Video Stabilization Project Report

Wangzhe Sun

***Abstract*—This report presents a digital video stabilization system that assumes the input video is stationary (defined in section III). The system can reduce the instability index (defined in section VIII) by up to 62.5%. This report examines the performance of this system using different mainstream feature extraction algorithms (ORB, Good Feature to Track, and SIFT). The results show that the system is compatible with most mainstream feature extraction algorithms, and ORB is the most suitable for the video stabilization application (both in efficiency and performance). Meanwhile, an exhaustive evaluation is made for the system under different video conditions. Numerical and visual results indicate that the system is robust in most settings while malfunctioning in certain situations.**

***Index Terms*—computer vision, video stabilization, feature extraction, feature tracking, euclidean transformation**

I. Introduction

THE rapid development of video capturing devices has extended the use of video in vast scenarios, ranging from home-use video recording to scientific research. These days, video recording is accessible to almost everyone with a single click on a cell phone. However, the development of video devices does not eliminate distortions during videoing, especially for personal recordings without professional devices. One of the most common distortions is the image instability caused by unintentional camera movements. More than just visual annoyance, shaky videos can sometimes lead to visual discomfort and even dizziness or sickness[1].

This report presents a simplified video stabilization system to alleviate the distortions caused by camera instability. The system is implemented in PyCharm IDE using the computer vision library OpenCV. This project follows the video stabilization tutorial[2] and borrows part of its code. This report intends to offer a detailed overview of the video stabilization system and examine the performance of mainstream feature extraction algorithms in this specific application.

The report is organized as follows. Section II describes some relevant concepts and related work. Section III clarifies the objective of the system and certain assumptions this system depends upon. Section IV presents an overview of the video stabilization system, and sections V, VI, and VII discuss each component in great detail. Section VIII provides a numerical evaluation of this system using different feature extraction algorithms. Section IX summarizes the project and discusses its limitation together with future steps.

II. Background

Different approaches have been introduced to solve the video stabilization problem. The video stabilization systems can be broadly divided into mechanical, optical, and digital solutions.

Mechanical video stabilization systems use the motion detected by mechanical sensors to move the image sensor to compensate for the camera's motion. This approach typically uses gyroscopes to detect motion and send signals to motors connected to small wheels so that the camera can move in the opposite direction of motion. Slightly different from mechanical video stabilization, optical video stabilization systems[3] compensate for the camera motion by using movable lens assembly that variably adjusts the path length. Digital video stabilization systems differ from mechanical and optical video stabilization systems because they do not have special hardware requirements. It stabilizes videos by processing video frames after the video is taken. This approach is typically slower than in-place video correction methods like mechanical and optical approaches. However, it remains competitive due to its near-to-zero cost and comparative stabilization qualities[4].

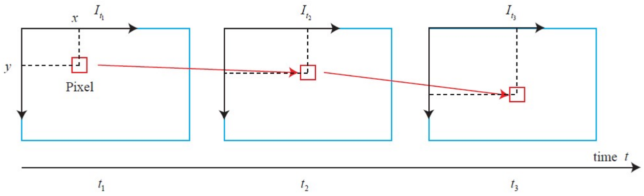

Fig. 1: Illustration of constant grayscale assumption

III. Objective and Assumption

The video stabilization system this report presents is a digital video stabilization system. It aims at a special video stabilization problem: the stationary video stabilization problem. A stationary video is defined as a video that, under ideal conditions, has a fixed camera location and orientation. It is opposed to mobile video, where the camera moves during videoing process. This report's system relies on stationary video and constant grayscale assumptions. The stationary video assumption assumes every input video is a stationary video. The constant grayscale assumption assumes the corresponding pixel's grayscale is constant in each video frame. Consider the image as a function of time $\boldsymbol{I}(t)$. Then, for a pixel at (x, y) at time t, denote its grayscale as $\boldsymbol{I}(x, y, t)$. The mathematical formulation of the assumption states that for the pixel at (x, y) at time t, suppose it moves to $(x + dx, y + dy)$ at time $t + dt$, there exists an equation $\boldsymbol{I}(x + dx, y + dy, t + dt) = \boldsymbol{I}(x, y, t)$ (Fig. 1).

IV. System Overview

The principle of stationary video stabilization is to smooth the trajectory of the involuntary camera movements. Fig. 2 illustrates the basic idea of the principle. The video stabilization system divides the process into three main components (Fig. 3): initialization, trajectory estimation, and frame stabilization.

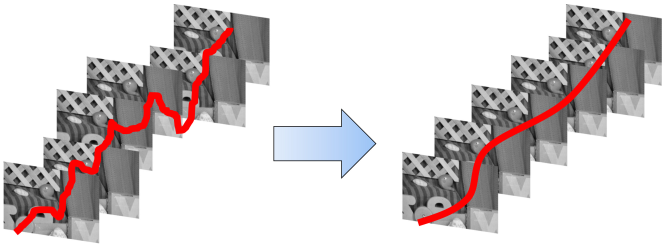

Fig. 2: Principle of video Stabilization[5]

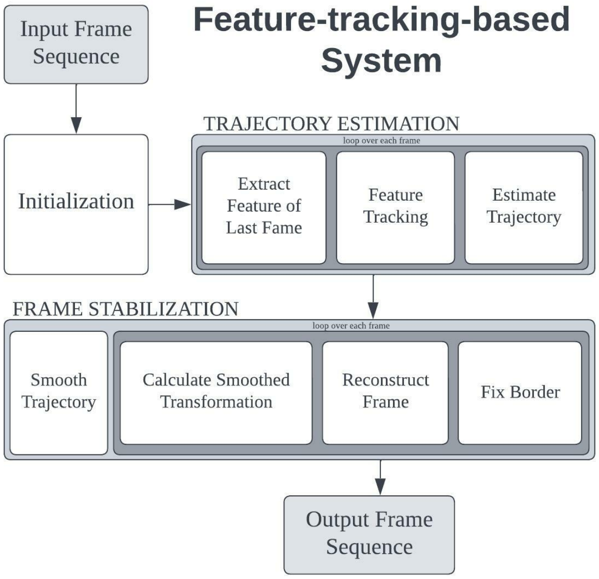

Fig. 3: Video stabilization system overview, showing all the steps performed by the trajectory estimation and frame stabilization steps.

These components are performed in sequential order, which means the latter component depends on the results of previous components. The initialization part is a procedure to initialize variables needed for later work. The trajectory estimation component calculates the accumulated trajectory of all frames. Finally, the frame stabilization component smooths the trajectory obtained in the last component and then reconstructs a stabilized video. The following few sections present the technical details of each component.

V. Initialization

The video stabilization system processes video as continuous image frames, making it necessary to set it up before any procedure starts. This initialization component sets up the member variables, including video object, output writer, and key-point containers, all of which are required in later procedures.

VI. Trajectory Estimation

This section describes the trajectory estimation component of the system. The goal of the trajectory estimation component is to obtain the estimated trajectory of the camera. It comprises three individual sub-tasks: feature extraction, feature tracking, and trajectory estimation. All three tasks are performed in a loop for each video frame.

A. Feature Extraction

The system extracts feature points from the previous frame for each video frame. This feature extraction step can be done using different feature extraction algorithms. Indeed, section VIII tests several mainstream feature extraction algorithms (ORB, Good Feature to Track, and SIFT), all performing properly in the system.

B. Feature Tracking

Once the system obtains the feature points of previous frames, it starts tracking these feature points' positions in the current frame. The system uses Lucas-Kanade Optical Flow (LK optical flow) as the feature tracking algorithm. The crux of LK optical flow is to estimate pixel translations in the x and y directions on the image frame. According to the optical flow, the movement in x and y directions can be obtained by solving

$$\begin{pmatrix} u \\ v \end{pmatrix}^{*} = -(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T b \tag{1}$$

where $A = \begin{bmatrix} [\boldsymbol{I}_x & \boldsymbol{I}_y]_1 \\ \vdots \\ [\boldsymbol{I}_x & \boldsymbol{I}_y]_k \end{bmatrix}$ is a matrix whose rows are image gradient in x and y directions for all feature points, $b = \begin{bmatrix} \boldsymbol{I}_{t1} \\ \vdots \\ \boldsymbol{I}_{tk} \end{bmatrix}$ is a vector whose elements are changes in image brightness with respect to time for all feature points. Lk optical flow algorithm relies on the constant grayscale assumption, which is one of the system's assumptions. The mathematical deviation of LK optical flow is provided in Appendix A.

C. Trajectory Estimation

The last step of this component aims to estimate the transformation of each frame from the last frame. Given corresponding feature points of both the last frame and the current frame, the system can obtain a 2-D Euclidean transformation matrix, $\boldsymbol{H}$, which is of the form

$$\boldsymbol{H} = \begin{bmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The first two elements of the third column of $\boldsymbol{H}$ are the translations in the x and y directions, and the upper-left corner consists of information about the rotation angle of the frame. The translation values can be obtained by extracting the third column, and the rotation value can be obtained by applying trigonometry. These three transformation values are stored for each video frame, which is the trajectory information for each video frame.

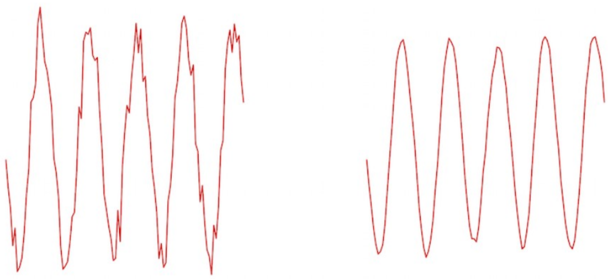

Fig. 4: Illustration of effect of moving average filter. Left: noisy curve. Right: smoothed curve

VII. Frame Stabilization

This section introduces the last component of the video stabilization system. This component constructs a new transformation matrix for each frame by calculating a smoothed trajectory of the camera motion and then reconstructing each frame using the updated transformation matrix. The frame stabilization component consists of four individual sub-tasks: trajectory smoothing, smoothed transformation calculation, frame reconstruction, and border fixation. The last three tasks are performed in a loop for each frame.

A. Trajectory Smoothing

The trajectory information stored in the system consists of x-translation, y-translation, and the rotation angle. So, there are three curves for us to smooth. The smoothing technique employed by the system is moving the average filter. This technique uses a sliding window and replaces the function's value at each point with the average of the neighbor in the window.

The mathematical formulation of this process can be summarized as follow. Let's say the values of a function is stored in an array f, and $f[0]$, $f[1]$, ..., $f[k]$ are points of this function. Let s be the array of points for the smoothed function after performing the moving average filter with a window size of n (n should be an odd integer). Then the k^{th} element of the array s is obtained using the function

$$s[k] = \frac{f[k - \frac{n-1}{2}] + ... + f[k] + ... + f[k + \frac{n-1}{2}]}{n} \tag{3}$$

This technique can smooth a curve, and an illustration of its effect can be seen in Fig. 4. The same smoothing procedure is performed on three curves separately in the system.

B. Smoothed Transformation Calculation

For each frame, its new transformation information (x-translation, y-translation, and rotation angle) can be obtained with ease by first finding the difference between the smoothed trajectory and the original trajectory of the frame and then adding this difference back to the original transformation information.

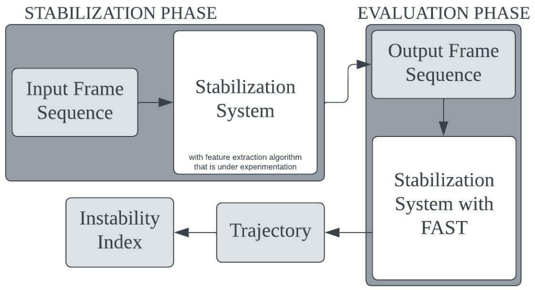

Fig. 5: Pipeline of the evaluation procedure

C. Frame Reconstruction

This step reconstructs the video frame by frame and is precisely the reverse of the trajectory estimation procedure (Section VI, part C). What the system has at this point is smoothed transformation information: the rotation angle θ, the x-translation dx, and the y-translation dy. Using these values, a corresponding 2-D Euclidean transformation matrix, $\boldsymbol{H}$, can be constructed, which can be further used to obtain the new frame.

D. Border Fixation

Transformation involves translation and rotation and unavoidably leads to inconsistency between the new frame size and the original frame size. As a result, the output video may contain frames surrounded by unexpected black borders. So, this ending step aims at alleviating this phenomenon. The system uses a straightforward technique: it zooms the inner part of each frame for the black borders to be out of bounds.

VIII. Experiment

This section performs experiments of the system on three videos (video files named 'video_1.mp4', 'video_2.mp4', and 'video_3.mp4'). Each video experiment uses three different feature extraction algorithms: ORB, Good Feature to Track, and SIFT.

A. Metrics

The evaluation statistics used in this project is the instability index (the lower, the better), which measures how shaky a given video is. It is defined as the square of the 2-norm of the average trajectory. Denote the trajectory of the video as $\boldsymbol{T}$ and the frame number of the video as n, a mathematical formula of the instability index, i, is

$$i = ||\frac{\boldsymbol{T}}{n}||_2^2 \tag{4}$$

The evaluation procedure for this project follows a strict pipeline (Fig. 5). The process is composed of a stabilization phase and an evaluation phase. In the stabilization phase, the input video is fed into the video stabilization system with the target feature extraction algorithm to obtain the stabilized output video. Then in the evaluation phase, the output video

TABLE I: Experiment result using video 1

Algorithm	Time / s	Instability Index (lower the better)
ORB	24.96	19.46
GFFT	31.74	20.13
SIFT	85.41	20.53
Original Video	N/A	25.8

TABLE II: Experiment result using video 2

Algorithm	Time / s	Instability Index (lower the better)
ORB	16.67	3.08
GFFT	21.35	7.65
SIFT	55.41	5.37
Original Video	N/A	8.22

TABLE III: Experiment result using video 3

Algorithm	Time / s	Instability Index (lower the better)
ORB	12.77	55.76
GFFT	15.45	96.51
SIFT	39.54	102.72
Original Video	N/A	85.8

is processed again by a video stabilization system with the FAST algorithm. For the second-time system processing, only trajectory information is required. The instability index is then calculated from this trajectory information.

It is highly likely that the system with the same feature extraction algorithms in both stabilization and evaluation phases tends to have a lower instability index than using different algorithms, regardless of the actual performance. So, the evaluation phase uses a system with FAST as the feature extraction algorithm, different from all three algorithms used for experiments in this project.

B. System Performance using Video 1

Video 1 is an 18-second video simulating the instability of a hand-held camera in real-life settings. The video is processed three times separately using the system with three different feature extraction algorithms. The output video files are named 'stablized_vid_1_orb.avi', 'stablized_vid_1_gfft.avi', and 'stablized_vid_1_sift.avi', respectively. Table I lists the numerical results of this experiment. After video stabilization with any of the three feature extraction algorithms, the output video achieves an instability index much smaller than that of the original video (about 22% lower). The visual comparisons between all three output videos and the original ones also indicate that the system works properly, consistent with the numerical results. Meanwhile, ORB achieves the lowest instability index within the shortest time, and SIFT achieves the highest instability index within the longest time (about four times slower than ORB).

C. System Performance using Video 2

Video 2 is a 12-second video with high-frequency perturbations. Like the first experiment, video 2 is processed three times separately using the system with three different feature extraction algorithms. The output video files are named 'stablized_vid_2_orb.avi', 'stablized_vid_2_gfft.avi', and 'stablized_vid_2_sift.avi', respectively. The numerical results are listed in Table II. The system works properly with all three feature extraction algorithms from the numerical values, and ORB achieves the lowest instability index within the shortest time. However, a visual comparison reveals little difference between the smoothed video and the original one in the sense of the shaking. This inconsistency between numerical and visual results reveals that the system works but is ineffective in dealing with high-frequency perturbations. This problem will be further discussed in section IX, part B.

D. System Performance using Video 3

Video 3 is an 8-second video with strong fluctuations between frames. It is processed three times separately using the system with three different feature extraction algorithms as well. The output video files are named 'stablized_vid_3_orb.avi', 'stablized_vid_3_gfft.avi', and 'stablized_vid_3_sift.avi', respectively. The numerical results listed in Table III show that all systems except that using the ORB algorithm achieve an instability index much worse than the original video. The visual results are consistent with the numerical statistics. The worsening stability is partly due to the malfunction of feature tracking in the system and will be further discussed in section IX, part B.

E. Result

The system works in the first two experiments but fails the third, implying that the system only works properly in limited settings. Section IX, part B will discuss the conditions that will lead to such system malfunction. This section only evaluates statistics from the first two experiments. Among all three feature extraction algorithms used in the experiment, ORB proves to be the best suitable algorithm for this particular system. Its performance is, on average, 31.5% better than GFFT and 23.9% better than SIFT. Meanwhile, the ORB system is 21.6% faster than using GFFT and more than 70% faster than using SIFT.

IX. Conclusions and Discussion

A. Conclusions

This work presents a simplified digital video stabilization system with details of each component and exhaustive evaluations of incorporating different mainstream feature extraction algorithms. The system demonstrates its robust performance in most settings with all three feature extraction algorithms, among which ORB proves to be the best suitable algorithm for this particular system. The system provides stability for videos with low-frequency and small-magnitude amplitude. However, strict restrictions for perturbation frequency and frame translation extent are limitations for the system. Implementing certain mechanisms to loosen these restrictions can be the future step of this project.

B. Limitations

The video stabilization system this report presents has potential limitations. The system can not work properly if the input video meets any of the following conditions: (1). the video involves high-frequency perturbations; (2). the video involves translations greater than the frame size between two consecutive frames ; (3). the video involves severe changes in light conditions. The remaining of this part discusses each limitation in great detail.

1) High-frequency Perturbations: High-frequency perturbations refer to the continuous high-frequency and small-amplitude translations between frames. Video 2 is an example of this type of situation, and it is usually caused by the sudden and high-frequency shaking of the camera. This report's system cannot handle this shaking correctly, partly due to the moving average filter technique used for smoothing curves. The moving average filter uses a sliding window to average the curve within a specific window size. For curves with high-frequency fluctuations (small up-and-down fluctuations along the curve), a sliding window with a small window size cannot eliminate the fluctuations, and a sliding window with a large window size would produce a video losing too much information.

2) Too large translation between frames: The nature of a feature-based system requires that the input video of the system does not contain consecutive frames with translation extent exceeding the frame size. This limitation comes from the assumption used by the LK optical flow algorithm, which assumes that pixels in a certain window have the same motion (see Appendix A for detail). If the translation extent is too big, the feature tracking will provide unreliable estimations for the trajectory between frames, ultimately leading to a bizarre output.

3) Changes in Light Conditions: The system this report presents is sensitive to severe changes in light conditions. This limitation derives from the constant grayscale assumption of the system. If the light condition changes drastically in the video, the core equation LK optical flow based on, $\boldsymbol{I}(x+dx, y+dy, t+dt) = \boldsymbol{I}(x, y, t)$, does not hold, and would similarly produce unreliable estimations for the trajectory.

C. Future Work

The video stabilization system this report presents provides a simplified solution for stationary video stabilization. The system can still be improved in certain ways. One way to improve the system's performance is to replace the feature tracking process with a feature matching process (Fig. 6). The feature matching technique achieves correspondences between pixels of the last and current frames by matching feature points extracted from the two frames. This eliminates the possible influence of light conditions on the system since it no longer depends on feature tracking. However, at an expense, the feature-matching-based system requires a longer processing time as feature extraction will be performed twice (both the last and the current frame) for each frame. Although the processing time can be further improved by storing each corresponding frame's feature points and descriptors, this only works for short videos and will ultimately reach an unacceptable space complexity as the video grows.

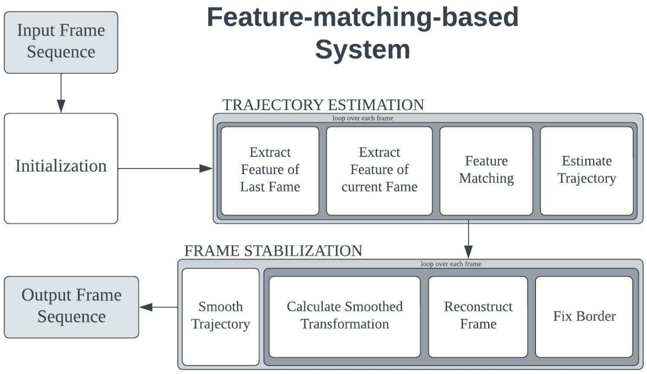

Fig. 6: Feature-matching-based System Overview

A possible extension of the system is to achieve real-time video stabilization. Thanks to the fast processing speed of ORB algorithms, the real-time system can be built on top of the existing system. It can work with a latency of about 44ms per frame (calculated from two experiments), which is within an acceptable range.

Appendix A
Lucas-Kanade Optical Flow[6]

An important assumption for the optical flow method is the constant grayscale assumption. According to the assumption, for the pixel at (x, y) at time t, suppose it moves to $(x+dx, y+dy)$ at time $t+dt$, there exists equation

$$\boldsymbol{I}(x+dx, y+dy, t+dt) = \boldsymbol{I}(x, y, t) \tag{5}$$

Applying first-order Taylor Expansion on $\boldsymbol{I}(x+dx, y+dy, t+dt)$ gives

$$\boldsymbol{I}(x+dx, y+dy, t+dt) \approx \boldsymbol{I}(x, y, t) + \frac{\partial \boldsymbol{I}}{\partial x} + \frac{\partial \boldsymbol{I}}{\partial y}dy + \frac{\partial \boldsymbol{I}}{\partial t}dt \tag{6}$$

Combining Eq. 5 and Eq. 6 leads to

$$\frac{\partial \boldsymbol{I}}{\partial x}dx + \frac{\partial \boldsymbol{I}}{\partial y}dy + \frac{\partial \boldsymbol{I}}{\partial t}dt = 0 \tag{7}$$

Devide both sides by dt:

$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{dx}{dt} + \frac{\partial \boldsymbol{I}}{\partial y}\frac{dy}{dt} = -\frac{\partial \boldsymbol{I}}{\partial t} \tag{8}$$

$\frac{dx}{dt}$ is the speed of the pixel on the x direction, and $\frac{dy}{dt}$ is the speed of the pixel on the y direction. Meanwhile, $\frac{\partial \boldsymbol{I}}{\partial x}$ is the gradient of the image in the x direction at this point, $\frac{\partial \boldsymbol{I}}{\partial y}$ is the gradient in the y direction, and $\frac{\partial \boldsymbol{I}}{\partial t}$ is the change of the image brightness with respect to time. Let's denote $u = \frac{dx}{dt}$,

$v = \frac{dy}{dt}$, $\boldsymbol{I}_x = \frac{\partial \boldsymbol{I}}{\partial x}$, $\boldsymbol{I}_y = \frac{\partial \boldsymbol{I}}{\partial y}$, and $\boldsymbol{I}_t = \frac{\partial \boldsymbol{I}}{\partial t}$. Then, the above equation can be written in the matrix form:

$$\begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\boldsymbol{I}_t \tag{9}$$

The goal of the LK optical flow algorithm is to solve for u and v in this equation, and they can not be solved using a single pixel. So, LK optical flow assumes that pixels within a certain window have the same motion. Consider a window of size $w \times w$, which contains w^2 pixels. This gives a total of w^2 equations:

$$\begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix}_k \begin{pmatrix} u \\ v \end{pmatrix} = -\boldsymbol{I}_{tk}, \quad k = 1, \ldots, w^2 \tag{10}$$

Denote $A = \begin{bmatrix} \begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix}_1 \\ \vdots \\ \begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix}_k \end{bmatrix}$, $b = \begin{bmatrix} \boldsymbol{I}_{t1} \\ \vdots \\ \boldsymbol{I}_{tk} \end{bmatrix}$. The whole equation can be stacked into one:

$$\boldsymbol{A} \begin{pmatrix} u \\ v \end{pmatrix} = -b \tag{11}$$

This is an over-determined linear equation, so the least-square solution is

$$\begin{pmatrix} u \\ v \end{pmatrix}^* = -(\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T b \tag{12}$$

With the knowledge of the time change dt, the pixel's new position can be calculated using the changes in both axes (dx and dy) once u and v are solved.

ACKNOWLEDGMENT

The author would like to thank Dr. Alan Peters for his support and instructions for this project.

REFERENCES

[1] *Shaky Camera.* https://en.wikipedia.org/wiki/Shaky_camera.

[2] A. S. Thakur, *Video Stabilization Using Point Feature Matching in OpenCV.* https://learnopencv.com/video-stabilization-using-point-feature-matching-in-opencv/, 2019.

[3] B. Cardani, *Optical Image Stabilization for Digital Cameras.* IEEE Control. Syst. 26(2), 21–22 (2006), 2006.

[4] M. R. Souza, H. Pedrini, *Digital video stabilization based on adaptive camera trajectory smoothing.* EURASIP Journal on Image and Video Processing volume 2018, Article number: 37 (2018), 2018.

[5] W. Guilluy, L. Oudre, and A. Beghdadi, *Video stabilization: Overview, challenges and perspectives.* Signal Processing: Image Communication, 2021.

[6] X. Gao, T. Zhang, *Introduction to Visual SLAM From Theory to Practice.* 2021. Available: https://github.com/gaoxiang12/slambook-en/blob/master/slambook-en.pdf