A. V. Kornilova, I. A. Kirilenko, N. I. Zabelina, Real-time digital video stabilization using MEMS-sensors, *Proceedings of ISP RAS*, 2017, Volume 29, Issue 4, 73–86

DOI: https://doi.org/10.15514/ISPRAS-2017-29(4)-5

Real-time digital video stabilization using MEMS-sensors

A.V. Kornilova <kornilova.anastasiia@gmail.com>
I.A. Kirilenko <y.kirilenko@spbu.ru>
N.I. Zabelina <zabelina.nattaly@gmail.com>
Saint Petersburg State University, Software Engineering
28 Universitetskiy prospect, Petergof, Sankt-Peterburg, 198504, Russia

Abstract. This article describes our ongoing research on real-time digital video stabilization using MEMS-sensors. The authors propose to use the described method for stabilizing the video that is transmitted to the mobile robot operator who controls the vehicle remotely, as well as increasing the precision of video-based navigation for subminiature autonomous models. The article describes the general mathematical models needed to implement the video stabilization module based on the MEMS sensors readings. These models includes the camera motion model, frame transformation model and rolling-shutter model. The existing approaches to stabilization using sensors data were analyzed and considered from the point of view of the application in a real-time mode. This article considers the main problems that came up during the experiments that were not resolved in the previous research papers. Such problems include: calibration of the camera and sensors, synchronization of the camera and sensors, increasing the accuracy of determining the camera position from sensors data. The authors offer possible solutions to these problems that would help improve quality of the work of existing algorithms, such as a system for parallel synchronized recording of video and sensor data based on the Android operating system. As the main result, the authors represent a framework for implementing video stabilization algorithms based on MEMS sensors readings.

Keywords: video stabilization; MEMS sensors; real-time system; digital signal processing; computer vision; rolling shutter.

DOI: 10.15514/ISPRAS-2017-29(4)-5

Для цитирования: Kornilova A.V., Kirilenko I.A., Zabelina N.I. Real-time digital video stabilization using MEMS-sensors. *Trudy ISP RAN/Proc. ISP RAS*, vol. 29, issue 4, 2017, pp. 73-86. DOI: 10.15514/ISPRAS-2017-29(4)-5

1. Introduction

Modern cameras' matrices allow to take high-quality pictures that are comparable to professional photographs. However, the quality of video that they are able to record

leaves much to be desired and lately it has grown into a problem that needs to be resolved. If modern devices could improve quality of video recording in real time it would not only enable owners of smartphones and action cameras to stream more beautiful and visually appealing video, but would also solve more significant problems. For instance, in case of remotely controlled mobile robots and drones (quadcopters) that perform area monitoring, the low quality of video drastically decreases the precision of control and also leads to greater fatigue of the vehicle operator.

In most cases, you need to get rid of camera shake to solve the problem of poor video quality. It can be achieved either by fixing camera in one place (alternatively, by cancelling out its movement using specially designed mechanisms) or by transforming the frames digitally in such a way so that the video becomes jitterless.

If you choose the first option, you will need special external devices, such as SteadyCam, GyroStick, gimbal (for drones), or specially designed lenses and matrices similar to those available in professional cameras. This approach is not only extremely costly, but also not always applicable. For example, it is impossible to install an external stabilizer on smaller flying vehicles.

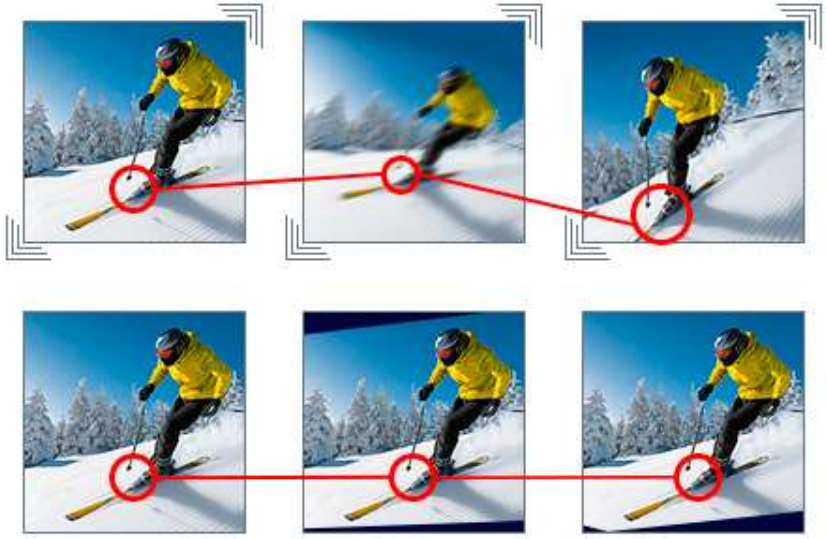


Fig. 1. Image transformation for trajectory smooth

If you opt for the second way, or digital stabilization, you will face the challenge of camera motion estimation and image warping (Fig. 1). Video editing software developers have already advanced significantly in this area. Products like Adobe Premiere, Deshaker, Movavi are all already able to stabilize videos digitally. Similar functionality is also available on YouTube that uses the algorithm proposed in the

work [1]. The main disadvantage of these algorithms [2], [3], [4], [5] is the amount of calculations needed to determine the camera motion. This makes this method inapplicable for real-time video stabilization. Besides that, these algorithms only use the data available in the images themselves, which makes them unreliable in case the shot has poor lighting or features large moving objects.

Alternatively, you can estimate the camera motion during the recording by using the information from MEMS (MicroElectroMechanical Systems) motion sensors, including angular rate sensors (gyroscope), accelerometer and magnetometer. This method requires less processing power to determine camera positioning and, consequently, is more energy-efficient, which makes it suitable for real-time video stabilization. For instance, a common gyroscope consumes only 2-5 mW of power. At the same time, the CPU consumes several hundreds of milliwatts while analyzing frames.

This approach is applied more and more in recent years, as MEMS sensors are becoming widespread on different platforms, especially on smartphones. For instance, Google Pixel, introduced in October 2016, completely lacks mechanical stabilization and uses only gyroscope-based stabilization algorithm. IPhone 7 also uses MEMS sensors for video stabilization but employs camera lenses and matrices for this purpose at the same time.

Mobile applications that offer similar functionality are just now coming up on the market and they are only able to perform video stabilization during post-processing. Some of the most prominent ones are: Instagram Hyperlapse, Microsoft Hyperlapse. Gallus is especially noteworthy, because, unlike others, it utilizes data from MEMS sensors.

This article considers different methods of real-time digital video stabilization that utilize MEMS sensors. Given that this research area is located at the junction of computer vision and digital signal processing, a lot of additional tasks arise, that are worth researching both separately and altogether. The main difficulties, when it comes to creating an application that allows to stabilize videos in real time, are the synchronization of frames and sensor data and the creation of a lightweight stabilization algorithm.

Authors review different existing algorithms and approaches as well as describe the problems that surfaced when these methods were implemented. During this research, we have encountered the following challenges: synchronization of frames and sensor reading, efficient frame transformation and increasing the accuracy of camera positioning. This article solves the found problems and offers more stable and universal implementation of the described algorithm.

In the second section of the article, we review the existing approaches to digital video stabilization that utilize MEMS-sensors, analyze whether these algorithms are suitable for use in real time and also list the mathematical models. In the third section, we describe the methods that improve positioning accuracy by using filters and combining readings from different sensors. In the fourth section, we analyze how to efficiently transform frames during camera rotation. In the fifth section, we consider

the problem of synchronizing frames and sensor readings and use Android OS as an example. There we also review existing methods of automatic camera and sensor parameters calibration. In the sixth section, we list the main results of the ongoing research.

2. Video stabilization

Video stabilization process can be divided into 3 independent stages:

- estimating camera motion using MEMS sensors;
- calculating the desired camera motion in accordance to some logic (for instance, trajectory smoothing);
- transforming the frame to match camera motion to the desired one.

In order to perform video stabilization in real time, we need to find a solution to each of the above listed tasks that would be satisfactory in terms of quality and performance.

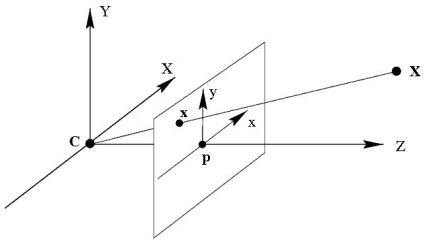
The second stage is the most crucial. When smoothing trajectory, it's important to not only consider jitter as noise, but also to take into account that camera needs to move similarly to the way eye moves naturally. In the beginning of this section, we list the mathematical models and terms that are used and describe the existing algorithms. Then we analyze their advantages and disadvantages, and also propose various improvements.

Authors pay special attention to the two remaining stages, that can be improved significantly, yet still were not touched on in previous papers.

In this section, we suppose that all camera and sensor parameters are known, as well as that sensor readings and camera shots are synchronized in time. The abovementioned problems will be thoroughly discussed in the section dedicated to the parametrization of the stabilization system.

2.1 Mathematical models

Let's take a look at how frame is transformed when camera is rotated. We'll assume that x is the coordinates of a point on a projective plane, and X is the coordinates of a point in space (Fig. 2). Also, for each particular camera, let's assume that it has the matrix K with the following parameters: (o_x, o_y) is the optical center of the camera and f is its focal length. We'll get the following formulas for the projective transformation [6]:

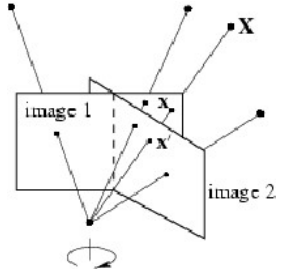


$$x = KX$$

$$K^{-1} = \begin{pmatrix} 1 & 0 & -o_x \\ 0 & 1 & -o_y \\ 0 & 0 & f \end{pmatrix}$$

Fig. 2. Projective transformation

Let's fix the global coordinate system and assume, that in moment *t* the camera is rotated against it, using the rotation matrix *R(t)* (Fig. 3). Then projective transformation will look this way:



$$x = KR(t)X$$

Fig. 3. Location of a point in frames during camera rotation

Let's assume, that x_i и x_j are both projections of the same point *X* in space, but they are located in frames *i* и j respectively, meaning:

$$x_i = KR(t_i)X$$

$$x_j = KR(t_j)X$$

By transforming these expressions, we establish the following connection between projections of the same point in different moments of time:

$$x_j = KR(t_j)R^T(t_i)K^{-1}x_i$$

Thus, let's define the matrix of image transformation between moments in time t_1 и t_2 as:

$$W(t_1, t_2) = KR(t_1)R^T(t_2)K^{-1}$$

$$x_j = W(t_j, t_i)x_i$$

We want to include an additional parameter to the above-described mathematical model of camera and its rotations. It's defined by the camera shutter and solves the problem of blurring when recording fast moving objects. Rolling shutter is a visual distortion that happens, because when the shutter is released, each row of the frame is shot at a different moment in time (Fig. 4-5).

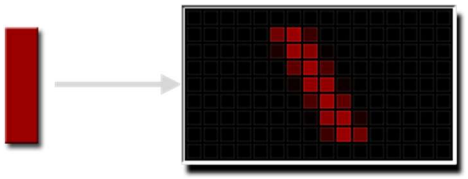


Fig. 4-5. Object movement and Rolling-shutter effect during capturing the moving object

When shutter scans the scene vertically, the moment in time at which each point of frame is shot, is directly dependent on the row it is located in. Thus, if we assume that *i* is the number of the frame and *y* is the row of that frame, then the moment at which it was shot can be calculated this way:

$$t(i, y) = t_i + t_s \frac{y}{h}$$

where t_i is the moment when frame number i was shot, t_i is the time it takes to shot a single frame, *h* is the height of the frame. This can be used to make the general model more precise, when calculating the image transformation matrix.

2.2 Stabilization algorithms

Among the solutions discussed in the scientific society, two are especially worth noting, and we will describe them in this section.

2.3 Algorithm with Gaussian filter

Algorithm described in the article [7] in 2011, is based on Gaussian filter. Camera positioning is calculated by integrating the readings of a MEMS gyroscope for each frame. Then the sequence of camera movements is smoothed by utilizing the Gaussian filter (Fig. 6), and the frames are sequenced using the new motion model. Gaussian filter can be customized by changing the window size (how many discrete points it effects) and the size of the core (how strong the smoothing is). By altering these parameters one can either get rid of local jitter or significant movements.

The use of Gaussian filter is very effective during post-processing, but is not always applicable for real-time stabilization. During post-processing movement can be analyzed completely from start to finish, which allows to increase the size of the window of the filter and smooth the movement stronger. During real-time stabilization, processing buffer needs to include 10-15 frames, which results in a significant delay of 0,3-0,5 seconds.

The source code of the prototype was presented in Matlab, but the article states that algorithm was tested on an iPhone 4. During open realization, the algorithm features narrowed camera rotation parameters. Namely, only horizontal camera rotation is taken into account, which does not always reflect the movement of a shaking camera.

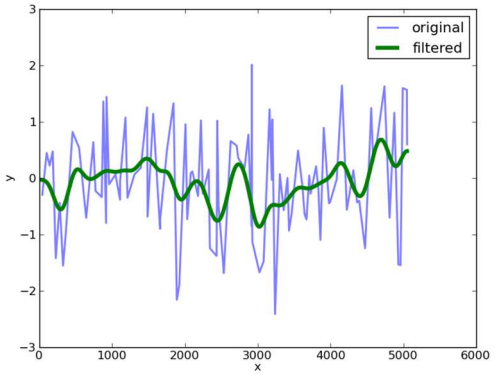


Fig. 6. Trajectory smoothing using the Gaussian filter

2.4 Algorithm utilizing nonlinear filter

Algorithm described in the article [8] in 2014 utilizes a more complex nonlinear filter to smooth camera movement.

In the offered method, the definition of a virtual camera is given. Two concentric zones are selected on the frame – the inner region and the outer region (Fig. 7). Then the rectangle zone is selected in the inner region. Positioning of a virtual camera is determined by the position of this rectangle.

For each new frame, a new position of the abovementioned rectangle is calculated. If it lies within the inner zone, the camera orientation remains the same. If any part of the rectangle lies outside the inner region then the virtual camera's angular velocity is updated by using spherical linear interpolation to bring it closer the physical camera's velocity. Authors note that this algorithm works rather well, but when rectangle hits the edge of the inner zone sudden changes can be expected.

The article offers a way how to make this method suitable for real time video stabilization. If a buffer has k frames, than the camera is supposed to move during these frames with the same velocity it did before. If the rectangle crosses the inner zone, then the spherical interpolation is used to bring the virtual camera velocity closer to the velocity of the physical camera.

Besides significantly decreasing the buffer size, this method has one more advantage. It does not take into account the absolute positioning of the camera, as it only uses the velocity of the camera. Therefore, due to the absence of integration, the error is not accumulated.

Sadly, the authors of the article did not offer a repository with source code of the program, realizing this algorithm. Therefore, it was impossible to repeat the experiment at the time. We plan to realize this approach in the nearest future.

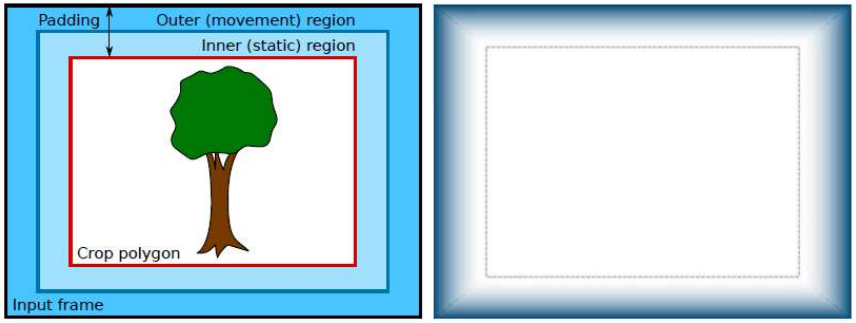


Fig. 7. Inner and outer stabilization zones

3. Determining the positioning

When we were constructing the above-described model, it was assumed that the sensor readings are continuous and accurate. In reality, however, as in all physical devices, MEMS sensors have noise. If the algorithm requires integrating the gyroscope readings, the error caused by the noise will only increase. To solve this problem we will combine the readings of two or more different MEMS sensors, for instance gyroscope and accelerometer. This will allow to eliminate significant errors. The following filters offer similar functionality:

- Complementary filter;
- Madgwick filter [9] – filter that utilizes the gradient descent and allows the use of magnetometer;
- Mahony filter [10];
- Extended Kalman filter – the most successful realization is presented in the work [11].

It is important to mention that the processing complexity of the offered algorithms needs to be minimized for real-time video stabilization. The algorithms are listed in the increasing order of complexity. It is worth noting, that the use of quaternions for estimating positioning and integrating is significantly less complex than other positioning methods like Euler angles or rotation matrices [12].

4. Frame transfomation

After it was determined how much the frame positioning should change, projective transformation should be performed. Realization of the OpenCV library offers this functionality via warpTransoform and perspectiveTransform functions. The first

option performs projective transformation for the whole image, while the second one allows to determine the position of particular points on the frame after transformation.

Using the second function allows us to realize the following algorithm. We choose several points on the frame, a 10x10 grid, for instance. After that a projective transformation is performed for each point, and their new positions are calculated (Fig. 8). The values in the other spots are calculated using interpolation.

By varying the size of the grid, it is possible to find the balance between quality of the image after the rotation and speed of processing of the new frame. While experimenting with 1920x1080 frames, it was determined that the best results are achieved with 10x10 grids.

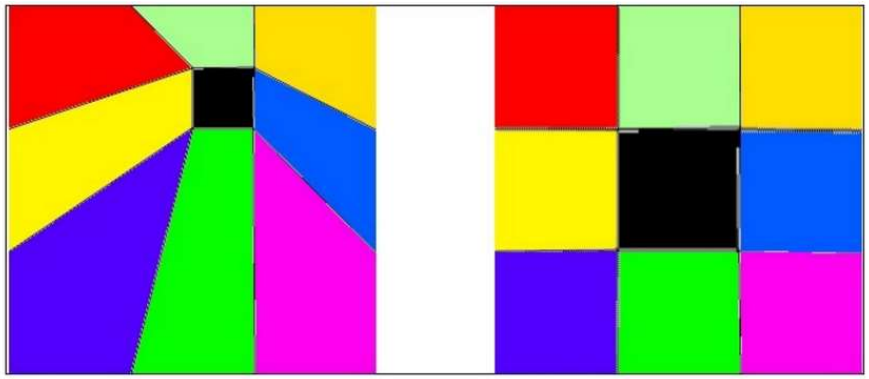


Fig. 8. Image warping

5. Camera calibrations and synchronization

Camera model and the stabilization algorithms, described above, are based on certain assumptions that are not always true in reality. First, it is assumed that sensor readings are a continuous function and are synchronized with frames. Second, we assume that all the necessary parameters for the mathematical model, such as: optical center, focal length and shutter release time are known.

In this section we describe these issues in more detail and offer different solution to the problems.

5.1 Calibrating the unknown camera and sensor parameters

In order for stabilization algorithms to work correctly, we need to have detailed information about camera's and MEMS sensors' parameters. Namely, the optical center, focal length, location of the MEMS sensor coordinate axes in relation to the camera's coordinate axes and shutter release time (rolling shutter). Assuming all pixels are square, we'll set the optical center at *(0, 0)*.

In case of all sensors, a gyroscope in particular, the main unknown parameter is the *bias* – almost constant skew of angular velocities against the exact measurements.

Smartphones sensors are calibrated automatically, while in case of some embedded systems, you need to monitor this parameter closely, as the bias can result in error during integration. To determine the bias, we need to find the mean deviation of angular velocities against the null, when the camera is stable.

The calibration and synchronization problems are solved in the article [13], where the process of online calibration using the extended Kalman filter is described in full detail. Also, in the article [14] the minimization method including determining of the cost function is offered to calibrate the parameters listed above. The full review of camera parameters calibration methods is available in the article [15].

Currently, authors select camera parameters manually for the models used to test algorithms. Automatic calibrations will be realized only after successful experiments with the algorithms.

5.2 Synchronization of a camera and sensors

First, it is important to understand that MEMS sensor readings are discrete. Therefore, even if you know the exact time each frame was taken, it would be impossible to determine the current positioning of the camera. However, since signal's frequency of the MEMS sensor is between 100 and 200 Hz and the frame rate is 30 fps, we can use simple interpolation to get a relatively accurate estimation.

Unlike embedded systems, that offer hardware synchronization of frames and MEMS sensor reading, operating systems of smartphones sometimes do not offer this functionality. Authors encountered this problem on Android when prototyping the application for simultaneous recording of video and data from sensors.

It turned out, that the main API of the camera, available on each phone does not provide the event scheme for processing single frames. Therefore it was impossible to use software to determine the place of each frame in the time series of sensor readings (Fig. 9). The possible solution to this problem is using the mathematical methods to match two time series with different degrees of discretization: frequent – sensor reading and rare – video frames. The use of displacement of features as metric is suggested.

Starting with level 21 Android API, a new API for Camera2 was introduced. It features the event driven programming that would allow to determine the taking of a frame by using the event handler OnImageAvailableListener. Even if this improvement can't be used to determine the exact timestamp of a frame, it will help to estimate the place of the frame on the time series of sensor readings. Therefore, this approximation can be used for realizing the mathematical method for matching series.

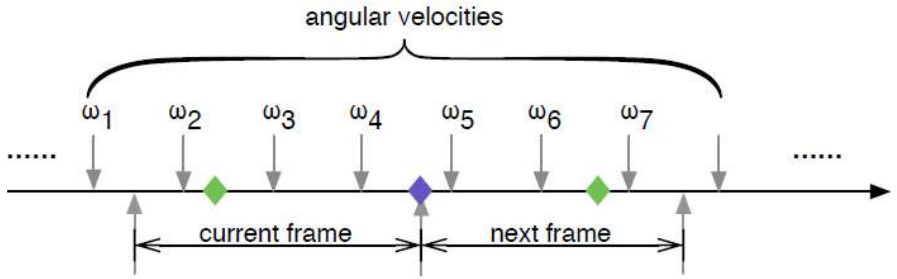


Fig. 9. Matching the time series of frames and gyroscope

6. Current results

Currently, authors have implemented the prototype of the algorithm utilizing the Gaussian filter on Python, that cover the model of 3-dimensional camera rotation. Provided the synchronized sensor readings and frames, as well as intrinsic camera parameters, this algorithm shows great results during post-processing.

Synchronization of sensor readings and camera is performed by an application, described in the corresponding section. Based on this, we plan to execute this algorithm in real-time mode in the nearest future. To decrease latency we will use the optimal filters, that are described in the section dedicated to them, as well as piece-by-piece frame transformation.

To make the software video stabilization module cross-platform, we plan to test the suggested methods of real-time calibration of intrinsic camera parameters and implement them.

7. Conclusion

At this moment, there are many different approaches to digital video stabilization, but not all of them require too much processing power to be used in real time use. Methods utilizing MEMS sensors are worth noting as they allow to save processing resources. Scientific community offers several stabilization algorithms utilizing these sensors. They show great results during post-processing and several prototypes for real-time processing are available.

Despite the possibilities and the need for real-time digital stabilization, its implementation is hard from a technical standpoint, because the video sensor and MEMS sensors need to be coordinated. Besides that, a lot of work still needs to be done to optimize these algorithms for work in real-time.

Many additional challenges and problems described by the authors, show that there is a lot of room for improvement in existing solutions, namely in the way algorithms work. All algorithms that we studied employ a quite primitive mathematical model, which makes it viable to continue research in this area using more advanced mathematics.

Authors set their next goal as using the work they have already done to build a full-fledged software module for real-time digital video stabilization and increase its ability to function on different platforms.

8. Acknowledgment

Funding for this work was provided by JetBrains Research.

References

[1]. Grundmann M., Kwatra V. and Essa I. Auto-directed video stabilization with robust L1 optimal camera paths, CVPR 2011, Providence, RI, 2011, pp. 225-232. DOI: 10.1109/CVPR.2011.5995525.

[2]. Y. Matsushita, E. Ofek, Weina Ge, Xiaoou Tang and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 7, pp. 1150-1163, July 2006.

[3]. Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin and Aseem Agarwala. Subspace Video Stabilization. ACM Transactions on Graphics (presented at SIGGRAPH 2011). Vol. 30, Issue 1, 2011: 4:1-4:10.

[4]. Y. S. Wang, F. Liu, P. S. Hsu and T. Y. Lee. Spatially and Temporally Optimized Video Stabilization. IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 8, pp. 1354-1361,Aug .2013. DOI: 10.1109/TVCG.2013.11.

[5]. S. Liu, L. Yuan, P. Tan, and J. Sun. Bundled camera paths for video stabilization. ACM Transactions on Graphics (TOG) - SIGGRAPH 2013 Conference Proceedings. Volume 32 Issue 4, July 2013 . Article No. 78. DOI: 10.1145/2461912.2461995.

[6]. R. Szeliski. Computer Vision: Algorithms and Applications. 2010.

[7]. A. Karpenko. Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes. Stanford Tech Report CTSR 2011-03. Stanford University.

[8]. Bell, S., Troccoli, A. J. & Pulli, K. (2014). A Non-Linear Filter for Gyroscope-Based Video Stabilization.. In D. J. Fleet, T. Pajdla, B. Schiele & T. Tuytelaars (eds.), ECCV (4) (p./pp. 294-308), : Springer. ISBN: 978-3-319-10592-5.

[9]. S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Report x-io and University of Bristol (UK), 2010.

[10]. R. Mahony, T. Hamel, J. M. Pflimlin. Complementary filter design on the special orthogonal group SO(3). Proceedings of the 44th IEEE Conference on Decision and Control, 2005, pp. 1477-1484. DOI: 10.1109/CDC.2005.1582367.

[11]. Rong Zhu, Dong Sun, Zhaoying Zhou, Dingqu Wang, A linear fusion algorithm for attitude determination using low cost MEMS-based sensors, Measurement, Volume 40, Issue 3, 2007, Pages 322-328, ISSN 0263-2241.

[12]. J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors, 2006.

[13]. C. Jia and B. L. Evans. Online Camera-Gyroscope Autocalibration for Cell Phones. IEEE Transactions on Image Processing, vol. 23, no. 12, pp. 5070-5081, Dec. 2014. DOI: 10.1109/TIP.2014.2360120.

[14]. H. Ovrén and P. E. Forssén. Gyroscope-based video stabilisation with auto-calibration. 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 2090-2097. DOI: 10.1109/ICRA.2015.7139474.

[15]. Wang Qi, Fu Li and Liu Zhenzhong. Review on camera calibration. 2010 Chinese Control and Decision Conference, Xuzhou, 2010, pp. 3354-3358. DOI: 10.1109/CCDC.2010.5498574.

Стабилизация видеоизображения в режиме реального времени с использованием MEMS-датчиков

А.В. Корнилова <kornilova.anastasiia@gmail.com>
Я.А. Кириленко <y.kirilenko@spbu.ru>
Н.И. Забелина <zabelina.nattaly@gmail.com>
Санкт-Петербургский государственный университет,
Кафедра системного программирования
198504, Россия, г. Санкт-Петербург, г. Петергоф, Университетский проспект, 28

Аннотация. Данная статья описывает текущие исследования по цифровой стабилизации видеоизображения в режиме реального времени с использованием MEMS датчиков. Авторы предполагают использование данного метода для стабилизации видеоизображения, передаваемого оператору дистанционно управляемых мобильных роботов, в частности, для улучшения качества управления малыми летательными аппаратами и снижения усталости оператора. В статье вводятся основные математические модели и понятия необходимые для реализации программного модуля цифровой стабилизации с использованием показаний MEMS датчиков. К таким моделям необходимо отнести: модель вращения камеры, модель трансформации кадра и модель rolling shutter эффекта. Также в статье рассматриваются существующие подходы к стабилизации видеоизображения с использованием MEMS датчиков и дается оценка их применимости в режиме реального времени. Кроме того, освещаются проблемы, возникающие при воспроизведении результатов предыдущих работ и неразрешенные в данных статьях. К таким проблемам следует отнести: синхронизацию показаний датчиков и кадров, калибровку камеры и датчиков, повышение точности определения вращения камеры, эффективную трансформацию кадра при повороте. Авторы предлагают возможные решения данных проблем, в частности, одним из результатов является система параллельной синхронизированной записи кадров и показаний датчиков движения — гироскопа и акселерометра — на базе операционной системы Android. В качестве основного результата представляется фреймворк для тестирования алгоритмов по стабилизации видеоизображения, а также реализация алгоритма стабилизации с использованием фильтра Гаусса для сглаживания траектории движения камеры в рамках данного фреймворка.

Keywords: стабилизация видео; MEMS-датчики; системы реального времени; цифровая обработка сигналов; компьютерное зрение; rolling shutter

DOI: 10.15514/ISPRAS-2017-29(4)-5

Для цитирования: Корнилова А.В., Кириленко Я.А., Забелина Н.И. Стабилизация видеоизображения в режиме реального времени с использованием MEMS-датчиков. *Труды ИСП РАН*, том 29, вып. 4, 2017 г., стр. 73-86. DOI: 10.15514/ISPRAS-2017-29(4)-5

Список литературы

[1]. Grundmann M., Kwatra V. and Essa I. Auto-directed video stabilization with robust L1 optimal camera paths, *CVPR 2011*, Providence, RI, 2011, pp. 225-232. DOI: 10.1109/CVPR.2011.5995525.

[2]. Y. Matsushita, E. Ofek, Weina Ge, Xiaoou Tang and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 7, pp. 1150-1163, July 2006.

[3]. Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin and Aseem Agarwala. Subspace Video Stabilization. ACM Transactions on Graphics (presented at SIGGRAPH 2011). Vol. 30, Issue 1, 2011: 4:1-4:10.

[4]. Y. S. Wang, F. Liu, P. S. Hsu and T. Y. Lee. Spatially and Temporally Optimized Video Stabilization. IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 8, pp. 1354-1361, Aug .2013. DOI: 10.1109/TVCG.2013.11.

[5]. S. Liu, L. Yuan, P. Tan, and J. Sun. Bundled camera paths for video stabilization. ACM Transactions on Graphics (TOG) - SIGGRAPH 2013 Conference Proceedings. Volume 32 Issue 4, July 2013. Article No. 78. DOI: 10.1145/2461912.2461995.

[6]. R. Szeliski. Computer Vision: Algorithms and Applications. 2010.

[7]. A. Karpenko. Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes. Stanford Tech Report CTSR 2011-03. Stanford University.

[8]. Bell, S., Troccoli, A. J. & Pulli, K. (2014). A Non-Linear Filter for Gyroscope-Based Video Stabilization.. In D. J. Fleet, T. Pajdla, B. Schiele & T. Tuytelaars (eds.), *ECCV (4)* (p./pp. 294-308), : Springer. ISBN: 978-3-319-10592-5.

[9]. S. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Report x-io and University of Bristol (UK), 2010.

[10]. R. Mahony, T. Hamel, J. M. Pflimlin. Complementary filter design on the special orthogonal group SO(3). Proceedings of the 44th IEEE Conference on Decision and Control, 2005, pp. 1477-1484. DOI: 10.1109/CDC.2005.1582367.

[11]. Rong Zhu, Dong Sun, Zhaoying Zhou, Dingqu Wang, A linear fusion algorithm for attitude determination using low cost MEMS-based sensors, Measurement, Volume 40, Issue 3, 2007, Pages 322-328, ISSN 0263-2241.

[12]. J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors, 2006.

[13]. C. Jia and B. L. Evans. Online Camera-Gyroscope Autocalibration for Cell Phones. IEEE Transactions on Image Processing, vol. 23, no. 12, pp. 5070-5081, Dec. 2014. DOI: 10.1109/TIP.2014.2360120.

[14]. H. Ovrén and P. E. Forssén. Gyroscope-based video stabilisation with auto-calibration. 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 2090-2097. DOI: 10.1109/ICRA.2015.7139474.

[15]. Wang Qi, Fu Li and Liu Zhenzhong. Review on camera calibration. 2010 Chinese Control and Decision Conference, Xuzhou, 2010, pp. 3354-3358. DOI: 10.1109/CCDC.2010.5498574.