ECE 278A

Date: 2 November 2017
To: Dr. Manjunath
From: Austin McEver and Christian Lee
Subject: Project 1 Report

This report describes each function in the source code of our project one. It also includes output images computed by our project, and lists webpages we used to help with this project. Source code is in the same zip file as this report with file names ImageProcessor.py, ImageWarper.py, and ImageMosaic.py.

Overview

Within this project there is an ImageProcessor class, which handles all the heavy lifting. Using it are two scripts, ImageWarper.py and ImageMosaic.py. ImageWarper.py takes two command line arguments: the path to two images. ImageWarper then warps the first image into the plane of the second image and should make objects in the first image look like they do in the second image. ImageMosaic takes multiple arguments and attempts to combine the images into one larger image such that images share the same perspective plane.

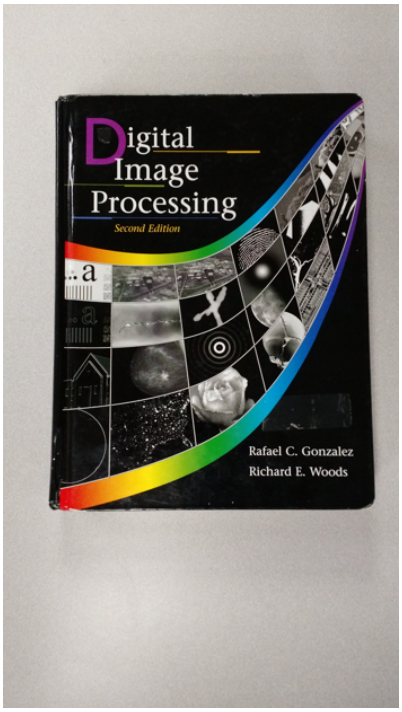
ImageWarper

ImageWarper simply creates an instance of ImageProcessor and calls its Warp method. Warp uses OpenCV's SIFT method to find key points, and then feeds those key points into CV's brute force matcher. It then calls ImageProcessor's ratio test method to filter the matches. Then, depending on the options passed to Warp, ImageProcessor runs DLT, normalized DLT (nDLT), and/or RANSAC. We compute the homography matrix to warp the perspective of the input image into the other image plane. We implemented DLT and nDLT ourselves using an online note resource (https://me363.byu.edu/sites/me363.byu.edu/files/userfiles/5/DLTNotes.pdf).

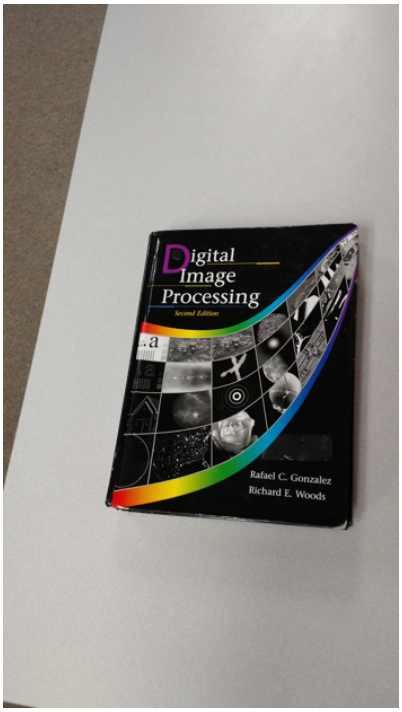
ImageMosaic

ImageMosaic works similarly to ImageWarper in that it creates an instance of ImageProcessor and calls its Mosaic method. The Mosaic method begins similarly to Warp: it runs SIFT, brute force matcher, then RANSAC. Additionally, the Mosaic method creates a zero-padded image out of the second image, in order to accommodate stitching of the other to any orientation. Finally, it uses CV's warpPerspective method to warp one image into the zero-padded image plane. Then, it maps the original second image on to the warped image using Python list replacement. Finally, it shows the result.

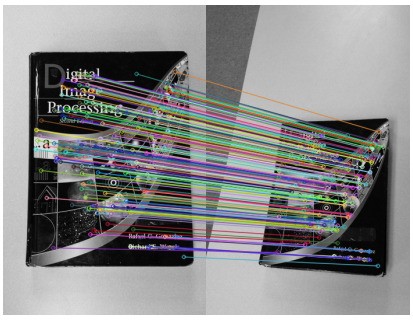
Image Results from ImageWarper.py



Warper input image 1



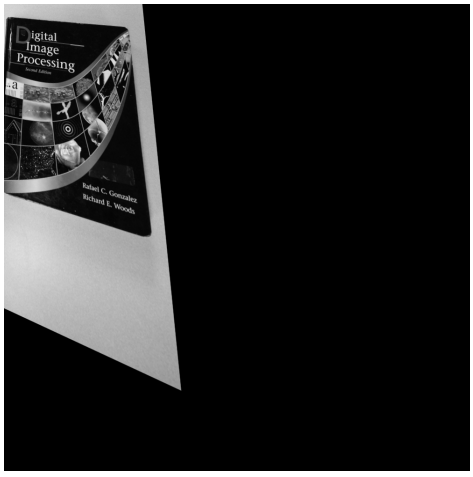
Warper input image 2



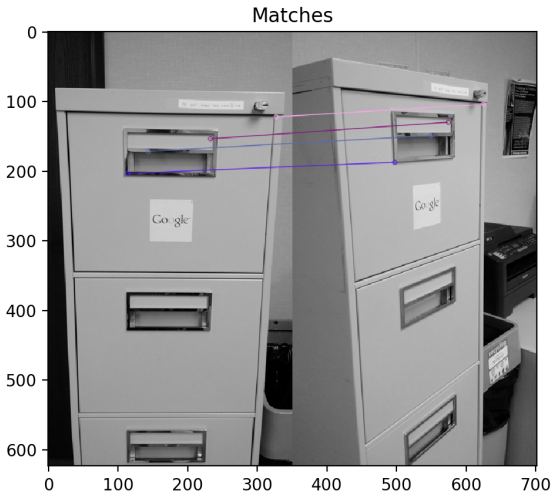
Key point match illustration

Warper RANSAC output

Warper DLT output



Warper normalized DLT output



Key point matches for drawer images

ImageWarper.py Analysis

We tested different combinations of key point searching and matching methods with the images of the Gonzalez book that had the provided homography matrix. According to the CV's tutorial, SIFT and their brute force matcher outperforms the others for the processes that do not require fast computation. Indeed, SIFT gave many more key points than SURF and ORB, and the FLANN key point matcher did not work as well as the brute force matcher did. SIFT key point searching and brute force matching was followed by the ratio test that D. Lowe used to filter outliers. The value of 0.5 yielded the best result when we compared our final homography matrix to the provided one.

We did not notice dramatic improvement with RANSAC in computing homography compared to our DLT. This means that our key point matching had very low number of outliers. However, considering that the Drawer images did not have enough distinctive key points for useful key point matching, it was clear that key point matching normally results in considerable number of outliers and that RANSAC needs to be used to compute a good homography matrix.

We need to improve our Normalized DLT as it should improve our results. We could not get a correct homography matrix for the drawer images, which gave us a lesson that it is hard to use SIFT for objects with repetitive features such similar corners because it operates based on key point descriptors that can only describe what they see within their little circular space, not the bigger picture.

Image Results from ImageMosaic.py

Kohn Input Images

kohn_2.png

kohn_3.png

kohn_4.png

kohn_5.png

kohn_7.png

Kohn Output Images

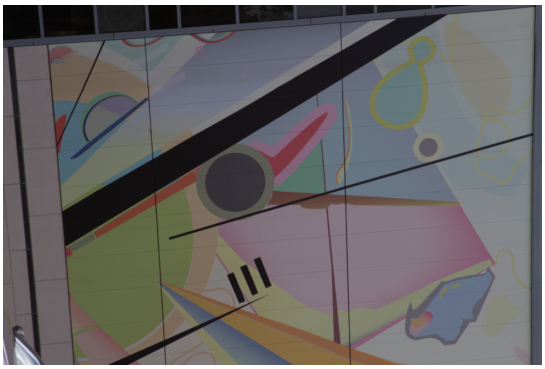
1.jpg: kohn_2.png stitched with kohn3.png

2.jpg: 1.jpg stitched with kohn_4.png

3.jpg: 2.jpg stitched with kohn_5.png

4.jpg: 3.jpg stitched with kohn_7.png

Graffiti Input Images



graffiti_3.tif

graffiti_6.tif

graffiti_7.tif

graffiti_10.tif

graffiti_14.tif

Graffiti Output Images

5.jpg: grapffiti_3.tif stitched with graffiti_6.tif

6.jpg: 5.jpg stitched with graffiti_7.tif

ImageMosaic.py Analysis

ImageMosaic.py accepts a variable number of arguments and stitches two images together at a time saving the result to process it with the next image. Instead of changing the perspective of the image we intend to add, we compute the homography from stitched image to the image we intend to add. In order to accommodate stitching to any orientation, we zero-padded the image being added. It worked well until the last stitch trial with an image that is off in scale (kohn_7.png) compared to the stitched image (3.jpg). Also, ambiguous features of the sky from sunlight might have contributed to the last trial's bug. Similarly, with the graffiti images, we were unable to stitch images that included sky. We could improve Image Mosaic by making the amount of zero-padding increase as we add more images.

Conclusion

Mosaicking multiple images from different perspectives, our final and the most important objective, worked well other than with images including sky. We realized that images with dramatic difference in scale or lighting are hard to stitch together. We also found that SIFT cannot perform well with the images or objects with indistinctive features. In future projects, we could test deep learning's ability to resolve these issues or try other methods altogether.

Sources

- https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/
- https://docs.opencv.org/3.3.0/db/d27/tutorial_py_table_of_contents_feature2d.html
- Fischler (RANSAC) - https://gauchospace.ucsb.edu/courses/pluginfile.php/1416851/mod_label/intro/PaperpFischler1981.pdf
- Lowe (SIFT) - https://gauchospace.ucsb.edu/courses/pluginfile.php/1416851/mod_label/intro/Paper_SIFT_IJCV2004.pdf
- https://www.youtube.com/watch?v=oT9c_LlFBqs&t=2173s
- https://me363.byu.edu/sites/me363.byu.edu/files/userfiles/5/DLTNotes.pdf