Video Stabilization with Classical Methods

Jianqiao Li
Tandon School of Engineering
New York University
Email: jl7136@nyu.edu

Tyler Lin
Tandon School of Engineering
New York University
Email: yl8798@nyu.edu

Xiao Shen
Tandon School of Engineering
New York University
Email: xs2156@nyu.edu

***Abstract*—One of the most apparent video quality differences between professional and amateur level contents resides in camera motion stability, and while the industry has developed various optical sensors/equipment that specifically target in-motion-shooting stabilization, those approaches are sometimes unavailable, or impractical for non-professional content creators. In light of the obstacles above, this project aims to investigate a software-level solution to video stabilization. We decided to use classical methods in this project for implementing video stabilization.**

1. Overview

The classical methods for video stabilization involve the following process: We first use detectors to identify points of significance in every frame, then an affine transformation is estimated between each frame based on how the points of significance have shifted place. Since the transformations from shaky videos are too strong, we then need a way to stabilize the transform so that they become milder. Finally, we apply the stabilized transformations to the original frames, thus reducing the relative camera motion.

Jianqiao Li is responsible for implementing a version which uses average filters to stabilize the trajectory.

Tyler Lin and Xiao Shen are responsible for implementing a version which uses L1 optimal camera path and constraints to stabilize the trajectory.

2. Project Accomplishment

2.1. Average Filter Method

2.1.1. Method. Average filtering method for video stabilization involves three processing components: feature extraction, object-flow tracking, and camera-motion smoothing. While in real-life scenarios, camera motions can get complicated involving various jittering, we can safely assume that for the purpose of research, camera motion can be mimicked via Euclidean transformation(or, Similarity transformation)on a 2D model employing scaling, translating and rotation; this assumption of simple camera motion allows for a robust and efficient average filter algorithm.
For the implementation of features extraction, we chose to use the built-in feature detector function **goodFeaturesToTrack()**from OpenCV libraries considering the focus on trajectory smoothing (we may implement different feature extraction algorithms by our final stage of research). Also, in light of computational efficiency, we only choose top 50 feature points to track.
We then proceeded to object tracking using the classic Lucas-Kanade Optical Flow algorithm (Baker et al., 2004), implemented in OpenCV as **calcOpticalFlowPyrLK()**.The optical flow algorithm captures the inter-frame motion that can be decomposed into x and y coordinate difference, which could also allow us to calculate the corresponding rotation parameter.
With decomposed parameters from optical tracking when iterated through all video frames, we then turned to trajectory smoothing using the average filter on accumulated sum of our trajectory matrix. The smoothing is achieved by simply replacing inter-frame motion for each three tracked curves with their neighbor's mean via a small window (we chose 5 frames as the averaging radius), the proper mathematical illustration of averaging radius of n (assuming odd) is shown below (Smith, 2013):

$$f(k) = \frac{c(k-\frac{n}{2})+c(k-\frac{n}{2}+1)+...+c(k)+c(k+\frac{n}{2}-1)+c(k+\frac{n}{2})}{n}$$

where $f(k)$ is the value of the smoothed trajectory curve at point k, and $c(k)$ the original.
When completed smoothing all three trajectories, we add new transformation matrix calculated from smooth trajectory matrices back to original frame.

2.1.2. Partial Results. The moving average filter model is tested on 2 10-second cropped video clips, while the algorithm removes some of the camera jittering on the first test clip, it exercises little improvement on the second clip. One of the main causes might be that the second test clip has significant high-frequency jittering, and the frequency is higher than the threshold our chosen averaging radius can capture. Meanwhile, in order to get rid of the black boundaries when the algorithm tries to stabilize the video through frame shrinking, we implemented **getRotationMatrix2D()** from OpenCV libraries to prevent frame image from deviating from its center. In conclusion, this algorithm is computationally inexpensive and robust, but only works well on videos with infrequent camera vibrations. Figure 1 below demonstrates the incapability of average filter when processing videos with high moving speed or frequency.

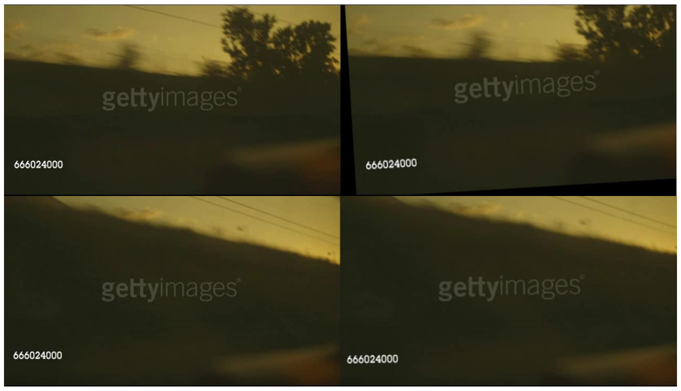

Figure 1. running train view clips

The top left and right are original clip screenshots and stabilized screenshot respectively in the first second. One explanation to why there is an extensive deviation from frame center is that the objects in the original video frequently change over short period of time, while our averaging window still takes in trajectory parameters from old objects that might not exist in the frame, and this is apparently one of the drawbacks using average filtering on videos with high frequency of content changes given fixed filter radius. The bottom left and right screenshots are examples of common techniques, **warpAffine()**, used to fix the boarder when the shape of processed frame is not in accordance with the original, as we can see that the bottom right frame has been zoomed in with scale of 5%. However, this approach indeed sacrifices resolution and creates video inconsistency, and we shall address on potential optimizations and resolutions to the problem discussed above in the final report.

2.2. L1 Optimal Camera Path Method

2.2.1. Method. The L1 Optimal Camera Path method differs from the average filter method in trajectory stabilization. Instead of averaging over the trajectories, L1 Optimal Path method introduces constraints on the camera path in order to mimic a camera path that conforms with cinematographic characteristics, turning stabilization into a Constraint Satisfaction Problem (CSP). The main objective function that we are trying to minimize is the camera path, or three derivatives of the camera path $P(t)$. The first derivative of the camera path stands for its velocity.

$$DP(t) = 0$$

where D is the differential operator. Minimizing it will minimize the relative motion of the camera.

$$D^2P(t) = 0$$

The second derivative represents the acceleration which we want to minimize to account for camera moving at a constant velocity.

$$D^3P(t) = 0$$

The third derivative represents the rate of increase of camera acceleration. Minimizing it will try to stabilize the acceleration of the camera. We minimize these three derivatives at the same time, in order to mimic the camera motion that seem natural.
There are three constraints mentioned in the research paper (Grundmann et al., 2011). Inclusion constraint, proximity constraint and saliency constraint. Inclusion constraint ensures that the video after applying stabilized transforms always stays in the cropped window. The Proximity constraint ensures that the stabilized trajectory should preserve the original motion of the video. The saliency constraint states that all saliency points should be included in all or parts of the cropped window. The original paper (Grundmann et al., 2011) states that the saliency constraint should be implemented as a soft constraint to avoid violent tracking of the saliency points. As of the time of this report, only the inclusion constraint is included in our implementation.

In our implementation, the objective function and the constraints are set up before using the CVXR package to form them as a constraint satisfaction problem. Then a solver is used to solve the CSP.

For feature detection, we are using three kinds of detectors. The SIFT detector, HARRIS detector and ORB detector. SIFT is the very first detection we decided to apply. We can take advantages of scale invariance and rotational invariance of SIFT features, for which, it is not disturbed by viewing angle changes, affine transformations, and noise. According to our learning of SIFT, it firstly select key-points from scale-space by using DoG(Difference of Gaussians). From those points, remove those point lie on the edge or with really low contrast. For the rest of the key-points, we have scale invariance, and we'll have rotational invariance after the orientations are assigned for the points. Then those points could be used for L1 Optimal Camera Path method.We also apply HARRIS and ORB as comparison. Different from SIFT, Harris detector extract corners and infer features of frames. The corner point is important to an image, which can effectively reduce the amount of information data while retaining the important features of images. However, HARRIS has grayscale invariance and rotation invariance, but not scale invariance, which is very important for local features of images, we can combine the application of HARRIS and

Gaussian scale space to solve this problem.ORB combines FAST and BRIEF, for which we want to take advantage of its speed, we also want to compare its performance with SIFT and HARRIS

2.2.2. Partial Results. For feature detection, we are using three kinds of detectors. The HARRIS detector, SIFT detector and ORB detector. Using The Harris detector and Sift detector yielded similar results, while the SIFT detector requires more computing power and time. The ORB detector is the fastest, and produces similar results. When no cropping is done, the frame after applying stabilized trajectory may move out of the window, and there may be missing pixels in the window.

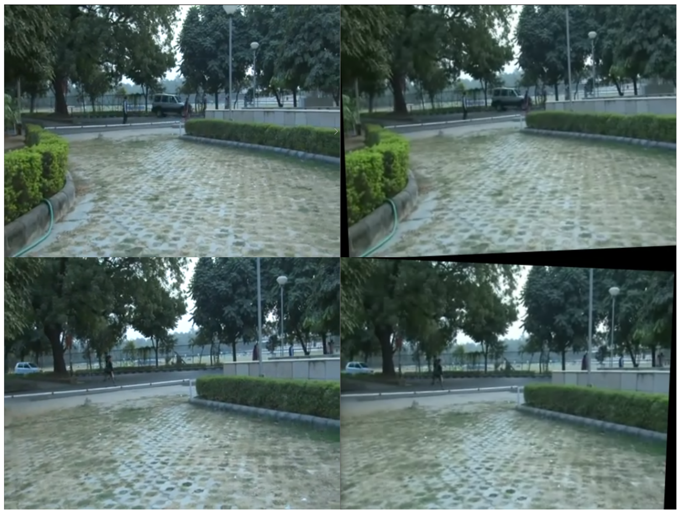
Figure 2. selected frames of static video

When applied to a static video, where the intended camera path is relatively stationary, the stabilization is able to cancel out some jitters. Since no cropping is used here for showcasing the results, we can see missing pixels and the shifted frame after applying the smoothed trajectory.

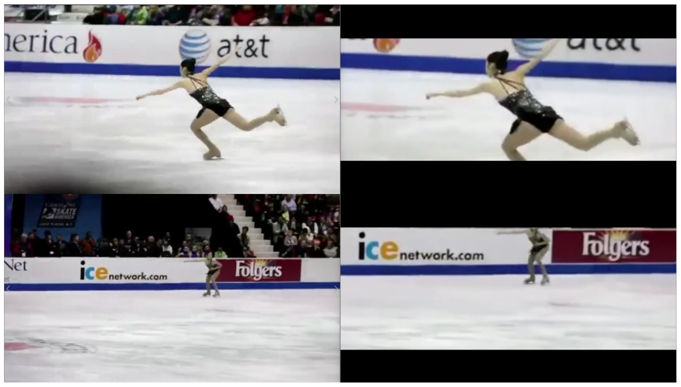

Figure 3. selected frame of moving video

We also tried our implementation on a moving video where the camera pans and accelerates to track a subject. The figure skater is moving very fast and the camera is tracking her but with very shaky motion. The stabilization is able to cancel out most of the camera's high frequency jitters and make the figure skater's movements much more pleasing to watch. Cropping is also used in this case. In order to have a crop window with no missing pixels, the effective resolution of the stabilized video is only 30 percent of that of the original video. For highly shaky videos, the resulting crop will be very small because of the larger offsets needed to cancel out the shaky motions. As we can see in the second pair of frames, even the figure skater's head got cropped, which defeats the purpose of stabilizing video and tracking its subject. Ideally, to track a subject perfectly we would need a sufficiently large frame where the subject is small and centered most of the time. However, this is not how people normally shoot videos.
There is also a short sequence where the camera pans nearly 90 degrees to focus on the skater, during which the stabilization fell a little short and was unable to cancel out unwanted jitters. This potentially can be improved by implementing the proximity constraint and the saliency constraint.

3. Remaining Work

For improvements upon our L1 method implementation, we need to first implement the remaining two constraints. With these two constraints we might be able to stabilize the video even more, since the current result from this method still has a lot of room for improvement. In addition to trying different feature detectors, we can also try different numbers of features to detect. For the CSP solver, we are currently using ECOS. It would be beneficial to explore the other two solvers provided in the package which are SCS and OSQP. Finally, proper window cropping needs to be done such that the resulting video does not have any missing pixels, and looks like a plausible video produced by the original camera. Tyler Lin will be responsible for this part and this will be done around Apr.20th.

We also need to select a robust metric to numerically evaluate our results and compare them. A common technique is to measure the Structural Similarity (SSIM) and inter-frame transformation fidelity (ITF) between each pair of adjacent frames, which are expected to be higher provided the video motion are smooth and few distortions occur (Guilluy et al., 2021). Another metric could be the minimum cropping ratio which also represents the loss of resolution. A good method should provide acceptable stabilizing result while losing as little resolution as possible. Since there is not a universal metric that is agreed upon for assessing the quality of video stabilization, we could introduce subjective assessment (Guilluy et al., 2021). A number of stabilized videos produced with different methods can be shown to different people where their preference can be recorded. Due to the nature of images and videos, subjective assessments are more important in our case, and can help us understand which results are acceptable and which are not. All members are responsible for this part, we will done evaluation and optimization around Apr.30th

For other methods, we could explore the Subspace method (Liu et al., 2011). This method employs a different stabilization technique, it tracked features in the video into a trajectory matrix, factors it into two low-rank matrices, and performs filtering in a low-dimensional linear space. We could also explore the method applied Kalman filter for video stabilization(Ohyun et al., 2005).This method compute the optical flow between two frames, and then use the points detected from optical flow to estimate the motion parameters. Kalman filter will be applied for smoothing those parameters and then warp frames by using smoothed parameters.Xiao Shen will be responsible for this part and one of these two method will be chosen. This will be done around Apr.20th.

References

[1] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, *Subspace Video Stabilization*. ACM Transactions on Graphics, vol. 30, no. 1, pp. 1–10, 2011.

[2] Kwon, Ohyun Shin, Jeongho Paik, Joonki. (2005). *Video Stabilization Using Kalman Filter and Phase Correlation Matching.*141-148. 10.1007/11559573_18.

[3] M. Grundmann, V. Kwatra, and I. Essa, *Auto-directed video stabilization with robust L1 optimal camera paths.*CVPR 2011, 2011.

[4] W. Guilluy, L. Oudre, and A. Beghdadi, *Video stabilization: Overview, challenges and perspectives*. Signal Processing: Image Communication, vol. 90, p. 116015, 2021.

[5] S. Baker and I. Matthews, *Lucas-Kanade 20 Years on: A unifying framework*. International Journal of Computer Vision, vol. 56, no. 3, pp. 221–255, 2004.

[6] S. Smith, *Digital Signal Processing: A practical guide for engineers and scientists: A practical guide for engineers and scientists*, Newnes, 2013.