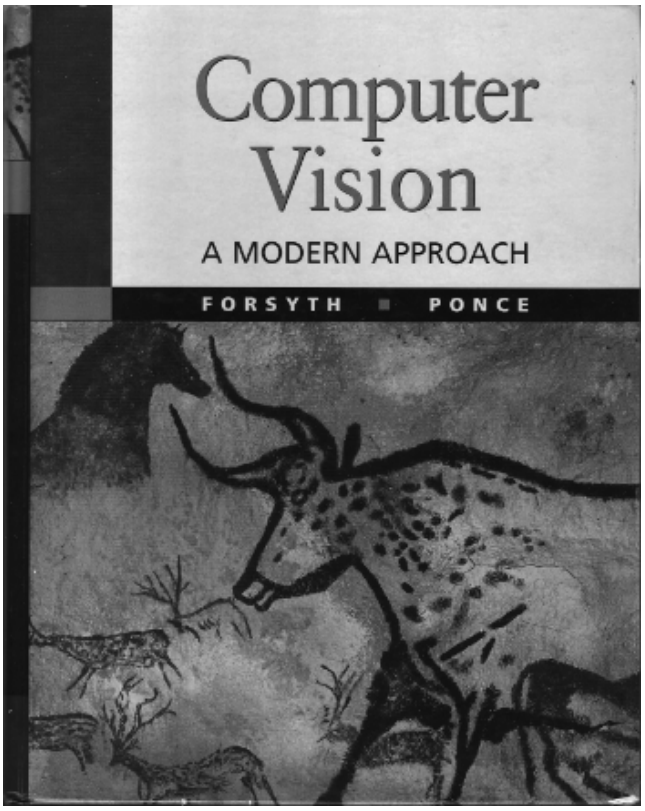

FIGURE 6. CV Book Cover

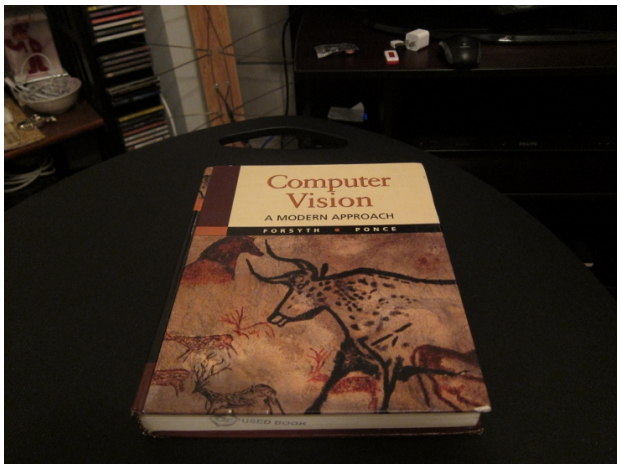

FIGURE 7. CV Book on Desk

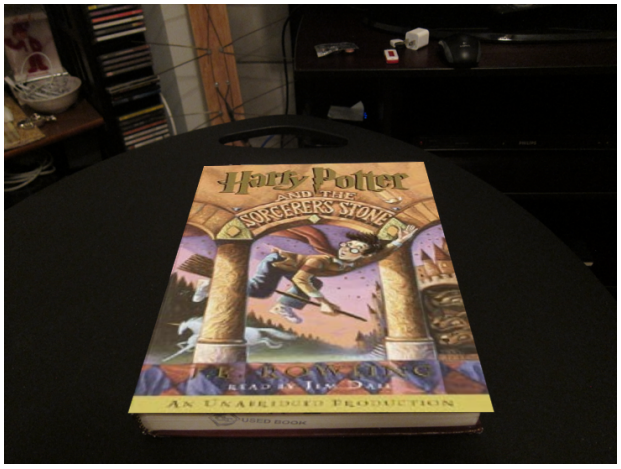

FIGURE 8. Harry Potter Book on Desk

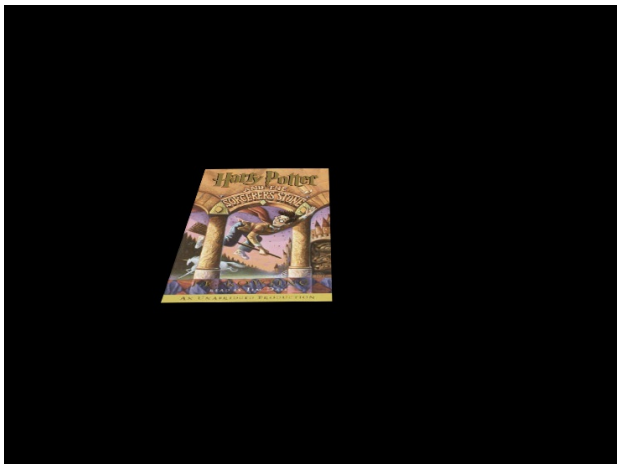

FIGURE 9. Warped Harry Potter Cover

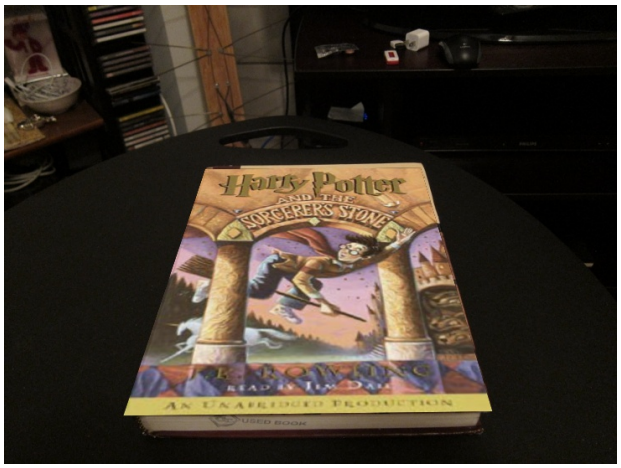

FIGURE 10. Harry Potter-rized Image

Because the original Harry Potter cover `hp_cover.jpg` is smaller than `cv_cover.jpg`, it appears smaller when we warp it into the desk frame. To make it appear the same size as the Computer Vision cover, we need to rescale it to match the size of the Computer Vision cover.

10. Extra-Credit: Panorama

These are the results I got with the given images. I noticed that a black region appears to the right of the warped right-image. I couldn't figure out why this was the case, especially since it does not occur when I use my own images. But the panorama still looks accurate.

Figure 11. Given Left Image

Figure 12. Given Right Image

Figure 13. Right Warped to Left

Figure 14. Left-Right

These are the results I got using my own images. I tried to be a bit creative and stitch together three images by recursively warping the right image to the middle image, and then warping the (middle + right) image to the left image.

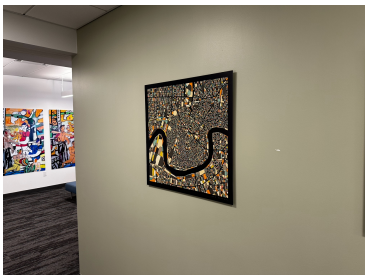

FIGURE 15. Left

FIGURE 16. Middle

FIGURE 17. Right

Left composed with Middle

FIGURE 18. Middle Warped to Left

FIGURE 19. Left + Middle

It is noticeable that the camera captured the two images with slightly different color temperatures. Still, the panorama looks accurate.

Middle composed with Right

FIGURE 20. Right Warped to Middle

FIGURE 21. Middle + Right

Left composed with Middle + Right

FIGURE 22. Right-Middle Warped to Left

FIGURE 23. Left + (Middle + Right)