Image Mosaicing

Ning Cao
cao.nin@northeastern.edu

Yihao Huang
huang.yihao@northeastern.edu

Abstract

In this project, we will be exploring a method for creating a mosaic of a sequence of images that have a significant overlap between consecutive frames. The overlap between frames allows for common features to be identified and correlated, giving us information about how the image planes transform between frames. We will be using the Harris Corner Feature detector algorithm to extract corner features, normalized cross correlation to match features across images, Random Sample Consensus (RANSAC) and least squares fitting to estimate homographies between images. Additionally, we will investigate blending techniques to address color differences between image frames and distortions in the final mosaic.

1. Introduction

In order to create a mosaic of two images, we can utilize corner detection techniques and exploit the Harris R function to identify corners in both images. By performing non-maximum suppression on the identified corners, we can extract a set of corner features from each image. To establish potential correspondences between the two sets of features, we can compute the Normalized Cross Correlation (NCC) values between them and choose the pairs with the highest NCC values.

After identifying correspondences, we can estimate a homography that maps one image onto the other. This homography can be used to warp one image and create a mosaic by blending the two images together.

In this project, we will compare the detected features in the two given images and analyze the effectiveness of the Harris corner detector. We will also evaluate the accuracy of the correspondences identified using NCC values and determine the quality of the resulting mosaic.

Code for our project can be found in our Github (. . . .), linked in the Appendix.

2. Approach

2.1. Harris Corner Detector

The Harris Corner Detector (HCD) is an algorithm that leverages the fact that corners are the intersection of two edges, and thus have gradients in the components of the edges. HCD tries to compute the sum of squared differences of neighborhoods between an original neighborhood and one with an offset. The derivations reveal that

$$F(\Delta x, \Delta y) \simeq \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x & \Delta y \end{bmatrix}$$

which is directly dependent on the middle matrix, known as the C matrix. In order to understand the strength / behavior of the matrix, it suffices to find its eigenvalues. If both eigenvalues are zero, then it is a flat area; if one is zero, then it is an edge; if both are non-zero, then it means that there is a corner of some sort. Furthermore, the actual values of λ_1, λ_2 must meet some minimum threshold (minimum strength of corner).

A common numerical approximation for this is to compute R scores according to

$$R = \det C - k(\text{ trace } C)^2 \tag{1}$$

$$R = \lambda_1 \lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2 \tag{2}$$

Since C is 2x2, detC and traceC are much easier to compute than eigenvalues of C. As such,we use this R score approximation for our implementation of HCD.

2.2. Non-Maximum Suppression

Non-max suppression is a technique used to identify local maxima in discrete data. When dealing with corner detection, corners may overlap with multiple regions, resulting in regions with high R-values. To group these regions as a single point, non-max suppression is used to select the maximum value as the representative corner feature.

The algorithm works by examining each pixel in the image. If a pixel is the maximum value within a specified window (in our implementation, a 5x5 window), then its

value is kept. Otherwise, it is set to zero. Pixels that have a larger neighbor are forced to zero, resulting in only the local maxima being retained. This process ensures that only the highest-value corners are preserved, while lower-value corners in overlapping regions are suppressed.

In summary, non-max suppression is a useful algorithm for identifying local maxima in discrete data and is commonly used in corner detection to ensure that only the highest-value corners are retained.

2.3. Normalized Cross Correlation

NCC measures the similarity between two images by computing the cross-correlation of the template and the target image, and then normalizing the result to account for variations in the brightness and contrast of the images.

$$\sum (f-g)^2 = \sum f^2 + g^2 - 2fg \tag{3}$$

The process involves sliding the template image over the target image and calculating the correlation coefficient at each position. The correlation coefficient is then normalized by dividing it by the square root of the product of the variances of the template and target images. This normalization accounts for variations in the brightness and contrast of the images, making NCC more robust to changes in lighting conditions.

The resulting NCC map shows the similarity between the template and the target image at each position, with higher values indicating a better match. The location of the maximum value in the NCC map corresponds to the position of the best match between the template and target image.

2.4. RANSAC

Random Sample Consensus (RANSAC) is an iterative algorithm that is used to estimate a model from data that is likely to contain outliers. The algorithm does not aim to optimize the least squares solution, which can be sensitive to outliers, but rather to find the best model that matches the maximum number of points within a given tolerance.

To determine the parameters of the model, a minimum of x points is required. At each iteration, RANSAC randomly selects x points from the data set without replacement and constructs a model. The algorithm then fits the remaining points to the model and counts the number of points that are within the given tolerance of the model prediction.

RANSAC continues to repeat this process until a termination criterion is met. This may include a minimum threshold of points within the tolerance or a maximum number of iterations. Once the criterion is met, the algorithm returns the best-fit model.

While RANSAC may not produce the most accurate model, it is highly effective at removing outliers from the data. By filtering the points based on whether they satisfy the best-fit model, the remaining data can be used to compute a least squares solution that is robust to outliers. Overall, RANSAC is a powerful tool for estimating models in the presence of noise and outliers.

2.5. Least Square

Least squares is a method to minimize the sum of the squared residuals between observed data points y_i and predicted values $\hat{y_i}$ from a model:

$$\sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

The method finds the line or curve that best fits the data points in a square sense. It is commonly used in regression analysis, where the goal is to model the relationship between a dependent variable and one or more independent variables.

2.6. Wrapping and Blending

After obtaining the homography, we can warp the first image to fit in the second image's domain. To determine the new image's bounding box, we pre-warp the image corners and select the smallest box that encloses all eight corners. Then, for each pixel in the new output domain, we use the inverse homography to calculate how the input images affect the output. When neither input image affects a specific output pixel, the pixel remains black. When both input images influence the output pixel, we use feathering to blend them using a weighted average, the equation is shown below:

$$p = \frac{w_1 p_1 + w_2 p_2}{w_1 + w_2}$$

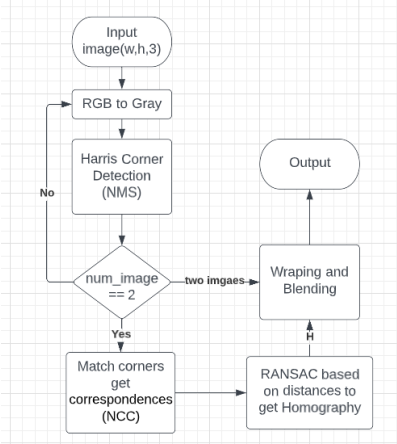
To address partial coordinates, we use bi-linear interpolation, which computes a weighted sum of the four pixels that overlap the hypothetical source pixel with floating point indices. The weights correspond to the proportion of the pixel that is overlapped by the hypothetical source pixel. This produces a weighted blend of the pixels, which is more representative of the true homography point.

2.7. Flow chat

This is the Flow Chat.

3. Experiments and Observation

This experiment aimed to investigate the effectiveness of the Harris corner detector in detecting key points in an image and how selecting the appropriate threshold can improve accuracy and efficiency. We implemented the detector on a set of input images and varied the threshold value to observe its impact on the number of detected corners and



(a) Flow Chat

Figure 1

false positives. Our results showed that selecting an appropriate threshold is crucial for achieving accurate and efficient detection. Furthermore, we explored the use of cross-correlation and RANSAC algorithm for key point matching and homography computation. We applied these techniques on the detected key points and used the homography to create a mosaic of input images with reduced artifacting. The results demonstrated the effectiveness of these techniques in producing seamless and visually appealing mosaics.

The result mosaic image Figure3 is shown below.

4. Conclusions

In conclusion, this project demonstrated the effectiveness of the Harris corner detector in detecting key points in an image and how selecting the appropriate threshold can lead to more accurate and efficient detection.

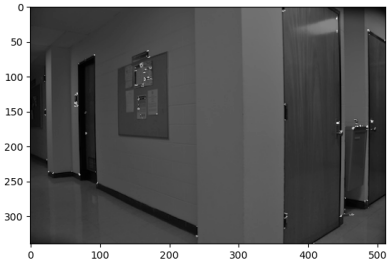
Furthermore, we also showed how cross-correlation and RANSAC algorithm can be used to match key points and compute a homography, which in turn can be used to create a mosaic of input images with reduced artifacting.

Overall, this project highlights the importance of key point detection and matching in computer vision tasks and demonstrates several techniques for achieving this. The techniques presented in this project can be used in a wide range of applications, from creating panoramic images to object recognition and tracking.

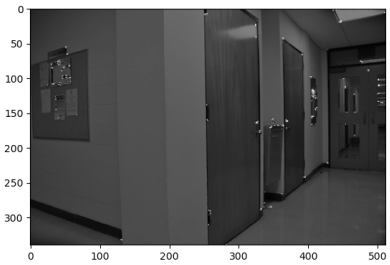
5. Extra Credit

The process of fitting images into frames is a fundamental aspect of the overall workflow described in the report. To achieve this task, the user is required to provide correspondences, which enable the complete definition of the homography. Once this is established, the image can be seamlessly integrated into the frame using the warping techniques discussed earlier.

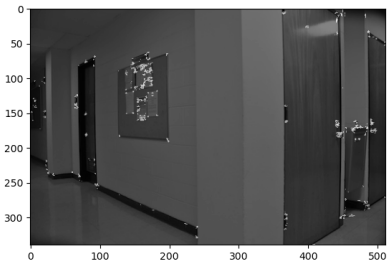
Figure 4 provides a concrete example of this process. The image showcases a professional setting where a serious-looking man is seen peering through a window. In the background, an oil painting pointing to the man, while the file on the desk has been replaced with a Matlab book to enhance the professional atmosphere.



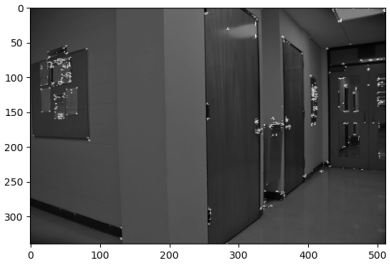
(a) Harris with threshold of 0.01



(b) Harris with threshold of 0.01

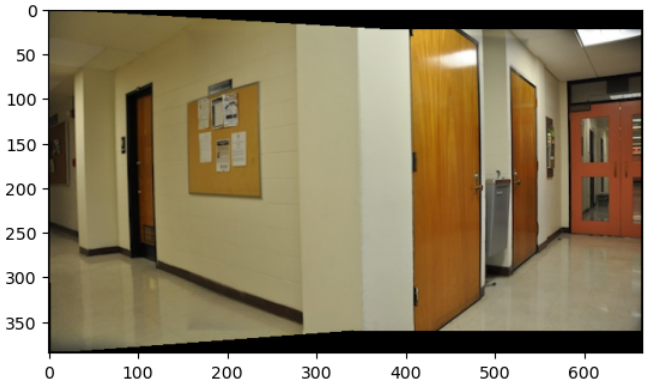


(c) Harris with threshold of 5×10^{-4}



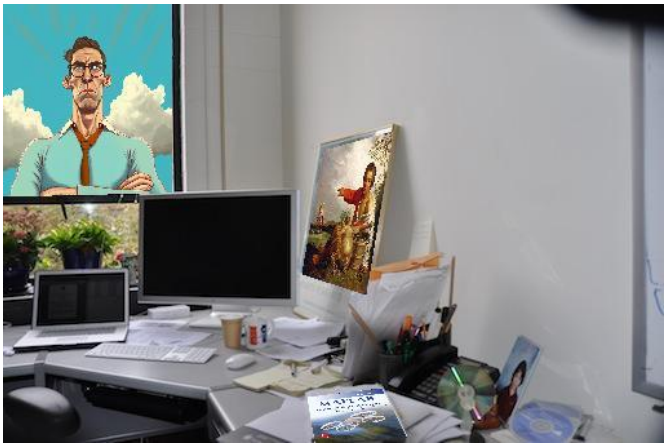
(d) Harris with threshold of 5×10^{-4}

Figure 2



(a) Image Mosacing

Figure 3



(a) replace one surface in an image with a picture

Figure 4

6. Appendix: Code

Please see our git repository for copies of our code. https://github.com/vegetablesB/EECE5639Project