Fully automatic panorama image stitching.

Manuel Felipe Pineda Loaiza

***Abstract*—This work covers the problem of fully automated panorama stitching, this is a common computer vision task which combines multiple images with overlapping sections to produce a larger image. I applied the theory about 2D projective transformations, features detection, local invariant descriptors and matching algorithms to produce a small library for image stitching.**

***Keywords*—Computer vision, 2d projective transformations, homography, perspective, RANSAC, descriptors, matching.**

I. Image stitching stages.

The image stitching is a process to combine several images into a larger one, It can be used in several applications like the creation of high resolution images or the image alignment process used in the video stabilization. This is an interesting problem with several stages in which we apply different techniques of computer vision. In this section, I will present an introduction to those stages.

A. Feature detection

This aims to detect "interesting" parts of an image, those points are generally detected when there are big changes of illumination or colors, as humans we can see them in the corners and edges of the objects. The most common techniques for edge detector are the sobel filter, and Canny edge detector, the Harry's corner detection and more. The basis for this methods are the derivatives of the image and the discontinuities.

Fig. 1: Example of edge detection using Canny edge detector. Taken from wikipedia.com

B. Visual descriptors

The visual descriptors are an extension of the features detected in the previous subsection. They pretend to collect "information" about the image that is used as an input for other stages. In this case, this information is used to generate a match of key points between a pair of images. In this work I used an Scale-invariant feature transform [1] (SIFT) as visual descriptor.

Fig. 2: Plot of feature vectors found using SIFT.

C. Matching

Once we found the visual descriptors in two images, the next step is to find a match between the descriptors in the first image with the descriptors in the second one. For this task I used the FLANN-based matcher provided by OpenCV and suggested in [2].

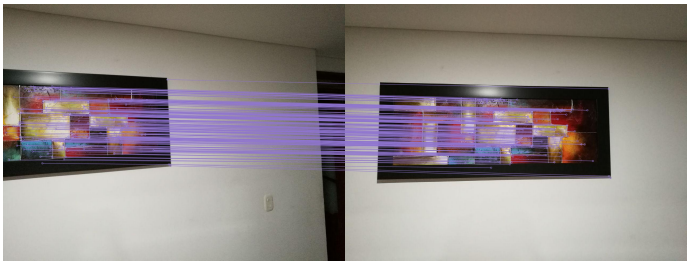


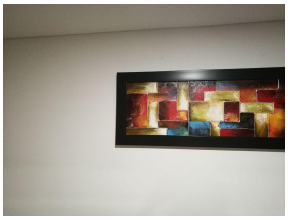
Fig. 3: Plot of the matching points between two images.

D. Homography estimation

After find the matches, we need to compute a projective transformation in order to put the two images in the same plane. It can be shown that we can estimate the projective transformation h using 4 or more matching correspondences between the two images x and x' using the following system of equations for each pair of correspondences:

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'x \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

We face a problem here because the previous stage could generate several outliers for our system of equations. To solve this, OpenCV provides the implementation of RANSAC [3]



(a) left side image of the panorama stitching

(b) right side of the panorama stitching

Fig. 4

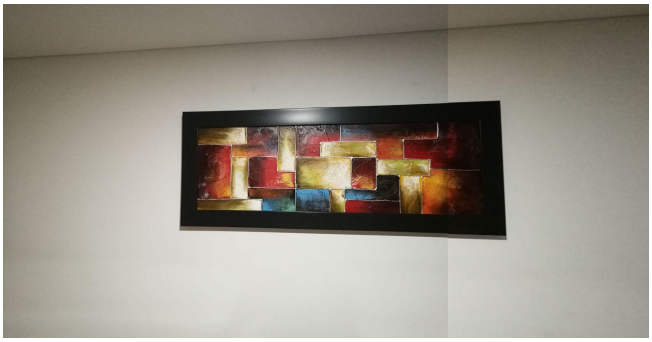


Fig. 5: Result of panoramic stitching of two images

to automatically remove the outliers and compute the best possible solution.

After we found the projective transformation, we just need to map one image to the same plane of the other one. This is the last step for the panorama stitching we can see the result of this in the figures 4 and 5.

E. Crop Contours

When we stitch two images, the result image width is not exactly the sum of the width of the two original images. As the images have an overlapping section, the result image will be smaller and will have an empty section in one of the sides (right side in this case). In order to handle this problem, I added an automatic crop of the contours using the functions `cv2.findContours` and `cv2.boundingRect`.

F. Generalization to multiple images

As the projective transformation is a linear operator, is quite simple to extend the algorithm to several images. To achieve this, we just need to keep track of all the intermediate projective transformations and carry them in a single matrix, with this, we can project all the images to the same plane. The figure 6 is an example of automatic stitching of 3 consecutive images.

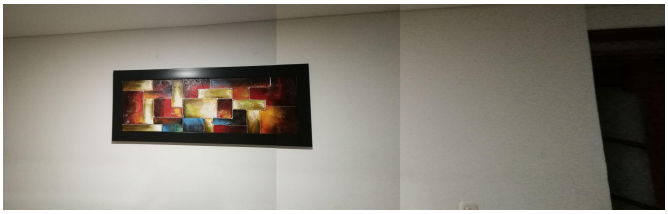


Fig. 6: Panorama stitching of three consecutive images.

II. Experiments

The images shown in this work (with the exception of the Canny edge Detector example) were generated using OpenCV and Python 3.6.

III. Conclusions

The fully automatic panorama stitching worked as expected using OpenCV utilities and was structured as a Python module for future use.

The result images show a big difference in the color and illumination, this can be used as future work to automatically balance the color and the illumination making the image more "smooth".

References

[1] Wikipedia. Scale-invariant feature transform — wikipedia, the free encyclopedia, 2017. [Online; accessed 6-October-2017].

[2] OpenCV. Feature matching + homography to find objects, 2017. [Online; accessed 6-October-2017].

[3] Wikipedia. Random sample consensus — wikipedia, the free encyclopedia, 2017. [Online; accessed 6-October-2017].