Towards A High-Performance and Causal Stabilization System for Video Captured from Moving Camera

Vu Nguyen Giap, Nguyen Binh Minh [*]

Hanoi University of Science and Technology - No. 1, Dai Co Viet, Hai Ba Trung, Hanoi, Vietnam

Received: March 05, 2018; Accepted: June 29, 2018

Abstract

Video shot from camera attached to moving devices like smartphone, and drone are often shaken because unwanted movements of the image sensors, which are caused by unstable motions of the devices during their operation (e.g. moving, fly). This phenomenon impacts on effectiveness of systems that use camera videos as input data such as security surveillance and object tracking. In this paper, we propose a novel software-based system to stabilize camera videos in real-time by combining several general models. The main contribution of proposed system is the capability of processing instantaneously video achieved from moving devices to meet quality requirements by using Harris with Optical-flow, and Lucas-Kanade methods for motion estimation. We also propose several mechanisms including frame partition and matching for corner detector when applying Harris method to ensure processing quality and system performance. In our system, we also use Kalman filter for prediction model of motion compensation. Our experiments proved that the average processing speed of our system can reach 35 fps, which satisfies the real-time requirement.

Keywords: Causal system, motion prediction, performance, real-time, video stabilization.

1. Introduction

Nowadays, in line with the development of hardware technologies, many devices such as vehicles, drones, and mobiles are equipped with cameras to provide video streaming for multiple monitoring purposes like instantaneous observation, object detection or tracking. However, due to limited size and structure, in most cases, those attached cameras cannot avoid mechanical vibrations, which are generated by unstable motions of the devices and environment factors. These vibrations cause uncontrollable movements of image sensors [1] and often seriously influences achieved video quality. Generally, with an unstable video, it is very difficult to effectively detect, and track interested objects. Therefore, the most important requirement is that shot videos should be stabilized through removing unwanted motions from host devices as well as image sensors. To approach the problem, software video stabilization techniques have been studied for decades and there are a lot of video stabilizing models have been proposed. The solutions operate based on image processing mechanisms. In this direction, video stability is attained through algorithms, which estimate camera motion trajectory.

Our goal in this work is to propose an effective model by combining and ameliorating several exiting techniques to improve performance in stabilizing streaming videos in real-time. There are two conditions for the real-time meaning here. First, the proposed system response time should be almost instantaneous in comparison with actual captured video from camera. Second, the processing system must be causal. In other word, the current frame stabilization uses only data obtained from this frame and previous frames in the past. If a system uses data extracted from subsequent frames to stabilize the current one, it cannot be a causal system as well as cannot respond in real-time. In this study, we put concrete requirements for our system as follows: (1) Improving image resolution that stabilization system can process to minimum of 640x480 pixels. (2) Improving speed processing to at least 33 milliseconds (corresponding to a frame rate of 30 fps), which suits almost all cameras with current general computer hardware configuration. (3) The proposed system must be causal. It means the system does not require knowing subsequent frames for stabilizing current frame.

To do that we focus mainly on improving the motion estimation stage by mechanisms as follows. For the corner detection, we employ Harris detector [10] because this algorithm is applied in turn and independent with each pixel in each frame. This approach would enable to calculate parallelly Harris algorithm, which significantly increases the overall processing speed. Instead of finding out corners in overall image frame one after another, we split the

[*] Corresponding author: Tel.: (+84) 967995584
Email: minh.nguyebinh@hust.edu.vn

frame into smaller partitions, then detect simultaneously corners in these image regions Besides, to restrict the corner detection in each frame, we suggest reusing gained results that repeat in previous frame. Finally, estimation of global motion model is replaced by Kalman filter [7, 8] as a prediction algorithm for the compensation stage. Through experiments, we proved that our improvements mentioned above can make system's stabilized videos to be only slower tens of milliseconds than the actual video and thus does not lose the system causality.

The rest of this paper is organized as follows. In Section 2, we analyze some related works to highlight our contributions. We present our system design and several mechanisms for the processing model to improve video stabilization performance in Section 3. Our experiments, gained outcomes, analyses and remarks are described in Section 4. Finally, conclusion and perspective are given in the last section.

2. Related work

According to [3], stabilization algorithms for videos are carried out with three main steps as follows: motion estimation, motion compensation, and image composition. As presented in Section 1, in this study, we focus on the second step to improve performance of entire stabilization system. To estimate image motion in video, most of existing studies use a certain feature detection as well as matching mechanisms to identify images. However, there is not any common definition for features of an image because the feature detection depends on different application targets. In video stabilization systems, features usually are defined as locations inside the image that have large gradient with all directions. The stabilization approaches are referred to as corners in [3, 5, 6] or an area in image frame presented in [4]. Lim et al. [3] developed a video stabilization system using Shi-Tomasi to detect corners and Optical-flow to match them in combination with the Lucas-Kanade algorithm [8]. In this way, motion model is estimated by means of a hybrid mechanism between rigid and similar transform. This trajectory is smoothed by an average window to obtain a trajectory with no undulation. The system was tested on a computer equipped with 1.7GHz CPU and gained average processing speed can be up to 30 fps with an input video of 640x480 pixels. Another method proposed by Vazquez et al. [5] used Lucas-Kanade feature tracker to detect interested points. Compensation for unwanted motions in this method is accomplished by adjusting the angle of rotation and the additional displacements that causes vibration. Through experiments, the authors shown that their approach could achieve processing speed from 20 to 28 fps for videos of 320x240 pixels on a MAC laptop with 2.16GHz Intel Core 2 Duo processor, 2GB RAM. The processing delay is only 3 frames with the authors' method. However, the corners pairing solution applied to this system uses a couple of current and previous frames or current and the next frames. Hence, the systems above are not causal systems.

There are several other methods like [4, 6] exploit corners for motion estimation step. A fast video stabilization algorithm introduced by Shen et al. [4] uses circular blocks to match and detect image features. The affine transformation thus is estimated based on motion parameters smoothed by a prediction method. However, this solution brings not very good stabilization performance: a video with resolution of 216x300 pixels can be processed with less than 10 fps speed. The authors' system is implemented on a desktop equipped with 3.0 GHz processor and 1GB RAM. In addition, in this approach, matching accuracy depends strongly on the appearance of moving objects in the selected areas. This characteristic affects the exact coupling process, and, in this way, it also reduces the algorithm accuracy. Wang et al. proposed a three steps video stabilizing method [6], in which Features from Accelerated Segment Test (FAST) detector is used to locate features in frames. Next, feature pairs are used to estimate affine transformation. Finally, motion estimation is executed based on that built affine model. According to the authors' tests, this method could handle up to 30 fps on a workstation computer equipped with an Intel Xeon processor of 2.26GHz and 6GB RAM for videos with a resolution of 320x240 pixels. Although the system proposed by Shen [4] is a causal system, however its stabilizing speed is very slow (less than 10 fps). Meanwhile, the stabilizing speed of Wang's system [6] is relatively high (up to 30 fps) and can be used in real-time. However, this speed achieved with a small video input (resolution of 320x240 pixels) while most current images have a minimum resolution of 640x480 pixels or larger. In [9], a dual video stabilization system uses an iterative method for estimating global motion. In addition, an adaptive smoothing window also is employed to estimate the intended movement among consecutive images. Unfortunately, due to the iteration approach, this mentioned method is only suitable for stabilizing offline video, though its processing speed can achieve up to 17 fps.

Through the discussion above, the current software solutions still have faced with the problems of non-causal system, real-time processing as well as low performance. In comparison with the existing efforts, our contributions of this work include: (1) Proposing novel combination of several existing algorithms together in single stabilization system including Harris, Optical-flow, Lucas-Kanade for corner detection, and Kalman filter for prediction model, which are applied to motion estimation and compensation step respectively. (2) Proposing novel mechanisms include frame partition and reuse of detected corners when applying Harris algorithm to the stabilization system to ensure processing quality and increase performance. (3) The proposed system is designed to provide causal characteristics that is the most critical point allowing real-time video processing.

3. Designing video stabilization system

3.1. Motion estimation

According to Harris [10], there are many feature types that can be chosen to represent an image, but one of the most effective methods to estimate motion parameters is to use corners. In this way, the motion estimation process is done in three steps: corner detection, matching, and estimating motion parameters. Our approach also is to detect corners in a frame then match them with corresponding corner in the next frame. Then the image transformation is estimated between these two consecutive frames. For the step detection, as mentioned before, we employ Harris detector [10] because this algorithm is applied in turn and independent with each pixel in each frame. Basically, the purpose of this algorithm is to find out variation intensity to displace (x, y) in all directions. This is simply expressed as follows:

$$E(x,y) = \sum_{u,v} w(u,v)\left|I(u+x,v+y) - I(u,v)\right|^2 \tag{1}$$

where w (u, v) is a rectangular window or Gaussian function, I am intensity of a pixel, and E (x, y) is intensity variation by a shift (x, y). Finally, a corner response is defined by Harris as follows:

$$R = \det(M) - k(trace(M))^2 \tag{2}$$

where:

$$M = \begin{bmatrix} w(u,v)\otimes\left(\frac{\partial I}{\partial x}\right)^2 & w(u,v)\otimes\left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ w(u,v)\otimes\left(\frac{\partial I}{\partial y}\right)\left(\frac{\partial I}{\partial x}\right) & w(u,v)\otimes\left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{3}$$

and det(M) is determinant of matrix M, trace(M) is sum of elements on the main diagonal of M, and k (0<k<0.25) corresponding to the sensitivity coefficient. At that point, corners are defined as the pixels with their corner responses R are the local maximums.

Furthermore, since pixels are near the borders of each frame that have very high probability that they will not appear in the next frame, so logically, we can eliminate corner detection in these areas to avoid wastage of computing time for these pixels. Otherwise, as mentioned above, with the used method, pixel processing is carried out successively in each frame. Consequently, needed processing time as well as performance are quite low. Instead of sequentially processing, in our model, we divide the processed image into smaller areas and detect corners in parallel on each of those partitions. After determining the corner positions, we need to find their respective places in the next frame. In this way, we can infer the local motion vector of each corner. In our system, Optical-flow algorithm [11] is used to accomplish this task. At that time, the relationship between intensities in two consecutive frames is shown as follows:

$$I(x,y,t) = I(x+dx, y+dy, t+dt) \tag{4}$$

Applying Taylor's expansion to the right-hand side of (4), approximate very small components, and divide all of them by dt. We have:

$$\frac{\partial I}{\partial x}.\frac{dx}{dt} + \frac{\partial I}{\partial y}.\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{5}$$

Set $u = \frac{dx}{dt}; v = \frac{dy}{dt}; f_x = \frac{\partial I}{\partial x}; f_y = \frac{\partial I}{\partial y}; f_t = \frac{\partial I}{\partial t}$, we receive the optical-flow equation as follows:

$$f_x u + f_y v + f_t = 0 \tag{6}$$

It can be recognized that f_x and f_y are the first gradients of the processed image, and f_t is the gradient over time, but u and v are unknown. To solve this issue, our proposed system uses Lucas-Kanade algorithm [9]. This method takes a 3x3 window around the corner to be coupled. Hence, there will be 9 points that have the same motion (according to assumption (2) of the optical-flow algorithm). Based on that we can find set of parameters $\left(f_x, f_y, f_t\right)$ for these 9 points. The problem is how to find two unknown parameters u and v when there are 9 equations. This problem is solved by least square fitting to bring the result as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix} \tag{7}$$

However, since camera is moving, in many cases, there are some corners that do not have their respective locations in the next frame. If 3D distortions in videos are very small. Based on a 2D affine transformation [12], from the local motion vectors obtained from the last stage, global motion parameters including scale s, rotation θ and translation in the directions T_x, T_y are estimated. Supposing (x, y) and (x', y') are the locations of corresponding corners in consecutive frames. The relationship between these two positions can be expressed as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = s.\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix} \quad (8)$$

Here, s is considered equal to 1 because interval between consecutive frames is only several tens of milliseconds. This leads to the change of scale parameters between two frames also is very small. The transformation in (8) thus is simplified as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix} \quad (9)$$

Finally, we obtain three parameters of global motion $\left(\theta, T_x, T_y\right)$.

3.2. Motion compensation

For this stage, like the common existing model, our goal also is to find intended movement parameters. Image frames thus is moved based on that information to remove vibrations. To estimate intended motion parameters, in our system, Kalman filter [7] is exploited to predict the controllable motion of device camera. The motion model shown in equation (9) is applied to the predicted parameters $\left(\theta^{predict}, T_x^{predict}, T_y^{predict}\right)$ as follows:

$$\begin{bmatrix} x^{predict} \\ y^{predict} \end{bmatrix} = \begin{bmatrix} \cos\theta^{predict} & -\sin\theta^{predict} \\ \sin\theta^{predict} & \cos\theta^{predict} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x^{predict} \\ T_y^{predict} \end{bmatrix} \quad (10)$$

Through proposed mechanisms presented above, we can obtain the predicted location $\left(x^{predict}, y^{predict}\right)$ for each pixel as well as shifts of each pixel to its corresponding predicted location.

3.3. Image composition

After motion compensation, there are some pixels, which are moved out from their original frame and some others are shifted to no pixel places because in those locations are empty. Therefore, before making the final output video, our system must resolve the problem, which is called image composition task. In this study, the first scenario is used for our system. The reason is that filling in the empty areas is not only waste time, but also it is not meaningful in the terms of data processing and performance improvement.

3.4. Modelling System

Our stabilization system is come into being based on the selected algorithms and proposed mechanisms introduced in previous Section. Figure 1 shows data flow of the system with used methods. As mentioned before, the most important feature is that during processing data, this proposed system does not use any future frame to stabilize video. This makes causal characteristic for the system and allows real-time processing capability.

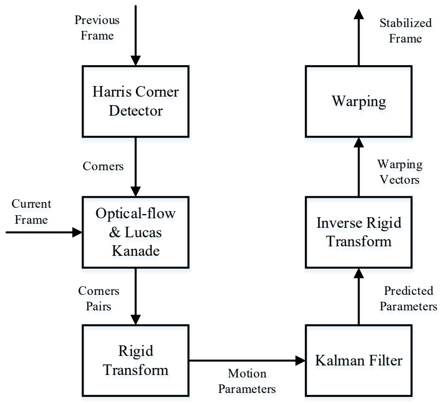

Fig. 1. Stabilization system operation modeling

Because Harris corner detection calculates many complex operations, hence it takes up a lot of processing time. To reduce the number of Harris usages in detecting corner points in a frame, we propose a novel mechanism based on remark as follows. Outcome of corner matching technique using Optical-flow and Lucas Kanade is the corresponding position of this corner in the successive frame and these points are considered approximately as corners in the new frame. Consequently, achieved matching results in the last frame can be used as input to detect corner in the new frame. However, this proposed mechanism will be caused a small decrease of obtained corner quality and amount. After a few processed frames, those corners need to be rediscovered by the Harris detection algorithm to avoid the deterioration. In our proposed system, there are three parameters, which are adjusted to ensure quality as well as performance when applying Harris detection. The first parameter is the number of image areas divided from a frame. These areas will be processed simultaneously to increase system performance. The second parameter is the distance between two successive iteration of corner detection. The last parameter is the minimum number of corner points required in a frame.

4. Experiments and evaluations

This section describes our experiments to evaluate the improvement solutions proposed in Section 3. All tests are run on a computer equipped with Intel Core i5-3210M 2.50GHz processor and 4GB of RAM. Test videos (*city.mp4, road.avi*, and *mountain.mp4*) have resolution of 640x480 pixels. These videos and their stabilized versions can be found in the following link[†]. Our program is written by C++ and inherited some basic modules from OpenCV libraries. Gained data and charts are handled on MATLAB. We carry out three tests as follows: (1) *Speeding up corner detection process*: aims at evaluating effect of partitions dividing mechanism, which allows increase performance for the corner detection step. (2) *Using matching results for corner detection process:* in this test, firstly, we compare the predicted movement trajectories with actual trajectories to demonstrate effectiveness of our system in stabilizing video with mechanism of reusing detected corners. Next, we assess error rate when using the matches results instead of Harris corner detector to show the feasibility of our proposal in resolving the video stabilization problem. (3) *System performance evaluation:* we focus on testing system with real videos to show its performance under the following aspects: the abilities of high resolution processing, achieved good speed and runnable in real-time. In addition, we evaluate processing performance between our system and existing causal systems presented in Section 2 to prove effect of our proposed system in comparison with existing methods.

4.1. Speeding up corner detection process

To take full advantage of the hardware capacities, we divide frames into smaller image areas and then process these areas simultaneously. However, the question is how many partitions generated from a frame is appropriate? This is also the first parameter that is mentioned in Section 3. With hardware configuration of our computer, we carry out this experiment to find the appropriate number of partitions to increase system performance but still ensure the output video accuracy. In this way, we use the *city.mp4* video for this test and compare the error in two partitioned cases of 4 and 8 to non-partitioned. The results are shown by Fig. 2. It can be made an important observation from the achieved outcomes. The intermittent line represents predicted trajectory when frames are divided into 4 partitions. In this case, the difference among obtained results and standard trajectory (solid line) is about 30 pixels. Meanwhile, for case of 8 partitions (dotted line), the achieved trajectory is very large, approximately 100 pixels. Through the experiment, we pick 4 for the partition number parameter in a frame for the tested video. Thus, our system ensures that occurred error during processing is still acceptable.

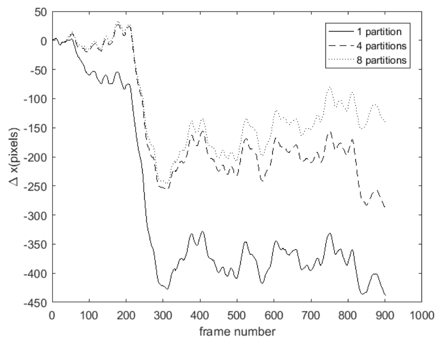

Fig. 2. Predicted trajectory with different partitions

We continue to evaluate system performance during corner detection process with *city.mp4* video. The first test (called A) performs to detect corners in turn in overall frame, and the maximum detected corner number is 60000 points. The other test (called B) removes areas, where are near borders. In this way, the second one detects at most 100 corners, which are divided equally in 4 partitions. We measure the processing speed of these systems in five times and obtain outcomes as shown in Table 1. These results show that system's processing speed has increased 1.2 times after we apply the improvements.

Table 1. The comparison of processing speed

Test	1st	2nd	3rd	4th	5th	Average
A	26581	26250	26800	26027	26533	26438
B	21726	21764	21734	21770	21739	21747

4.2. Using Matching Result for Corner Detection Process

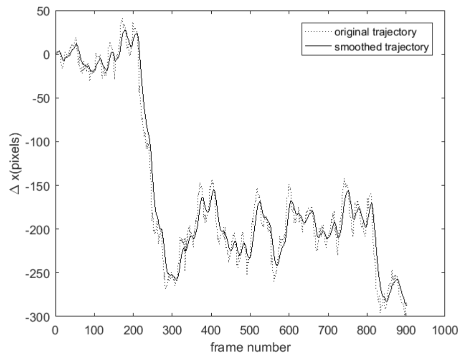

Fig. 3. Original and predicted trajectories comparison

[†] https://goo.gl/K3DNmM

Using corner matching mechanism instead of corner detection with Harris' algorithm will make the obtained corners be not the best corners anymore. This leads to a reduction in the accuracy of the motion estimation. Therefore, to ensure the error rate of the stabilization is still within the acceptable limit, it is necessary to re-detect corners after a few certain frames. In this test, we also use *city.mp4* video. Figure 3 shows actual and predicted movement trajectories when applying the selected algorithms in succession. We find out that undulating motions in the original trajectory are almost removed in the predicted trajectory. This proves the effectiveness of our system in stabilizing video.

Table 2. Processing speed with different iteration numbers

Iteration number	1st	2nd	3rd	4th	5th	Average
1	28590	28540	28642	28672	28706	28630
10	19819	19656	19507	19548	19484	19603

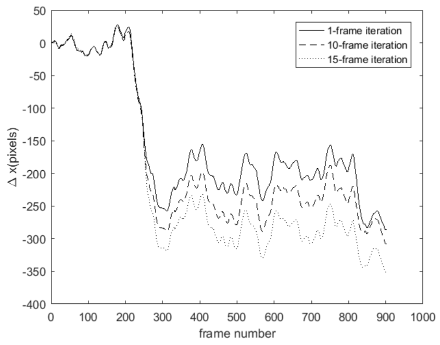

Fig. 4. Diverse iteration comparison for corner detection in predicted trajectories process

Next, we evaluate the system error rate in case of using the matching results instead of Harris corner detector. Specifically, we compare predicted trajectories when corner detection function is used after every 10 and 15 frames. The obtained outcomes are illustrated by Fig. 4. It can be remarked that with threshold parameter of 10, the error is acceptable, only about 50 pixels. While with the threshold parameter of 15, the error is very high, about 100 pixels. Based on the test results, with *city.mp4* video, the iteration of 10 is considered as an appropriate value to keep the difference from the standard trajectory being small enough. With the iteration selected by 10, we compare the effectiveness of this proposal and normal system (i.e. iteration number is 1). The achieved outcomes are shown in Table 2. Through the table, it can easily to see that our improvement mechanism significantly improves average processing speed as compared with normal system. The increase is about 1.5 times with our test computer.

4.3. System performance evaluation

To test system performance, we use all three test videos mentioned above, then run our stabilization system 10 times with each video. The recorded outcomes are presented in Table 3, which shows that our proposed system operates well with 640x480 pixel resolution videos. In addition, achieved processing speed is more than 33 fps and the most important thing is that with the proposed prediction motion estimation algorithm, our system can run in real-time. All the features demonstrate that our system can meet the real-time requirements described in Section 1.

Table 3. System performance evaluation

Input videos / Testing	Video 1 (city.mp4)	Video 2 (road.avi)	Video 3 (mountain.mp4)
Frame number	*903*	*654*	*434*
Processing speed average (ms)	25469	19552	12912
Frame rate (fps)	~ 35	~ 33	~ 33

We continue to compare performance between our system and the existing causal systems, including Shen [4] and Wang [6] introduced in Section 2. To compare with Shen's system [4], we create a virtual machine (VM) on our computer (Intel Core i5-3210M) with single core of 2.5GHz speed and 1GB of RAM. We consider that the created VM has similar hardware configuration with Shen's tests. Using the VM, we perform our stabilization system with all three videos to examine the gained performance and compare with Shen's works. The obtained outcomes are extracted and presented in Table 4. It can be made an important remark as follows. With the same hardware computer configuration and a higher resolution of input videos, our system can process real-time with good effect (up to 27 fps). Meantime, Shen's system has slow stabilizing speed (less than 10 fps).

Table 4. Performance comparison with existing methods

Input videos / Testing	Video 1 (city.mp4)	Video 2 (road.avi)	Video 3 (mountain.mp4)
Frame number	*903*	*654*	*434*
Processing speed average (ms)	33597	26662	17912
Frame rate (fps)	~ 27	~ 25	~ 24

5. Conclusion and future work

This paper presented our proposed system in the manner of improving the speed of motion estimation and the speed of video stabilization system in general. The main improvements involve: (1) combining Harris with Optical-flow and Lucas-Kanade algorithms for the motion estimation step and using Kalman filter to predict the controllable motion. (2) proposing mechanisms for using Harris method in the manner of ensuring the processing quality and system performance. (3) video stabilization system can run in real-time based on causality particularity, which is gained because subsequent frames do not need to use for the stabilization target. The developed system can work with good performance as compared with other existing models in general current computer hardware configuration. In the future, we will go further with goal of providing the process ability for video higher resolution and migrating the model into devices with low hardware configuration.

Acknowledgment

The work is supported by MOET's projects No. B2017-BKA-32 "Research on Developing Software Framework to Integrate IoT Gateways for Fog Computing Deployed on Multi-Cloud Environment"'.

References

[1] Vermeulen, Eddy, Real-time video stabilization for moving platforms, 21st Bristol UAV Systems Conference, 2007.

[2] Shen, Guturu, et al., Video stabilization using principal component analysis and scale invariant feature transform in particle filter framework, IEEE Transactions on Consumer Electronics 55.3 (2009) 1714-1721.

[3] Lim, Anli, et al., Real-time optical flow-based video stabilization for unmanned aerial vehicles, Journal of Real-Time Image Processing (2017) 1-11.

[4] Shen, Pan, et al., Fast video stabilization algorithm for UAV. Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on. Vol. 4. IEEE, 2009.

[5] Vazquez Marynel, and Carolina Chang, Real-time video smoothing for small RC helicopters. Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on. IEEE, 2009.

[6] Wang, Yue, et al., Real-Time Video Stabilization for Unmanned Aerial Vehicles. MVA. 2011.

[7] Kalman, Rudolph E., and Richard S. Bucy, New results in linear filtering and prediction theory. Journal of basic engineering 83.1 (1961) 95-108.

[8] Lucas, Bruce D., and Takeo Kanade. An iterative image registration technique with an application to stereo vision. (1981) 674-679.

[9] Pan, et al., A dual pass video stabilization system using iterative motion estimation and adaptive motion smoothing. Pattern Recognition (ICPR), 2010 20th International Conference on. IEEE, 2010.

[10] Harris, Chris, and Mike Stephens. A combined corner and edge detector. Alvey vision conference. Vol. 15. No. 50. 1988.

[11] Fleet, David, and Yair Weiss. Optical flow estimation. Handbook of mathematical models in computer vision. Springer US, 2006. 237-257.

[12] Holden, Mark. A review of geometric transformations for non-rigid body registration. IEEE transactions on medical imaging 27.1 (2008) 111-128.