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 overlaping 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. Introduction

The image stitching is a process to combine several images into a larger one, It can be used in several aplications like the creation of high resolution images or the image aligment 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, currenty there are serveral techniques to achieve that based in the detection of edges, corner and ”blobs”.

Sobel, Canny, Harris, Laplacian of Gaussians, etc.

B. Descriptors

Sift

C. Matching

Approximate Nearest Neighbors

D. Homography estimation

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

In matrix notation: $x' = Hx$

It can be shown that we can estimate the matrix h using 4 matching correspondences between x and x' where each correspondence induces a pair of equations to the linear system (assuming $h_{33} = 1$):

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

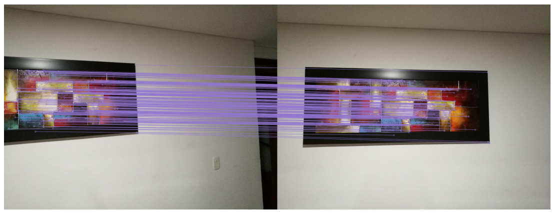
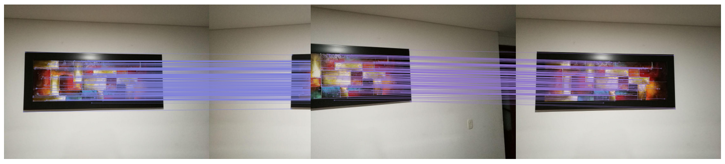


Fig. 1: Embedded image into another scene



(a) Bilinear (b) Nearest integer pixel

Fig. 2: Pixel estimation

Note that this points correspond to the original matrices (images) in euclidean coordinates, not homogeneous coordinates.

This was programmed as a function that can receive 4 or more points and solves the linear system using numpy.linalg.lstsq [1] (least-squares solution) to gain robustness.

E. Crop Contours

F. Generalization to multiple images

II. Experiments

III. Conclusions

References

[1] numpy. Least-squares solution to a linear matrix equation, 2017. [Online; accessed 17-September-2017].