CS 763/764: Computer Vision

Answers

Please find the outputs of this task here.

Question 1. The folder Q1 contains unstable videos of static scenes, i.e., the motions in the videos are introduced only by a shaky camera. Your job is to write code which will take these videos as input and use the three-pronged framework mentioned above. Follow the steps below:

1) Motion Estimation with OpenCV
   (a) Features : Use (detector, descriptor) correspondences between adjacent frames to estimate compact motion parameters. Needless to say, this process should not involve any manual input.

   OBSERVATION

   We detected features in adjacent frames using the FAST and BRIEF detector and descriptor pair (part of ORB implementation in opencv) because we wanted fast computation. We used brute force matcher of opencv to obtain the best matching keypoints in the two frames. The parameters for the FAST demanded (nfeatures = 800) and the rest were set to defaullt, out of which the best 50 matches were picked. Using the keypoints, we found the rigid transform to estimate the motion parameters between two frames.

   $$X' = R \cdot X + T$$

   Here X' is the rigid transformation of X points, R and T indicates rotation and translation respectively. R matrix consists of one degree of freedom (θ), whereas the translation matrix T has two degrees of freedom (T_x and T_y). This results in similarity transform and this is a better solution compared to homography, as rigid transform does not scale the image. We did this for all the frames with respect to the first reference frame and estimated the motion parameters of all the keypoints in all the frames. Then, using a Gaussian filter with mean 100 and standard deviation as 1000, we smoothed the trajectory of the keypoints. Finally, using the smoothed trajectories, we obtain the affine transform using the rigid transform matrix with the smoothed parameters. Thus, doing this for every frame, we obtain the stabilized video.

   (b) Optical Flow : Use (dense) OpenCV optical flow **calcOpticalFlowFarneback()** in order to obtain compact motion parameters

   OBSERVATION

   Using **calcOpticalFlowFarneback()** function in OpenCV, we found out the optical flow between two consecutive frames in the video. In order to compute the rigid transform, we need two set of images which are translated between them. Using the first image, and the computed flow values, we translate the pixels according to the displacement in their respective directions and compute the second image. Then using the two images, we compute the rigid transform to obtain the motion parameters between the two frames.

   $$I_2[(x, y) + (u, v)] = M \cdot I_1[(x, y)]$$

   Here M is the rigid transform matrix we discussed in first part of the question. We repeat this step for all the subsequent frame keeping the first frame as reference, and obtain the parameters for all the frames. We then smooth these parameters which represent the trajectories of the points in the image, and use the matrix of smoothed trajectories to obtain the rigid transform. We repeat this for every frame in the image and combine them all to obtain the stabilized video.

2) After estimating the motion parameters, use a Gaussian filter to temporally smooth the parameters.

OBSERVATION Here as we increased the size of Gaussian, the translation and rotation parameters trajectory became smoother, and thus leading to non abrupt frame rotations and translations.

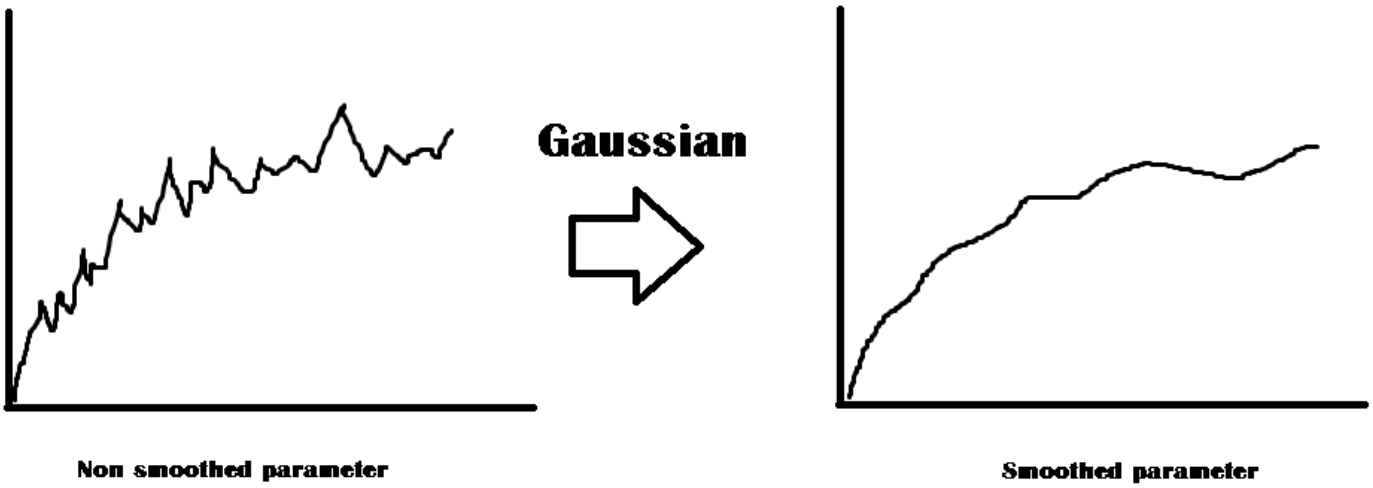

FIGURE 1. Smoothing

3) Correct all frames using the first frame as the reference.

Write down your observations. Which method is 'superior'? How did you established this? Report all quantifiable parameters (and be careful in reporting correct time). Use time per frame to quantify your answer.

OBSERVATION

From our experimentation, we found out that both methods gave similar quality of stabilization when used over the video in folders Q1. Time taken for processing a frame by different procedures is given in the table 1 and from this result we observed that the computation time of the feature based method is far less than that of the optical flow based method. Thus, the feature based method is more superior that the flow based method.

Question 2. In the previous question, the input did not contain any independently moving objects. In such scenarios, one can assume that the motion estimated in the scene is pertaining only due to the camera motion. The folder Q2 contains another set of unstable videos. Unlike the previous set of videos, these videos contain independently moving objects. Ideally, to restrict motion parameters to be dependent only on camera motion, one needs to eliminate the motion vectors corresponding to the independently moving objects.

In this context, estimate the motion parameters with an additional filtering mechanism as mentioned below.

a) Apply the program of Q1 on Q2. Provide your observations (in the context of the subsequent steps).

OBSERVATION

As video stabilization is being applied on the entire video frame, the motion of the moving object, i.e, the girl in this case, is affecting the positioning of the frame in the output video. The output seems stabilized but there is a lot of shakey-ness in the displayed frames. Also, when the object moves very quickly one direction, the frame moves along with it, leaving a black patch at the displaced location. This can be seen from the Figure 2 and Figure **??**, when the moving girl moves towards left the stabilized frame also moves towards left and similarly on the right.

b) Motion Estimation

FIGURE 2. Stabilized Frame moves left and right with the moving object

i Feature Points Descriptor : As before, use the (same) (detector, descriptor) idea. However, before estimating motion parameters, devise an automated (i.e. non-manual) mechanism to ensure that the subset of feature point descriptors you choose to estimate motion parameters belong only to the static objects in the scene. For us to evaluate your method, write a video with feature points marked out. The ones you used for motion estimation should be marked in green and the ones weeded out should be marked in red.

OBSERVATION

The video with the feature points marked out is present in the 2_A output folder. Here, we observe that feature points corresponding to the background are represented with green colored dots whereas those corresponding to the moving object are represented in red.

The keypoints are detected and matched in current frames keeping the previous frame as the reference. We observe that after applying **RanSac** on the detected keypoints, The inliers correspond to the features which are moving relatively less between the frames or are detected more accurately in the image. Thus, conversely the outliers correspond to the features which are detected on the moving object, i.e the girl, and have high motion. Thus, we are able to separately identify the keypoints of the static and moving objects in the scene. Note that, the extra red point that the extra outliers using RanSac will not affect the quality of the stabilization as primary objective of this filtering is to remove the feature points corresponding to independently moving objects. A sample frame with marked out feature points is presented in Figure 3

ii Optical Flow: Similarly, while estimating motion parameters through this method, devise a scheme to eliminate moving objects. In this case, you need to write a video of binary-masks for each frame, such that the frame has white color for the regions used for motion estimation and black for the regions weeded out. For this part, use two variants. First, use the stock OpenCV

FIGURE 3. Image showing the key points corresponding to the static and moving objects

version **calcOpticalFlowFarneback()**. Next, use the provided optical flow from PWCNet (takes 30ms per frame) and solve the problem (see more below).

OBSERVATION

Steps to find the mask for eliminating the moving objects between two frames:

(i) Find the dense optical flow using **calcOpticalFlowFarneback()**. (or the given flow from PWC was used)

(ii) Find the pixel location of the current frame using the calculated optical flow and pixel locations of previous frame

(iii) Use the two pixel locations to find the homography between two images and RanSac to filter out the outlier. Here, our assumption was that the moving object will be identified as outlier because it is moving independently and will not come under perspective transform between two images. And our assumption was correct.

(iv) Finally the inliers were given white colour and outliers were discarded.

An example frame is shown in the Figure 4 and Figure 5 by calculating the flow usin calcOpticalFlowFarneback() and PWC respectively. This can easily be observed here that the mask

FIGURE 4. Comparison of the original frame with the segmented mask obtained from flow values using calcOpticalFlowFarneback()

calculated using PWC method has less noisy components. It has identified the moving object very clearly. Again it shouldl be noted here that extra black pixel found using calcOpticalFlowFarneback() will not hurt the stabilizing of video for the same reason as explained in previous section. The videos of binary mask for each frame are saved in the folder 2_B and 2_C inside Output folder respectively.

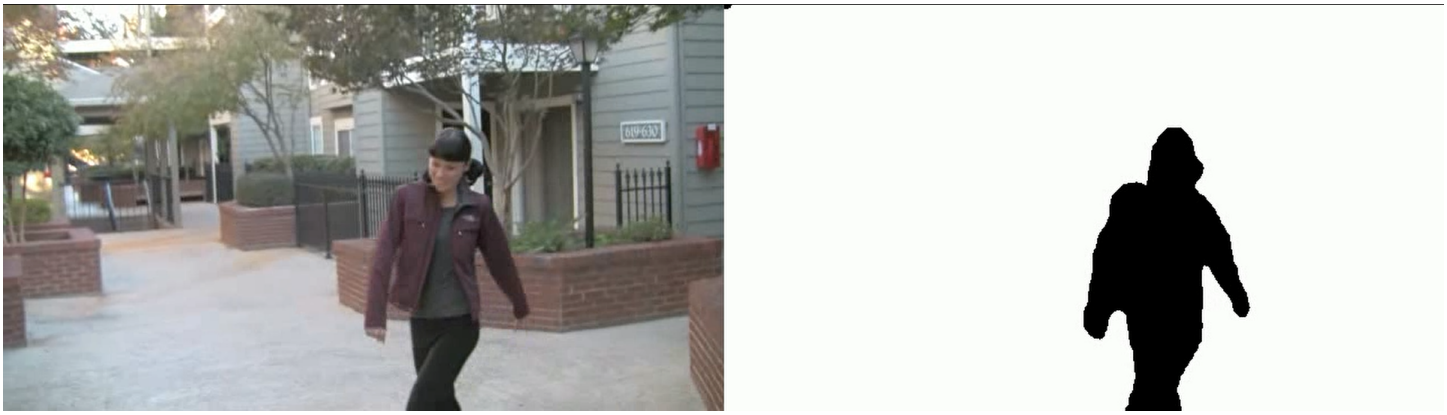

FIGURE 5. Comparison of the original frame with the segmented mask obtained from flow values using PWCNet

c) As before, smooth, and write the output frames to form the video.

TABLE 1. Time taken per frame for different procedures

Question No.	Video Stabilizing Procedure	Time (in seconds)
1A	Using sparse feature points	0.03 seconds
1B	Using (dense) optical flow	0.07 seconds
2A	Sparse feature points and additional filtering	0.03 seconds
2B	Optical flow (calcOpticalFlowFarneback()) and additional filtering	0.44 seconds
2C	Optical flow (Deep Learning derived) and additional filtering	0.39 seconds

OBSERVATION

Stabilization quality of all the methods were almost similar. However, we observed that method 1A and 1B failed when there is an independently moving object is present in the video. From the table above we also observe that Methods using sparse feature detection were faster than the methods using dense optical flow. 2B and 2C are taking a lot of time per frame because it is also trying to find the mask for filtering the independently moving objects. PWCNet method is still faster than that calcOpticalFlowFarneback() method by few milliseconds per frame.

Question 3. You need not give a code for this part. However, you should detail your proposal with (ideally) necessary mathematics.

a) Apply your code from Q1 to the videos in the folder Q3. You would notice that the stability comes at a "cost". Explain what is going on here.

OBSERVATION

We have observed that when we stabilize a video, there is displacement of the frame along with it to reduce the relative motion between the moving objects in the frame, this is what essentially gives a sense of stability in motion videos.

In the case of video in folder **Q3**, there is a large displacement of the camera when it rotates towards the back. To stabilize such actions, the image displaces itself through and equal amount in the same direction. This leads to the image going out of the frame leaving a black patch behind. This defies our purpose of stabilization, because information is lost in some frames giving us poor results. Results for one of such frame is shown in the Figure 6

b) In the previous questions, you used a simple Gaussian filter to smooth the motion. Can this simple smoothing be introducing the problem? Propose a better temporal smoothing technique.

OBSERVATION

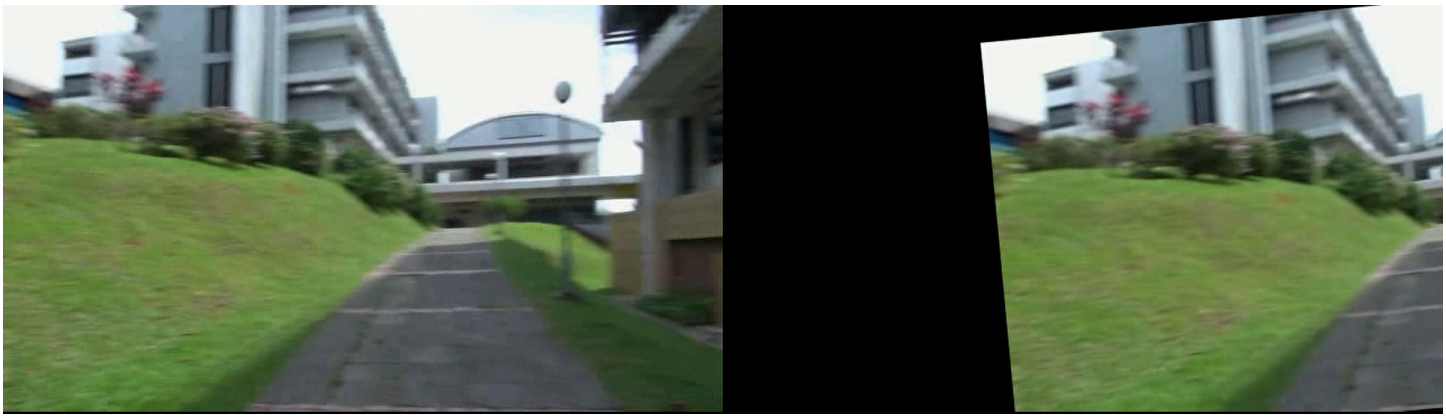

FIGURE 6. A frame showing the displacement of frame leading to several black pixels

Gaussian smoothing which uniformly applied over the image is the reason why large motion of the camera tends to throw the image out of the frame. Yes it is the problem.

If we use median filtering instead, such sharp peaks which correspond to large motion of the key-points can be removed, thus giving us a smoother flow.

c) List other problems with amateur videos including this one. Provide ideas for solutions.

Other problems in amateur videos include :

- Defocusing : Sometimes in low lighting conditions, the autofocus takes time, this leads to defocusing in images.
  Solution
  Defocusing essentially gives a blurry video, that can be enhanced using sharpening methods.
- Low light videography : Due to limited exposure of the cameras, night videography quality is lower than day time videography.
  Solution
  To capture videos at night camera needs more lights to capture a good video so the physical stability of the camera can provide a solution in this case.

Question 4. Data set question.

a) Provide 2 short videos to support your work with your choice of method. Each of the videos must include at least 2 of the 3 group members, and in (exactly) one of them, there should be independent swift moving actions which showcases your talent (e.g., a dance move or a basket ball throw). Note: For those not on campus, and only for those not on campus, try to be create but we can cut some slack on your choice of video.

OBSERVATION

We could not fulfill the criteria of "Each of the videos must include at least 2 of the 3 group members" because only one of our team member is on the campus. We have provided two video. In exactly one of videos, one of our teammate is performing an independent movement. To stabilize this video, we used the filtering based approach and we are getting a pretty good results for this and the points are correctly classified as background or independently moving object. For the second video, we used an old video that was shot while riding a bike, so there are a lot of shaking in the video that were stabilized by the first method and the output is acceptable. However, there might be cases when our approach may not work such as a very high movement in the camera or the person, or very low movement of moving object with respect to background and many such scenarios.

b) Challenge. You are also encouraged to provide an end-to-end solution which uses deep learning for video stabilization (e.g. include PWCNet in the pipeline).

OBSERVATION

We implemented a deep learning based method for video stabilization using two different approaches. Our first approach was using an inverted Y-Net autoencoder architecture to which, we send two consecutive frames from the unstable video sequence as input, and the corresponding stabilized frame as the ground truth. Our network had a total of $2,916,419$ parameters and was trained for 10000 epochs. We used mean squared error as the loss function and Adam as the optimizer with learning rate of 0.01. Our results were not satisfactory with the above approach, our loss stagnated at a value of 7787, and our reconstructions were not good. We believe a powerful network and a more efficient representation of the input and output image is required to enable the network to estimate the correction parameters better, thus giving us better results.

If provided more time and resources, we might approach the problem in following way:

(a) First we need a large and reliable dataset with lots of videos. We don't essentially require unstable and stable video pairs as we can use the stable videos to generate unstable videos.

(b) After we have a reliable dataset, we need a model and a loss function.For loss function we may try different already available loss functions such as Pixel wise mean squared error and others. Now for architecture, we need something that can find optical flow, account for the new pixels that were not present in previous frame, could detect the noisy movement and stabilize that. We will try out different architectures. Our main approach will be to learn the stabilization parameters through time, so we will try to use Recurrent Neural Networks (RNN) that learns the features in time to stabilize the video along with Convolutional Neural Networks (CNN) because we are working with images.

(c) After we have our dataset, losses and architectures we will train the model and test the model with new videos and pickup the model that gives best result on unseen videos.