CS 763/CS 764: Task 06: Stable now

- Announced: 16 Mar. Due: Mar 26 10PM
- Please write (only if true) the honor code. You can find the honor code on the web page. If you used any source (person or thing) explicitly state it.
- **This is NOT an individual assignment**

1 Overview

Videos shot by amateurs usually suffer from undesirable shakes — you know, you have been there. Video stabilization is the process of compensating for these shaky motions. There are multiple ways in which this problem can be solved, including mechanical stabilization methods, optical stabilization methods, and digital stabilization methods. The first two methods involve designing specialized hardware to handle instability. In this assignment, we will be focusing on the third class of techniques, i.e., digital video stabilization.

1.1 Digital Video Stabilization

This class of methods takes an unstable video as input and produces a stabilized video by digitally processing the video. Out of the many ways to digitally handle this problem, the most popular way is to approach this problem from a visual motion estimation point of view. The camera's movement introduces apparent motion into the objects present in the scene. Hence, the motion of objects perceived in a video is a combination of independent and apparent motion due to camera movement. If the camera path is stable, the video shot would also be stable and vice versa.

Based on this intuition, the majority of the visual motion estimation-based stabilization techniques follow the following three-stage template: 1) Motion Estimation 2) Motion Smoothing 3) Retargeting the frames based on the smoothed motion. Here, we hope you will be progressively developing basic to more advanced solutions for video stabilization as you walk through Tasks 1, 2 and 3.

There are two significant ways in which visual motion is estimated in computer vision viz. 1) Motion estimation using sparse feature points 2) Motion estimation using (dense) optical flow. **Either way**, the motion between consecutive frames for intrinsically non-moving objects (e.g., the background) is estimated compactly as either a similarity transform, or an affine transform (that is contributed by the instability of the camera). In short, the unstable camera results in the scene having unnecessary movement (which is easily perceived by the most naive viewer).

2 Tasks

For each of the questions, you are expected to approach the problem through the twin ideas of less ('sparse') and more ('dense'). You are expected to analyze these two cases using either qualitative or quantitative metrics of your choise and decide which method you'd recommend to your favourite app (instagram or snapchat).

Three sets of input folders containing shaky videos have been provided with obvious naming conventions. The questions are described first in broad terms, and the coding conventions are mentioned mainly in the first part leaving the others to be completed by extrapolation.

Also see sample output (link will be posted later).

1. **Q1:** 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:

(a) Motion Estimation with OpenCV

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

ii. Optical Flow: Use (dense) opencv opticalflow `calcOpticalFlowFarneback()` in order to obtain compact motion parameters

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

(c) Correct all frames using the first frame as the reference.

2.1 Programming conventions

Run example :

```
# optical flow version corresponds to B
python StabilizeVideo.py --question 1 --subQuestion B --video 1
# output lands in 1_B
python StabilizeVideo.py --question 1 --subQuestion A --video 1
```

(a) Adapt the provided stub in `VideoUtils` to read and write videos

(b) Commands should have a "question" argument and a video file name argument. Only the basename of the video is to be specified, not the entire path. (See example).

(c) Write results in the `Output` folder (directory). Use the question number and the input file name to create folder and filename (e.g. `Output/2_B/1.avi`)

(d) We think that the only external libraries that are allowed to be used are *numpy* and (stock) *opencv*. There should be no need for contrib.

(e) Stock OpenCv contains a video stabilization C++ module (`videostab`) but presumably there are no python bindings which seem to indicate that it is deprecated. You are not to use this video stabilization module.

(f) Third party `videostab` modules are available (with `pip`, for instance). You are not to use these as is. If you do end up looking up the source code and imitate, (with no penalty for doing this), you must clearly articulate the part you sourced, and what is new.

Do not forget Write down your observations. Which method is 'superior'? How did you establish this? Report all quantifiable parameters (and be careful in reporting correct time). Use time per frame to quantify your answer.

2. **Q2:** 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).

Run example :

```
cd Q1
cp -al ../Q2/1.avi ./5.avi  #1.avi now in Q1  with new name
python StabilizeVideo.py --question 1 --subQuestion B --video 5
# Run program of Q1 on video 5.avi
python StabilizeVideo.py --question 1 --subQuestion A --video 5
```

(b) Motion Estimation

i. Feature point descriptors: 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.

Figure 1: Example frame grab of output for Q2

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

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

Run example :

```
python StabilizeVideo.py --question 2 --subQuestion B --video 1
# stock open cv optical flow
python StabilizeVideo.py --question 2 --subQuestion C --video 1
# deep learning derived flow
```

2.2 Programming conventions

(a) Use the provided stub in `flowUtils` to read optical flow. A function here returns optical flow as a tensor of dimension $M \times N \times 2$, where a location (m, n) gives the displacement vector (x, y) for the pixel at location (m, n) of the source image. For this question, you are given pre-computed optical flows in the relevant folder. The flow files are stored in `.flo` format. The flow between two consecutive frames i and $i + 1$ is saved as $i.flo$. In the `Videoutils` folder you are provided a file `flowUtiliy.py`. This contains a function `readFlow()` which takes the path to a flow file as input and outputs a numpy array containing the optical flow.

Do not forget As before write down your observations.

3. **Q3:** 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.

   (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.

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

4. **Q4:** 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.

   (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).

3 Submission Guidelines

Your lab submission folder should look something like:

```
130010009_140076001_150050001_Task06
├── ReflectionEssay.pdf
├── answersQ1Q2Q3Q4.pdf
├── code
│   ├── StabilizeVideo.py
│   └── Utils
│       └── VideoUtils.py
├── Output
│   ├── 1_A
│   │   └──
│   └── 1_B
├── data
│   ├── Q1
│   │   └── mydata.avi
│   └── Q2
│       └── mydata.avi
└── readme.txt
```

Note: Do not include the flow or the provided videos or the output videos.