Indo-Iranian Journal of Scientific Research (IIJSR)
Volume 2, Issue 1, Pages 26-35, January-March 2018

Real Time of Video Stabilization Using Field-Programmable Gate Array (FPGA)

Mrs.S.Kokila[1], Mrs.M.Karthiga[2] and V. Monisha[3]

[1]*Assistant Professor, Department of Electronics and Communication Engineering, Vivekanandha College of Engineering for Women, Tiruchengode, India.*
[2]*Assistant Professor, Department of Electronics and Communication Engineering, Vivekanandha College of Engineering for Women, Tiruchengode, India.*
[3]*PG Scholar, Department of Electronics and Communication Engineering, Vivekanandha College of Engineering for Women, Tiruchengode, India.*

Article Received: 28 November 2017 *Article Accepted: 31 January 2018* *Article Published: 05 February 2018*

ABSTRACT

Digital video stabilization is an important video enhancement technology which aims to remove unwanted camera vibrations from video sequences. Trading off between stabilization performance and real-time hardware implementation feasibility, this paper presents a feature-based full-frame video stabilization method and a novel complete fully pipelined architectural design to implement it on Field-Programmable Gate Array (FPGA). Using pipelining and parallel processing strategies, the whole process has been designed using a novel complete fully pipelined architecture and implemented on Altera's Cyclone III FPGA to build a real-time stabilization system. Experimental results have shown that the proposed system can deal with standard PAL video input including arbitrate translation and rotation and can produce full-frame stabilized output providing a better viewing experience at 22.37 milliseconds per frame, thus achieving real-time processing performance.

Keywords: Digital video stabilization, Field-Programmable Gate Array and Pipelined architectural design.

1. INTRODUCTION

This paper offers a good trade-off between algorithm performance and ease of hardware implementation and presents a feature-based full-frame video stabilization method implemented on FPGA using a novel complete fully pipelined architecture for real-time processing. First, feature points are extracted with the ORB algorithm and matched between adjacent frames according to the Hamming distance, followed by robust interframe motion estimation using the affine transformation model with an outlier rejection scheme based on the Random Sample Consensus (RANSAC) algorithm.

By cascading the estimated interframe motion, the cumulative motion parameters between the current and reference frames can be computed, which are then used to construct a mosaicked image using an image mosaicking technique to avoid unfilled areas in the output frame caused by motion compensation. Then the translational components of the cumulative motion parameters are smoothed with a Kalman filter representing intentional camera movement, which is taken to be the position of a display window with the desired frame size on the mosaicked image, to obtain a full motion-compensated frame Feature-based video stabilization can be very powerful in most applications. However, its high-performance processing requirements pose a difficult challenge for real-time operation.

1) To the best of our knowledge, the proposed framework in this paper makes the first attempt to implement the whole feature-based video stabilization process on a single FPGA chip, achieving real-time performance. 2) A novel complete fully pipelined FPGA architecture has been first proposed to substantially accelerate feature-based video stabilization in a highly parallel manner, which also provides a reference to accelerating other feature-based video processing tasks such as object tracking and video fusion on FPGA. 3) Using the proposed framework, a real-time miniaturized video stabilization system with low power consumption can be built, which is particularly

favorable to portable applications. Simplification Considering the power, space, and real-time performance constraints of embedded systems, the proposed process has been designed in a novel complete fully pipelined architecture using parallel processing techniques and implemented on Altera's Cyclone III FPGA. The results obtained show that the FPGA-based video stabilization system is superior in stabilizing shaky and rotated videos with real-time processing capability. The proposed system in this paper differs from the above ones in several respects. 1) The feature-based method has been used to estimate the affine parameters between consecutive frames, which can deal with not only translational but also rotational and scaling jitter in the video2) The proposed system can handle large-scale shake over an unlimited search area. 3) Using the image mosaicking technique, the proposed system can produce full-frame stabilized output, offering a better viewing experience. 4) Benefiting from the proposed novel complete fully pipelined FPGA architecture, the whole stabilization process has been substantially accelerated in a highly parallel manner, thus achieving real-time performance.

Commonly used methods for feature detection and description such as Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) have gained wide consensus because of their good performance. However, they are difficult to implement on embedded systems for real-time applications because of their high computational complexity. Oriented FAST and Rotated BRIEF (ORB), first presented by Rublee *et al*. in 2011, is an efficient local feature descriptor that is rotation-invariant and resistant to noise. It was designed using the Features from Accelerated Segment Test (FAST) keypoint detector and the Binary Robust Independent Elementary Features (BRIEF) descriptor leading to a remarkable reduction in computational complexity and hardware cost. Moreover, it is very suitable for Field-Programmable Gate Array (FPGA) implementation because it consists mostly of basic comparison operations and has parallel computation capability.

2. RELATED WORK

In recent years, numerous video stabilization techniques have been developed. These methods generally produce good stabilization results, but few of them can achieve real-time performance for practical video stabilization in portable applications due to their high computational complexity. FPGAs provide an effective solution to accelerating real-time video processing because of the parallel computing capability. However, the implementation of complex algorithms on FPGA has always been a difficult challenge. Therefore, a trade-off must be made between algorithm performance and ease of hardware implementation.

So far, several approaches have been presented to implement video stabilization on FPGA. Araneda *et al*. used gray-scale projection to estimate the translational motion between consecutive frames and produced low-resolution output as a result of motion compensation. Yabuki *et al*. divided the frame into several image blocks and obtained the global motion vector by tracking each image block through template matching. Thus the correction range was limited .The proposed system in this paper differs from the above ones in several respects. 1) The feature-based method has been used to estimate the affine parameters between consecutive frames, which can deal with not only translational but also rotational and scaling jitter in the video. 2) The proposed system can handle large-scale shake

over an unlimited search area. 3) Using the image mosaicking technique, the proposed system can produce full-frame stabilized output, offering a better viewing experience. 4) Benefiting from the proposed novel complete fully pipelined FPGA architecture, the whole stabilization process has been substantially accelerated in a highly parallel manner, thus achieving real-time performance.

3. PROPOSED STABILIZATION ALGORITHM

The proposed stabilization algorithm consists of five major procedures: feature detection and description, feature matching, motion estimation, image mosaicking, and motion filtering.

A. Feature Detection and Description

First of all, the corners of each frame are extracted using the ORB algorithm, which is an efficient feature descriptor composed of a FAST corner detector and an oriented BRIEF descriptor.

1) FAST Detector: The FAST detector determines whether a pixel is a corner by comparing its intensity value with those of surrounding pixels in a circular ring. For a given intensity threshold *t*, pixels around the center point can have one of three states:

$$\begin{cases} I_i \le I_p - t & \text{(Darker)} \\ I_p - t < I_i < I_p + t & \text{(Similar)} \\ I_i \ge I_p + t & \text{(Brighter)} \end{cases}, \qquad (1)$$

Where *Ip* and *Ii* denote the intensity value of the center point and of a surrounding pixel, respectively. Pixel *p* is classified as a corner if *N* contiguous pixels in the circular ring are simultaneously darker or simultaneously brighter than the center point by threshold *t*, as illustrated in Fig. 1

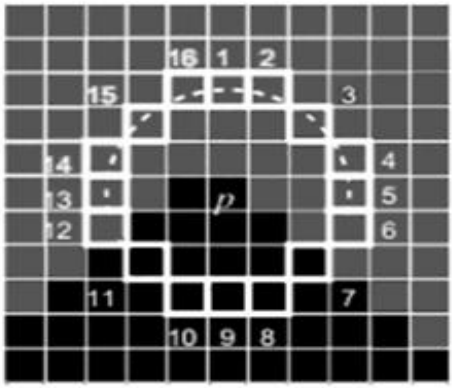

Fig. 1. Pixel allocation around the center point.

N was chosen to be 9, which has been proven to give good performance.

2) Orientation: The orientation of the detected corner is determined by Rosin's corner intensity. The moment of a patch is defined as:

$$C \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right).$$

The corner orientation is the angle of the vector from the patch center to the centroid:

$$T \text{ atan2}(m01 , m10) .$$

In the proposed method, the patch is taken as a circular region of radius 3 to improve the rotation invariance of this measure. The orientation is quantized to increments of 2π/32 (11.25 degrees), and a lookup table of orientation values is created based on the numerical relationship between $m01$ and $m10$ to facilitate implementation on embedded systems.

3) Feature Descriptor: The ORB algorithm generates a rotationally invariant feature description by submitting the corner orientation to the BRIEF algorithm. The BRIEF algorithm constructs a bit string description of an image patch through a set of binary intensity tests.

The ORB algorithm chooses 256 test point pairs with a Gaussian distribution in a 31 x 31 pixel patch around the corner, where each test point is a 5 x 5 patch for better noise immunity. For any feature set of n binary intensity tests at location (xi, yi), the locations of the test point pairs can be expressed as a 2 x n matrix.

By multiplying these locations by the rotation matrix $R\theta$, the test point pairs can be rotated around the patch center according to the corner orientation. Then the rotated version $S\theta$ of S is used for binary testing, producing a rotationally invariant description.

In the proposed method, the locations of the 256 test point pairs, as well as their rotated versions corresponding to each orientation from 1 to 32, are computed previously as a lookup table for ease of implementation on embedded systems.

B. Feature Matching

The feature matching step finds matched point pairs between adjacent frames based on the similarity of the detected corners, which can be measured using the Hamming distance. For each detected corner in the current frame, the candidate match with the minimum distance in the previous frame can be found. Meanwhile, the match with the second-minimum distance is also recorded. Then the ratio between the minimum and the second-minimum distances is checked against a preset threshold.

The proposed method contains another strategy for further removal of potential wrong matches. Because small translations and rotations occur between adjacent frames, correct matches are supposed to have similar locations and orientations. Based on this, the coordinate difference and the orientation difference of the matches obtained in the previous step can be checked to remove potential wrong matches. Suppose that the coordinates of the matches are (X, Y) and (X', Y') and the orientations are R and R'. The coordinate difference of the matches can then be checked.

C. Motion Estimation

Once the matched point pairs have been obtained, the motion estimation step is performed to estimate interframe motion. To describe the geometric transformation between two consecutive frames, a six-parameter 2D affine model was used in this research, which trades off model stability against representation capacity. The affine model can describe translation, rotation, scaling, and skew of images, which is enough for most video stabilization applications. Three point pairs are necessary to compute the affine parameters. Selection of point pairs that are incorrectly matched or related to moving objects in the scene, here called *outliers*, may lead to improper motion estimation. Because three points form a triangle, the three point pairs selected for affine parameter estimation may form two triangles in adjacent frames [23], as shown in Fig. 2.

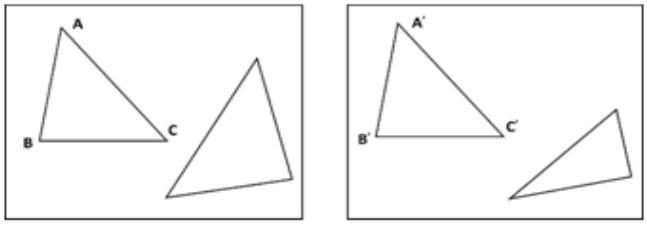

Fig. 2. Triangles formed by three selected point pairs.

Suppose that ABC and A'B'C' are the triangles formed in the previous frame and the current frame, respectively. The area difference between the two triangles can be checked to reject outliers for affine parameter estimation based on the fact that three correctly matched point pairs in the scene background have similar triangle areas in adjacent frames.

The proposed method includes a robust motion estimation approach based on the triangle area matching technique and the RANSAC algorithm. The procedure is as follows:

(1) Randomly select three point pairs from those produced in the feature matching step.

(2) If the selected point pairs are not collinear and the area difference between the two triangles formed in the adjacent frames is less than a given threshold, go to else go back.

(3) Compute the affine parameters using the selected point pairs.

(4) Calculate the geometric distance error for each point pair produced in the feature matching step using the computed affine model, and check the number of inliers for which the geometric distance error is less than agiven threshold.

Repeat (1)-(4) for a number of trials and choose the affine model with the most inliers as the result.

Furthermore, to improve the robustness of the approach, a numerical threshold test of the final estimated affine parameters is performed to reject obvious estimation errors.

D. Full-Frame Stabilized Output

To produce a full-frame stabilized video sequence, the proposed method uses the image mosaicking technique and a Kalman filter.

1) Image Mosaicking: In the previous steps, motion parameters between consecutive frames have been estimated. By cascading these interframe motion parameters, the cumulative motion parameters between the current frame and the selected reference frame can be calculated. Assuming that the first frame is the reference frame and the *N*th frame is the current frame, the cumulative motion parameters between them can be computed as:

$$P_{N\_1} = P_{2\_1} P_{3\_2} P_{N\_N-1},$$

Where P_{N_N-1} denotes the estimated interframe parameters between frames *N* and *N*-1.

Simple motion compensation by transforming the current frame using the computed cumulative motion parameters may result in unfilled areas in the output frame. To reconstruct the unfilled areas for a better viewing experience, the image mosaicking technique is used to align the current frame onto the reference frame by means of the cumulative motion parameters. Then a display window with the desired frame size is set onto the mosaicked image to obtain a full motion-compensated output, as illustrated in Fig. 3.

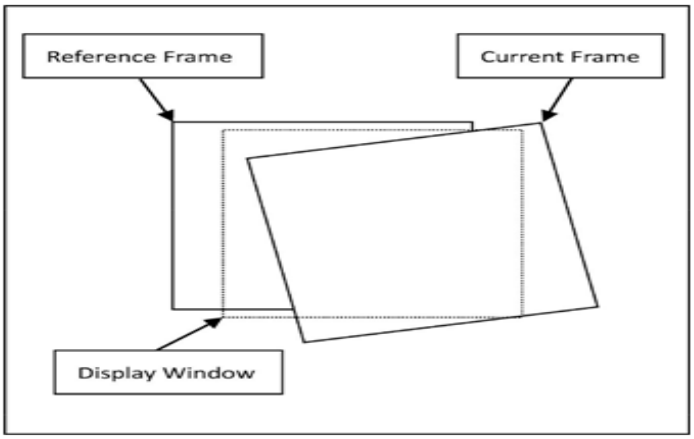

Fig. 3. Output frame based on mosaicked image.

2) Motion Filtering: Camera movement includes intentional camera movement and random jitter, which can be regarded as low-frequency signal and high-frequency noise, respectively. Generally, intentional camera movement contains only translational motion. To remove undesired noise due to jitter while preserving intentional camera movement, a low-pass filter can be used.

In the proposed method, the translational components of the cumulative motion parameters against frame number, referred to as ($X(n)$, $Y(n)$), which can be considered as the original absolute position of the display window on the mosaicked image, are low-pass filtered by the Kalman filter to remove undesired noise due to jitter. The filtered result with intentional camera movement only can be used as the final absolute position of the display window to produce a full-frame stabilized video sequence.

4. EXPERIMENTAL RESULTS

A) Experimental Setup

The proposed method was implemented on Altera's EP3C120F780 FPGA which is an ideal low-power video processing platform with large quantities of multipliers and M9K memory blocks and thus offers a good solution to achieving the miniaturized system with low power consumption. External DDR2 SDRAM and PAL video decoder/encoder chips were also necessary to build the whole system. Thus a real hardware platform was realized.

The Peak Signal-to-Noise Ratio (PSNR) was used to evaluate the effectiveness of the stabilization method. The PSNR between consecutive frames is defined as:

$$PSNR(I_1, I_0) = 10 \log \frac{I_{max}^2}{MSE(I_1, I_0)}$$

$$ITF = \frac{1}{N_{frame} - 1} \sum_{k=1}^{N_{frame}-1} PSNR(k),$$

Experimental results and discussion

For fair stabilization performance comparison with other methods, the proposed algorithm was first evaluated on the VS against 10 publically available video sequences covering different types of scenes. For illustration purpose, the results for two representative challenging sequences including fast large-scale shake and moving objects, respectively, are reported. Both sequences are publically available online at. Some frames of them are shown in Fig 4.

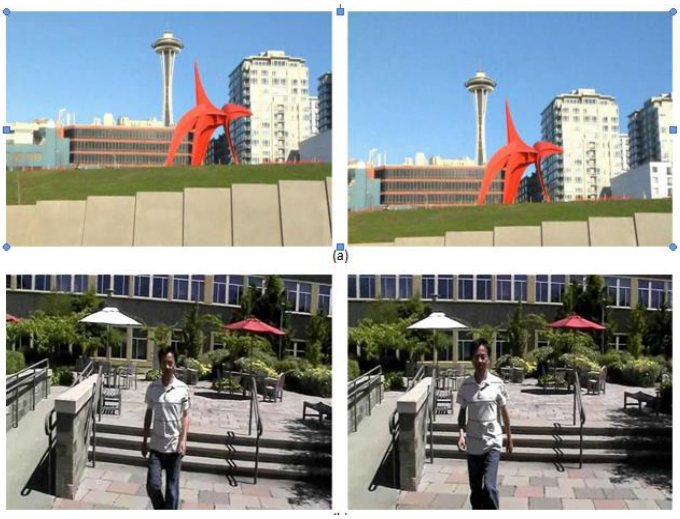

Fig. 4.Frames of test video sequences

The stabilization performance of the proposed method was compared to that of a traditional video stabilization program, Deshaker 3.1 on the above video sequences which were scaled to 640 □ 480 resolution. 200 frames of each sequence were analyzed gives the comparisons between the original PSNR and the stabilized PSNR resulted by the proposed algorithm and Deshaker. Correspondingly, the ITF values for the original and stabilized video sequences are also given. As shown in Table I, substantial improvements in ITF were obtained, confirming the effectiveness of the proposed algorithm under challenging conditions. Moreover, the proposed method outperformed Deshaker, with ITF increments of approximately 45-50% compared to 27-35%.

TABLE I
ITF OF ORIGINAL AND STABILIZED TEST VIDEO SEQUENCES

Video sequence	Original (dB)	Deshaker (dB)	Proposed (dB)
a	19.58	24.85	29.66
b	15.15	20.48	22.13

To evaluate the real-time performance of the proposed system, standard PAL video with 720 □ 576 resolution at a frame rate of 25 fps was used as input. A time counter located in the FPGA code was used to measure the computation time of one frame. For comparison, the computation time of the VS implementation using the default compilation options on a 2.93 GHz Intel Core i3 Quad was also measured. The results are shown in Table II.

TABLE II
COMPARISON OF COMPUTATION TIMES

Frame Duration	40.00 ms
Computation Time on VS	116.74 ms
Computation Time on FPGA	22.37 ms

The average computation time per frame on the FPGA at a clock frequency of 80MHz was 22.37 milliseconds compared to 116.74 milliseconds on the VS, making the proposed implementation more than four times faster, which confirms that the proposed fully pipelined FPGA architecture can substantially accelerate the feature-based stabilization method. Moreover, the results show that the system can complete the operation on one frame within a frame duration of 40.00 milliseconds, which meets the requirements of real-time processing.

In addition, the real-time performance of the proposed system was also compared to that of the intensity-based systems presented by and as illustrated in Table III. The results show that although the proposed system uses the feature-based method with much stronger stabilization capability and higher computational complexity, it can perform as well as or better than the intensity-based systems in terms of real-time performance, which also confirms the effectiveness of the proposed fully pipelined FPGA implementation.

TABLE III
COMPARISON OF REAL-TIME PERFORMANCE

Reference	Resolution (pixels)	Frame Rate (fps)
Yabuki *et al.* [20]	542×496	30
Li [21]	720×576	25
Proposed	720×576	25

Finally, the proposed framework can be feasible for high-definition videos mainly through the following modifications. 1) Adjust the video decoder/encoder modules according to the encoding format of the high-definition video to be processed. 2) Resize the frame buffers distributed in the external DDR2 SDRAM according to the frame size of the video. 3) Resize the line FIFOs in the modules that have read or write access to the DDR2 SDRAM according to the line size of the video frame. Moreover, additional FPGA timing optimization can also be considered for real-time high-definition video processing.

5. CONCLUSION

In this paper, a feature-based stabilization method and its FPGA implementation using a designed novel complete fully pipelined hardware architecture have been presented. Using an ORB feature descriptor and robust RANSAC-based motion estimation, the method is applicable to videos with arbitrate translational and rotational jitter and has a certain robustness to moving objects in the foreground. In addition, the image mosaicking technique in conjunction with a Kalman filter produces full-frame stabilized output, yielding results which provide a better viewing experience. Experiments on real video data have been conducted to verify the effectiveness of the proposed system. The observed results demonstrate that the proposed fully pipelined FPGA architecture substantially accelerates the feature-based stabilization method in a highly parallel manner and that the proposed system is attractive for practical video stabilization with real-time requirements. Future work will be devoted to implementing the proposed framework on high-definition videos. Moreover, because the system is based on an

FPGA and a few peripherals it can achieve miniaturization with low power consumption, and therefore applications on moving platforms will be considered.

REFERENCES

[1] Battiato. S, Bruna. A. R, and Puglisi. G, "A robust block-based image/video registration approach for mobile imaging devices," *IEEE Trans. Multimedia*, vol. 12, no. 7, pp. 622-635, 2010.

[2] Kim .S.-K,Kang. S.-J,Wang.T.-S, and Ko.S.-J, "Feature point classification based global motion estimation for video stabilization," *IEEE Trans. Consum. Electron.*, vol. 59, no. 1, pp. 267-272, 2013.

[3] Liang. C.-K, Peng. Y.-C, Chang. H.-A, Su. C.-C, and Chen. H, "The effect of digital image stabilization on coding performance [video coding]," in *Proc. ISIMP*, 2004, pp. 402-405.

[4] Morimoto. C, and Chellappa. R, "Evaluation of image stabilization algorithms," in *Proc. ICASSP*, 1998, pp. 2789-2792.

[5] Puglisi. G, and Battiato. S, "A robust image alignment algorithm for video stabilization purposes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 10, pp. 1390-1400, 2011.

[6] Song. C, Zhao. H, Jing. W, and Zhu. H, "Robust video stabilization based on particle filtering with weighted feature points," *IEEE Trans. Consum. Electron.*, vol. 58, no. 2, pp. 570-577, 2012.

[7] Yang. J, Schonfeld. D, and Mohamed. M, "Robust video stabilization based on particle filter tracking of projected camera motion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 945-954, 2009.

[8] Zhou. J, Hu. H, and Wan. D, "Video stabilization and completion using two cameras," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 12, pp. 1879-1889, 2011.