Sharif University of Technology

Computer Vision

HW02:
Make panorama and Image Stitching

Author:
Saeed Razavi (98106542)

May 10, 2022

section1(Stitch two key frames)

1.in first section , we stitch two frames (270 and 450) . To stitch two overlapping video frames together,first we need to map one image plane to the other. To do that, we need to identify key points in both image. we use "sift.detectAndCompute" function from OpenCv library that find the corresponding points between two frames and then use $cv2.findHomography$. in this method we use Ransac with 7000 iterations and confidence of 99.5 . because of space limitation , first i downsize the frames by factor of 4 and then using sifts and find homography .we check that your homography is correct by plotting four points that form a rectangle in frame 450 and it's projections in frame 270:

(a) frame450

(b) frame270

Figure 1

now , we combine these two frames without any blending and for common pixels , we get the value from frame 450 . below you can see the result :

Figure 2: two frames combined without blending methods

section2(Panorama using five key frames)

in this section we will produce a panorama using five key frames. we have some challenges :

1) mapping from some frames to frame 450 is difficult because they share very little area. Therefore we need to perform a two step mapping , using frame (270,630) as a guide frames and then multiply two homographies to reach the final homography

2) for stitching part, we use both dp optimization(minimum cut) and after finding best boarder , use Laplacian pyramid blending two stitch these frames with the best result .
we show the procedure for stitching two frames : first if we just copy one frame into other frame we have :

Figure 3: combining without dp and blending

now if we apply dp optimization(minimum error cut) we have :

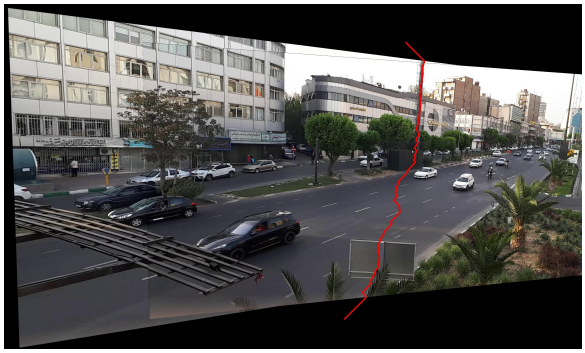


Figure 4: find minimum cut between two frames

after detecting the minimum cut , if we combine two frames without any blending , differences between both sides of boarder is very clear and obvious . so we use Laplacian pyramid with 9 levels to combine frames very smoothly and naturally . below you can see the differences :

(a) before Laplacian pyramid

(b) after Laplacian pyramid

Figure 5

we do the above procedure for each two frames that are next to each other with following condition :

- we apply dp optimization for three boarders between two frame (vertical boarder , horizontal boarder that is on the top of page and horizontal boarder that is in the bottom of the page and make the mask based one the common areas that these three boarder make) below you can see the final result :

you can see the final panorama image below :

Figure 6: panorama after applying minimum cut and Laplacian pyramid

section3(Map the video to the reference plane)

in the section we make a video by projecting all frames onto the plane corresponding to the reference frame 450. For those frames that have small or no overlap with the reference frame we use two step mapping that we discussed before (using key frames). you can see one frame of these video below :

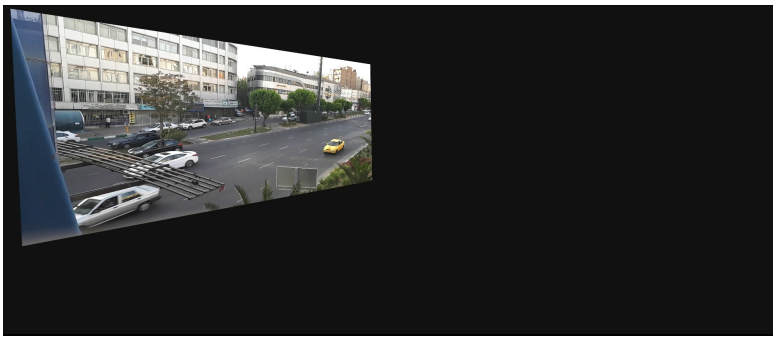


Figure 7: one frame of video

to see video of this section go to the google drive that i shared with you :)

section4(Create background panorama)

in this section we remove moving objects from the video and create a background panorama .to do this , i try two different methods with different characteristics .

- first method : i divided the whole panorama image vertically into 230 small windows . for each small Windows, i iterate all over the frames and if the projection of frame into frame 450 includes that specific small windows, i save that part into the list . after gathering all of the proper frames that includes the small window , we get median from this list and set the small window to that median . the advantages of this method is that it's very fast (took around 3 minutes to extract background)
- second method : in this method that is very time-consuming in comparison with previous method , we iterate all over the pixels and for each pixel we iterate over all frames and see if the projection of that frames into frame 450 contains that pixel . and at the end get the median from all of these frame that contains that pixel .

  you can see the result of two method . i prefer the first method due to it's run time and use the result of method 1 in next sections .

(a) method(1)run time: around 3 minutes

(b) method(2)-run time: around 1hour and half

Figure 8

section5(Create background movie)

in this section we should map the background panorama to the movie coordinates. For each frame of the movie, we need to estimate a projection from the panorama to each frame using homographies that we calculated in section 3 . you can see one frame of these video below :

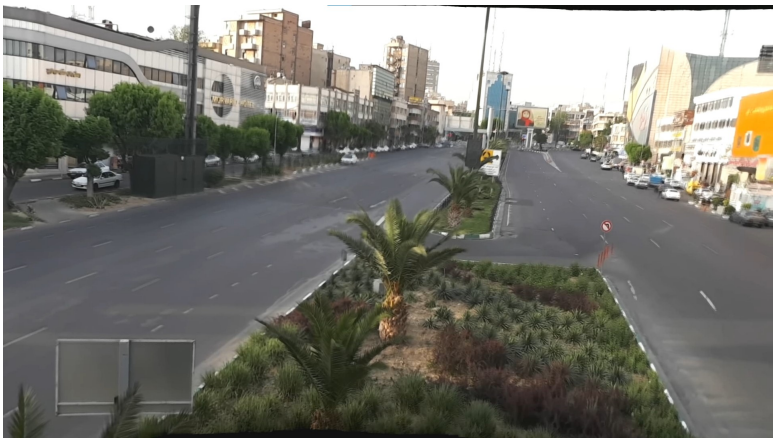
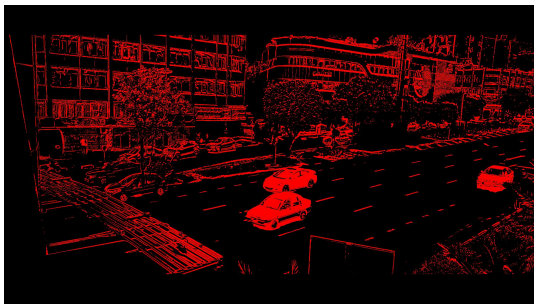


Figure 9: one frame of video

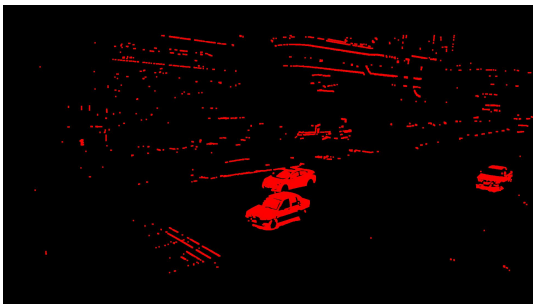
to see video of this section go to the google drive that i shared with you :)

section6(Create for-ground movie)

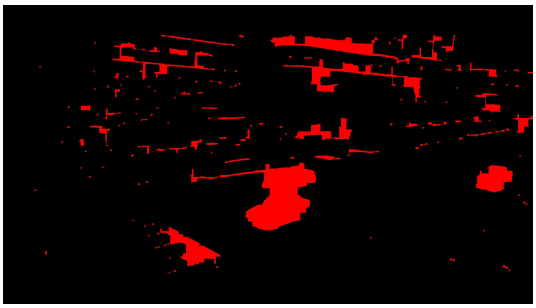
in this section we want to make a video from moving objects .in each frame we calculate the absolute difference (L_1 norm) between original frame and background . and set the threshold for these differences (in my code it is 140) if the differences in a pixel is bigger than this threshold , it can be pixel of moving object . to remove background noise we use two morphological transformation . first we use "opening"(erosion followed by dilation) to remove noises on the mask . we use the kernel with size 5 for erosion and remove noises and then we use "closing"(dilation followed by erosion) to make the mask over moving objects bigger (make it more recognizable)
below you can see the result of these procedure :



(a) mask after threshold



(b) opening morphology



(c) closing morphology

Figure 10

Figure 11: moving object detection

to see video of this section go to the google drive that i shared with you :)

section7(Generate a wide video)

to generate wider background video , we just have to set the width of the window bigger in the perspective transformation . an interesting point is that if we want that wider video doesn't include black pixels we have to set a rule :
1)first frame (frame1) has to show all of the pixels that are equal or greater than it's pixels and the last frame(frame900) show all of the pixels that are equal or smaller than it's pixels and for middle frames, linear relationship holds . i show three wider frames (frame1, frame 450, frame900) below:

Figure 12: frame1

Figure 13: frame450

Figure 14: frame900

section8(Remove camera shake)

if we assume that camera parameters change smoothly and obtain a temporally smoothed estimate for each camera parameter we reach shake less video to do so, for each element of homography matrix we use polynomial approximation . to say it more accurately for example for element[1,1] , we have 900 data from 900 homography matrix and we have to fit the function that best approximate the model . in this section i use "savgol-filter" from scipy.signal library and fit the polynomial with 7 degree and kernel size of 351 for these 900data . for better intuition about smoothing noisy data see the picture below :

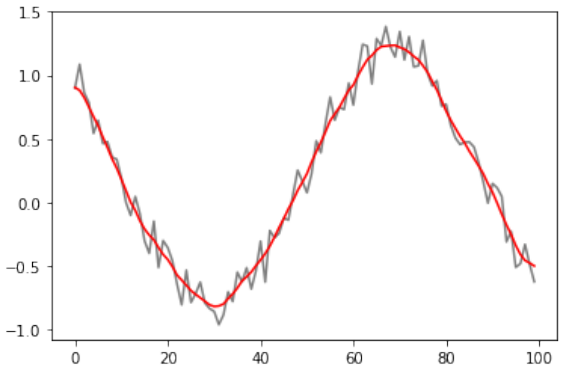


Figure 15: smoothing data

to see video of this section go to the google drive that i shared with you :)