Shalom Weinberg
Tal Siman Tov
Neil Michaeli

Panorama
final report

Introduction

Sometimes when we are taking a photo we want the photo to describe the exact moment we are experiencing, but the problem with camera is that their field of view is not as vast as the human eyes and the consequence is narrow photo that does not deliver the feel we wanted .

So, how will we overcome narrow field of view?

by taking x pictures one after the other, in order and obviously of the same scenery, we will be able to connect them to our "big picture" and overcome the situation and overcome the problem with vast, beautiful picture that can describe better what we see.

Thats great! but there's a few Difficulties.

the camera can easily move from one sub-picture to another and not moving in uniform speed. we will want to connect the sub-pictures to one and we will overcome these difficulties by finding connection or "descriptors" between pictures, calculate the error and direction they move in order to construct one big picture from them

The Algorithm

1) For every two pictures we will find matching features by using Orb algorithms .
2) Calculate the Homograph with 2 points from the different pictures with Ransac.

I. Sample 4 matching points.
II. Calculator the Homography matrix.
III. Calculating the inliers s.t $\| Pi' - Hpi \| < e$.
IV. Taking the biggest inliers group.

V. Calculating inliers by the biggest homography.

3) Display the matching features on 2 picture one side of the other.

4) Warping the pictures to a grid that he's side determined by the minimum and maximum x's and y's

5) At the end, we will return the "full picture"

Different approaches and Methods

1) When we tried to draw the matches we found, sometimes we needed to resize the photo to smaller dimensions .
2) We worked with greyscale to lower complexity in the code
3) Orb gave us the best results and worked in all platforms (compared to sift & surf)
4) Orb consumes the same key points with original and the rotated picture

So how does Orb works?

Orb is using Fast to find key-points and then using Harris Corner Measure in order to find n leading (Highest rated) points, he also uses a pyramid to create multi scale feature.

Any restrictions?

Well, he is great, but does not calculating direction.

https://medium.com/@shehan.a.perera/a-comparison-of-sift-surf-and-orb-333d64bcaaea

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/

1) After we used orb, we looked ay several methods to filter it, we used opencv's bf, ssd and eventually used normal difference.
2)After that we used ransac algorithm there, we used a conversion of point to homographic point and from there the homographic application.
3) Then we used function that draws the matches and updated the homographic matrix by it neighbour (for more than 2 pictures)
4) sent the coloured pictures with the Homographics and created the final picture

Results

Tests we did:

1) we tried to use both sift and orb but eventually we chose orb because it works more easily with nearly any errors, it fast and ob course free.

2) as Mentioned sometimes we photo resize needed.

Ransac Parameters 1000 iterations, 1 threshold

we tried way more option but 1 as a threshold produced the best results.

Basically, we want the threshold to be as low as possible and the number of iteration to be as high as possible because these parameters will produce the best results. we've seen that if we will use 1000 iteration and even half of the data not relevant we will get 99% success.

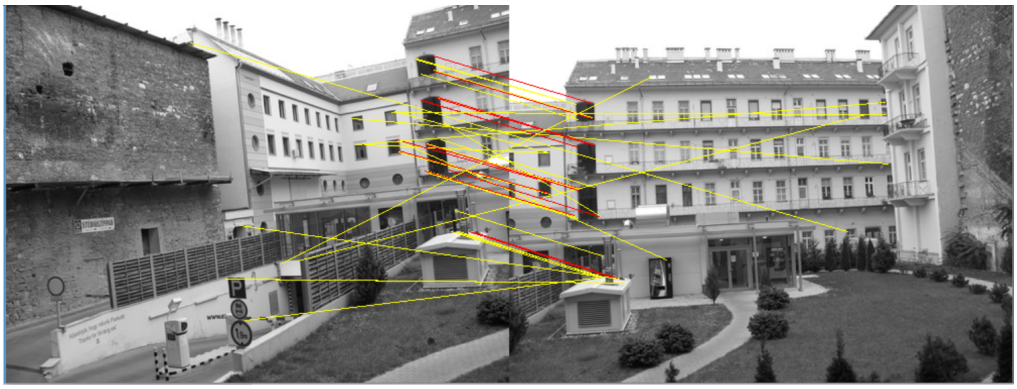
What did you use in order to measure success and why?

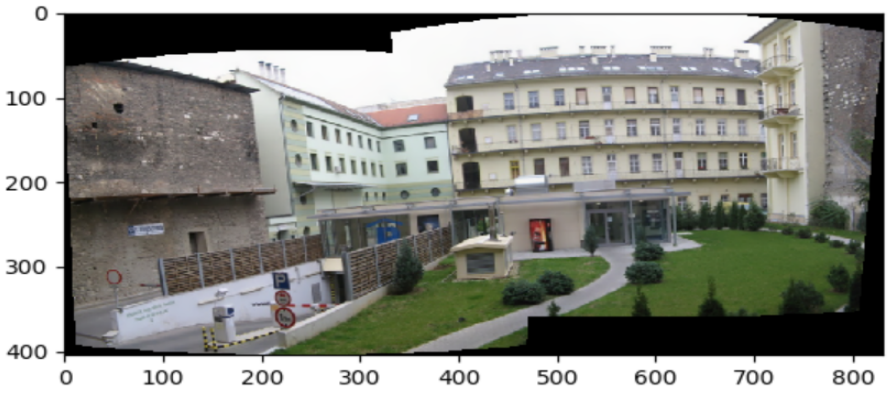
with the number of iteration and threshold as mentioned above

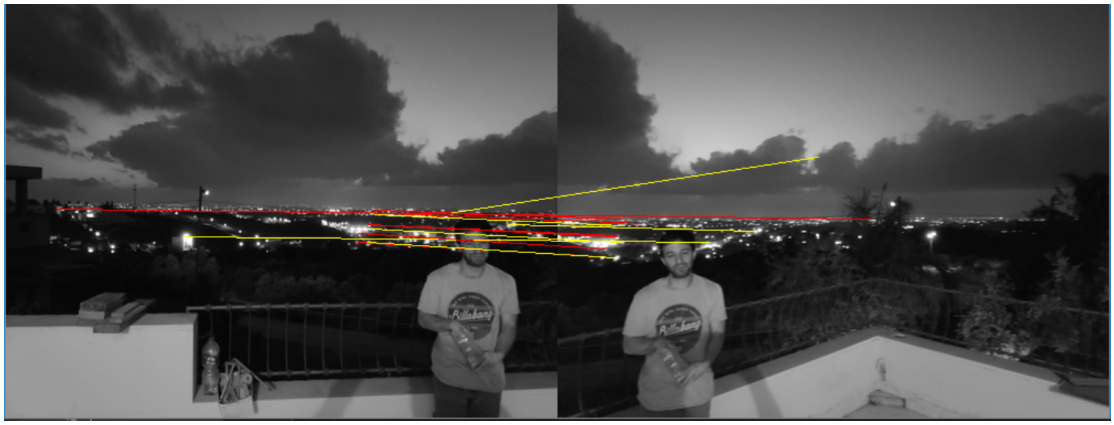
with these parameter we can measure the model success (the homography).

we chose the final picture by literally seeing if it looks like it created from a few sub photos.

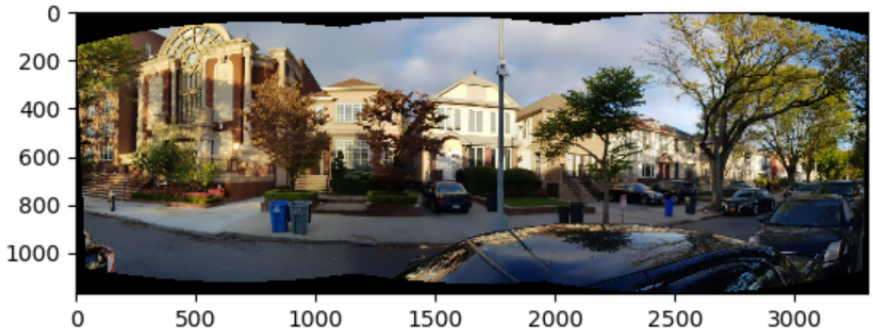
Examples

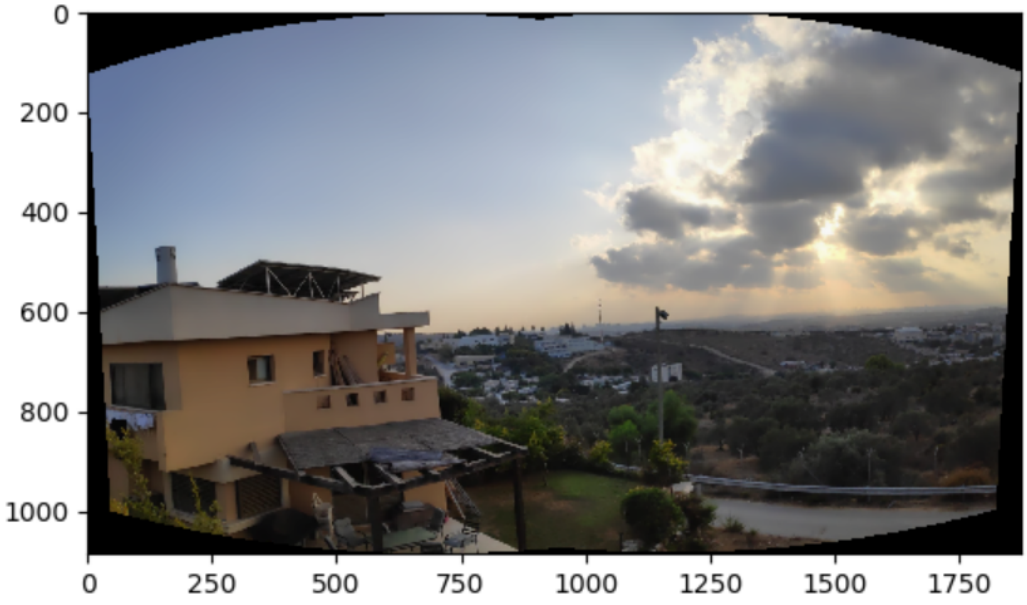



0
100
200
300
400
0
100
200
300
400
500
600
700
800




0
200
400
600
800
1000
0
500
1000
1500
2000


0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000


0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750