



# Article An Unsupervised Video Stabilization Algorithm Based on Key Point Detection

Yue Luan, Chunyan Han \* and Bingran Wang

School of Software, Northeastern University (NEU), Shenyang 110169, China \* Correspondence: hancy@swc.neu.edu.cn

Abstract: In recent years, video stabilization has improved significantly in simple scenes, but is not as effective as it could be in complex scenes. In this study, we built an unsupervised video stabilization model. In order to improve the accurate distribution of key points in the full frame, a DNN-based key-point detector was introduced to generate rich key points and optimize the key points and the optical flow in the largest area of the untextured region. Furthermore, for complex scenes with moving foreground targets, we used a foreground and background separation-based approach to obtain unstable motion trajectories, which were then smoothed. For the generated frames, adaptive cropping was conducted to completely remove the black edges while maintaining the maximum detail of the original frame. The results of public benchmark tests showed that this method resulted in less visual distortion than current state-of-the-art video stabilization methods, while retaining greater detail in the original stable frames and completely removing black edges. It also outperformed current stabilization models in terms of both quantitative and operational speed.

Keywords: video stabilization; unsupervised learning; key-point detection; adaptive cropping; RAFT



Citation: Luan, Y.; Han, C.; Wang, B. An Unsupervised Video Stabilization Algorithm Based on Key Point Detection. *Entropy* **2022**, *24*, 1326. https://doi.org/10.3390/e24101326

Academic Editor: Armando J. Pinho

Received: 24 July 2022 Accepted: 16 September 2022 Published: 21 September 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/).

# 1. Introduction

With the growing popularity of mobile devices, more and more people are adopting mobile devices to record their lives, but unprofessional shooting often leads to the introduction of shakiness into the final video, which affects people's subsequent viewing and applications of the video, so more and more people are beginning to pay attention to video stabilization. Especially in complex scenes, the difficulty of video stabilization is increased by the influence of the interactions between the moving foreground and background in the video. It is therefore essential to develop high-performance video stabilization algorithms for shaky video in complex scenes.

The process of a video stabilization algorithm generally starts with the detection of key points, obtaining the unstable motion trajectory according to the key points, then smoothing the motion trajectory, and finally generating stable frames.

Traditional video stabilization methods have the following shortcomings:

- Key-point detection relies on manual methods, such as ORB and SIFT. However, the detection will be inaccurate or fail when there are drastic changes in external conditions, such as light, darkness, large movements, and the video itself is not of high quality, etc. [1–3];
- Traditional methods of key-point tracking usually use optical flow or Kalman filtering to track the key points. This assumes that there is no significant motion between the two frames and that the grayscale between the two frames is unchanged. However, the scenes in real life are ever-changing, and this produces a large mismatch for complex videos [4];
- The traditional method of trajectory smoothing uses a fixed kernel of smoothing. However, because the trajectories are ever-changing, the use of a fixed kernel does not achieve the best stabilization effect [2,5–9]. It can also be achieved by fitting to obtain the smoothed trajectories, but this method involves a high levle of error [10];

• Over-cropping the black edge of the generated frames results in the final frame retaining a smaller field of view and less detail [1,5,7].

DNN-based methods avoid the drawbacks of the traditional algorithm. However, there are still several problems:

- There are few pairs of stable/unstable videos in the existing official datasets, which is not conducive to supervised models [11–13];
- Most of the existing stabilization models still use traditional methods to detect key
  points in the key-point detection stage, but the traditional methods do not detect the
  key points completely and many key points are lost. In the key-point detection stage
  of the video stabilization method using a DNN-based model, although the key points
  generated are abundant, the method does not consider the impact of redundant key
  points in a large untextured area [14];
- For generated frames, a fixed cropping method is used to remove black edges, which preserves less of the original frame content [11–16].

In this paper, an unsupervised video stabilization algorithm based on key-point detection is proposed to deal with the above problems. A DNN-based key-point detector was first used to detect key points. The key points and optical flow inside the untextured area with the largest area were optimized to obtain an accurate unstable motion trajectory. The motion trajectory was then placed into the smoother to adaptively generate a smoothing kernel to smooth the motion trajectory iteratively. Finally, based on the smoothed trajectory, a homography transformation was applied to generate a stable frame with black edges. For the processing of the black edges, in this study we used an adaptive cropping method to crop and retain the maximum area of the original frame.

The four contributions of this paper are as follows:

- An unsupervised deep learning model is used for video stabilization, which does not require pairs of stable/unstable datasets;
- A method is proposed to optimize key points and optical flow in large texture-free areas to obtain accurate non-stationary motion trajectories;
- Adaptive cropping is proposed to retain the maximum field of view for the stable video, while not having a black edge; and
- The model runs faster on the basis of improved model performance, which is more suitable for the requirements of video stabilization.

## 2. Related Work

In this section, we summarize traditional stabilization methods and DNN-based stabilization methods. Traditional stabilization methods include 2D and 3D methods.

**2D methods.** They use 2D transforms to describe the 2D motion trajectory in the video and stabilize the video based on these transforms. For example, MeshFlow [1] uses a traditional method for key-point detection, then uses Kalman filtering [17] to track key points and obtains motion trajectories. The motion trajectories are smoothed by optimizing the minimum energy function and stable frames are generated by means of grid-based warping. Liu et al. optimized camera trajectories by means of bundling to achieve motion smoothing, and generated stable frames based on the grid [5]. Wang et al. developed a stabilizer based on finding the relative position of key points [6]. The developers of the Steadyflow [7] method used Jacobi's method to smooth the motion trajectories. Matsushita Y et al. achieved video stabilization by interpolating and inpainting the video [8]. Grundmann M et al. achieved video stabilization by optimizing the camera path using L1 [9]. Hu R et al. used SIFT for keypoint detection, and these key points were used to estimate the motion trajectories of the camera [3]. The 2D method is relatively simple to implement, but its time complexity is high. However, when it comes to complex scenes, the method's processing capability for stabilizing video is relatively limited. This is because the focus when stabilizing a simple scene is only the jitter factor introduced by the camera, whereas in a complex scene one also needs to consider whether there is a moving foreground target in the video. At the same time, the 2D method only uses a fixed smooth

kernel in the process of generating stable frames, but unstable trajectories are various, so it is incorrect to use a fixed smooth kernel.

**3D methods.** They find the camera trajectory in 3D space and smooth it to achieve video stabilization. Liu and Wang et al. used a depth camera for stabilization [2]. Liu and Gleicher et al. projected the 3D feature points onto a new camera location for stabilization [10]. Karpenko et al. used a gyroscope sensor for stabilization [18]. Zhang et al. stabilized video sequences based on a 3D perspective camera model [19]. Bhat P et al. modified the multi-view stereo algorithm to calculate the view-dependent depth for video stabilization [20]. 3D methods obtain more information and therefore perform better. However, the 3D stabilization algorithm has strict hardware conditions and limited applications.

**DNN-based stabilization methods.** They aim at the transformation of unstable frames to stable frames, whereas traditional stabilization methods aim at the obtainment of motion trajectories. Yu and Ramamoorthi et al. achieved the generation of stable frames by inputting the processed accurate optical flow into the network to generate the warping fields of stable frames [15]. The advantage of this method is that the optical flow is accurate. The disadvantage is that the post-processing of the optical flow leads to complex operation costs and long running time. Furthermore, optical flow over-patching can result in local distortion. Moreover, this method uses a manual key-point detector, which may fail to detect complex video or texture-free region cases. Xu et al. obtained the stable video directly using unsupervised learning models [14]. The advantages of this approach are its short running time, clear model logic, and the lack of a need for training data pairs. However, the disadvantage is that the key points are generated using a depth model, so redundant key points will be generated, especially in regions with relatively large untextured areas. Furthermore, the motion trend inside the large untextured area is different from the overall motion trend. This means that the final motion trajectories obtained are inaccurate, which introduces distortion. Wang et al. obtained the homography matrix of the grid through the use of a Siamese network, which generated stable frames [11]. This method has the advantage of using both unstabilized frames and frames that have been stabilized up to the current time as inputs to obtain the warping field until all stabilized frames are generated, which is performed cyclically, as it does not stabilize only one frame individually. The disadvantages of this method are that, firstly, many training data pairs are required, and secondly, the model only uses the simple homography matrix to generate stable frames, so the method will not work for videos with violent motion and close-up scenes. Choi et al. used interpolation for video stabilization but introduced artifacts [21]. The advantage of this approach is that the full-frame is stabilized, so there are no cropping problems with the final stabilized video obtained. The disadvantage is that the generated frames can introduce artifacts due to the need for multiple interpolations. The PWStableNet method [12] introduces a cascade structure to generate pixel-by-pixel warp fields for video stabilization. The advantage of this method is that it takes into account the fact that different pixels may have different distortions, so it stabilizes in a pixel-by-pixel manner in adjacent frames, and the use of a cascade structure is beneficial to learning the warp field more accurately. A disadvantage of this method is that it uses pixel-by-pixel stabilization, so the running time is relatively long, and it also requires a large number of training data pairs. It is not as effective as the stable grid-based model for complex video processing.

In contrast, the method proposed in this paper uses an unsupervised DNN-based model for video stabilization, so the requirements on the dataset are not strict. For videos of complex scenes, we focus on the problem of obtaining rich key points, while optimizing key points and optical flow over large untextured areas to generate more accurate motion trajectories. In this study, we used SuperPoint [22] to generate rich key points, then used gradient information to optimize key points and optical flow so that the final result would have less distortion. For the problem of black edges in the generated frames, an adaptive cropping method was used to ensure that the black edges were completely removed, while retaining more of the original information.

## 3. Pipeline

The overall structure of the algorithm is shown in Figure 1. The algorithm is divided into four stages: the coarse key-point detection and tracking stage, the accurate key-point detection and accurate optical flow stage, the smooth motion trajectory stage, the stable frame generation and adaptive cropping stage. The above stages are described in Sections 3.1–3.4.



Figure 1. Overall model of the pipeline.

#### 3.1. Coarse KeyPoint Detection and Tracking Stage

This stage is divided into two parts, one involves using SuperPoint to detect key points and return the matched key points, and the other involves using RAFT [23] to generate the optical flow iteratively. The stage is illustrated in Figure 2. This optical flow is then combined with the key points returned by SuperPoint to complete the tracking of the key points.



Figure 2. The process of coarse key point detection and tracking stage.

In the optical flow stage, we refer to the RAFT method [23] developed by Zachary Tee et al., whereas in the key-point detection stage, we refer to the SuperPoint method of Daniel Detone et al. [22].

This stage takes adjacent frames  $I_i$  and  $I_{i+1}$ ,  $i \in \{0, C\}$ , where C is the total number of frames. It uses SuperPoint for key-point detection to obtain  $kp_i$  and  $kp_{i+1}$ , which are matched to obtain the matched pairs  $\{m_i, m_{i+1}\}$ . Then  $I_i$  and  $I_{i+1}$  are input into RAFT to obtain the optical flow  $F_i$ . According to  $F_i$ , the motion vector  $f_i$  of  $m_i$ , at the position corresponding to the key point in  $F_i$ , is extracted.  $m_i$  is added to the motion vector  $f_i$  to obtain the position of the  $m_i$  in relation to the next frame, which completes the tracking of the key points and obtains the motion trajectories of the key points. As shown in Figure 3, the reason for adopting SuperPoint was that SuperPoint is robust compared with manual key-point detection methods. Manual methods may miss many potential points of interest. Another problem is that SuperPoint produces too many key points, as shown in Figure 4. As a result, for videos with large untextured areas, redundant key points will be generated and distortion will be introduced. This problem is solved in the second stage of this paper.



**Figure 3.** Complex video (moving foreground): the left shows the distribution of key points obtained using the traditional manual method and the right shows the distribution of key points obtained using SuperPoint.



**Figure 4.** Video of a large flat area (sky): the left shows the distribution of key points generated using manual methods and the right shows the distribution of key points generated using SuperPoint.

#### 3.2. Accurate Alignment and Motion Trajectory Stage

The second stage mainly involves obtaining accurate key points and optical flow, as well as obtaining the trajectory of the keypoints. This stage is divided into two parts. One is related to obtaining the accurate alignment, and the other is related to obtaining the motion trajectories. The accurate alignment portion of this stage includes the generation of the mask, ensuring the precision of the key points and the precision of the optical flow, as shown in Figure 5.



Figure 5. Overall flowchart of the accurate alignment part of the process.

## 3.2.1. Mask Generation

From experiments, we know that key-point redundancy in large untextured areas and inaccurate optical flow inside large untextured areas may distort the generated stable frames. Thus, in our method, we obtain the mask of the *i*-th frame by means of a gradient, expressed as  $M_i$ :  $M_i^{(x,y)} = 0$ , where  $G_i^{(x,y)} \ge 3\overline{G_i}$ , (x, y) denotes the position of the pixel and  $\overline{G_i}$  denotes the average of the gradient of the *i*-th frame.

The visualization of  $M_i$  according to the gradient, denoted by  $part_1$  in Figure 5, shows that there are many burrs around the untextured area and irregularities, which will affect

the subsequent processing. Therefore, the operation of dilation and erosion on  $M_i$  is used to obtain  $M'_i$ .

Connecting small and non-continuous non-flat areas at small intervals via dilation, Equation (1) is as follows:

dilated 
$$(x, y) = \max\{M((x + x', y + y'))\}$$
 (1)

and rejecting non-flat areas that are too small via erosion, Equation (2) is as follows:

$$\operatorname{eroded}(x, y) = \min\{M((x + x', y + y'))\}$$

$$(2)$$

where the area around (x, y) is denoted by (x + x', y + y')  $(|x'| \le 1, |y'| \le 1)$ .

# 3.2.2. Accurate Key Points

The mask  $M'_i$  can be obtained from the mask generation stage. As shown in *part*<sub>2</sub> of Figure 5, the maximum untextured area in  $M'_i$  is obtained via calculation. We remove the key points within the maximum untextured area to obtain the accurate key points, as shown in Figure 6.



**Figure 6.** Accurate key points: (**a**) shows key-point detection using SuperPoint only and (**b**) shows the distribution of key points after removing the flat areas.

## 3.2.3. Accurate Optical Flow

The unprocessed optical flow is shown in Figure 7a. It can be noted that the optical flow within the untextured area is not consistent with that in the textured area, which will distort the generated frame in the local region. Next, PCAFlow [24] is used to fill in the optical flow in the removed area based on the optical flow in the unremoved area, so that the motion of the frame tends to be consistent and an accurate optical flow is obtained, as shown in Figure 7b.



**Figure 7.** Accurate optical flow: (**a**) shows the optical flow obtained using RAFT for the original frame, (**b**) shows the optical flow obtained by fitting the optical flow to the removed position using PCAFlow.

#### 3.2.4. Trajectory Generation

The final step is to obtain a mesh-based vertex motion trajectory. Due to the various levels of complexity of the video, in this study we used homography based on the separation of foreground and background and mesh-based vertex profiles to obtain the motion trajectories. In this study, the frames were divided into one grid every 16 pixels, with a total of  $A \times B$  grids. According to  $d_i$ , the  $m_i$  value is then propagated to the nearby grid vertices  $v_i$  according to Equation (3):

$$d_i = \sqrt{\left|m_i^2 - v_i^2\right|} \tag{3}$$

where  $m_i$  denotes the position of the key points,  $v_i$  is the position of the mesh vertexes, and "| |" denotes an absolute value.

The next two median filters are used for the motion trajectories obtained for each vertex. The first one aims at assigning each vertex a most representative motion vector and the second one aims at preventing the appearance of noise. The motion vector  $f_i$  is then divided into two categories representing the background and the foreground of motion according to Equation (4):

$$B, P = \operatorname{Kmeans}(f_i, 2). \tag{4}$$

According to the above,  $\{m_i, m_{i+1}\}$  are used in RANSAC to obtain two homographies,  $H_b$  and  $H_p$ . The local residual motion vectors for the foreground and background are obtained according to Equation (5):

$$\widetilde{v}_i^p = m_i^p - H_p \times m_{i+1}^p \\
\widetilde{v}_i^b = m_i^b - H_b \times m_{i+1}^b$$
(5)

where  $\times$  means matrix multiplication.

According to Equation (6), the foreground and background motions are obtained:

$$v^{p} = v_{i} + \tilde{v}_{i}^{p}$$

$$v^{b} = v_{i} + \tilde{v}_{i}^{b}$$

$$v = v^{p} + v^{b}$$
(6)

#### 3.3. Motion Trajectory Smoothing

The unstable trajectory is input into the encoder/decoder structure to generate a smooth kernel for the trajectory, and then the smooth kernel is applied to the iterative smoothing of the unstable trajectories, and finally, the stable trajectories are obtained.

The overall loss function is composed of three parts: temporal smoothness loss  $L_t$ , shape-preserving loss  $L_s$ , and content-preserving loss  $L_c$ :

According to Equation (7),  $L_t$  is formed by minimizing the energy function:

$$L_{t} = \sum_{t=0}^{T-1} (\|P(t) - C(t)\|_{2})^{2} + \lambda \sum_{r \in 16} (\|P(t) - P(r)\|^{2})$$
  
$$C(t) = \sum_{t=0}^{T-1} v(t)$$
(7)

where C(t) denotes the camera path, t denotes time moment, P(t) denotes the optimized path, *r* denotes the radius ( $r \in 16$ ), and v(t) denotes the motion vector of the mesh vertices.

Suppose that  $\{p, \hat{p}\}$  is the matching pair of key points from  $I_i$  to  $I_{i+1}$ , and p consists of a closed grid including four vertices, according to Equation (8):

$$p = V_p \times w_p^{1}$$

$$V_p = [v_{ia}^{1}, v_{ia}^{2}, v_{ia}^{3}, v_{ia}^{4}]$$

$$w_p = \left[w_p^{1}, w_p^{2}, w_p^{3}, w_p^{4}\right]$$
(8)

where  $w_p$  denotes the bilinear interpolation operation, *a* denotes the grid, and *i* denotes the *i*-th frame.

 $\hat{p}$  can then be expressed using  $[\hat{v}_{ia}^1, \hat{v}_{ia}^2, \hat{v}_{ia}^3, \hat{v}_{ia}^4]$ , then the shape-preserving loss  $L_s$  is built on each grid, according to Equation (9):

$$L_{s} = \sum_{i=0}^{C-1} \sum_{a=1}^{(A-1)(B-1)} \left\| \vartheta_{ia}^{3} - \left( \vartheta_{ia}^{2} + R_{90} \left( \vartheta_{ia}^{1} - \vartheta_{ia}^{2} \right) \right) \right\|_{2}^{2}$$
(9)

where  $R_{90} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ .

To prevent distortion around key points, and to avoid distortion between flat and non-flat areas, content-preserving loss  $L_c$  is added here, according to Equation (10):

$$L_{c} = \sum_{x=0}^{H-1} \sum_{y=0}^{W} \left\| p - H_{g}(p) \right\|^{2}$$
(10)

where  $H_g$  is the homography matrix of the grid containing p, and p is found as in Equation (8).

## 3.4. Adaptive Cropping

The fourth stage is the adaptive cropping stage. The stable frames generated in the previous stage have black edges. Here, we present an adaptive cropping method designed to crop the stable frames with black edges, cropping all the black edges while retaining more original frames.

The process of this stage first involves obtaining the binary image of the black edge of each frame according to the generated stable frame with a black edge. Then, the binary image is eroded. Finally, the global maximum inscribed rectangle is obtained according to the maximum inner contour of the black edge, and the video frame is cropped according to the inscribed rectangle. The overall processes are as follows.

**Step1:** Find the black-edge binary image, as shown in Figure 8. A specific binary image with only black edges is first obtained from the generated stable frames with black edges. Using a grid-based format, each grid generates its own homography matrix  $H_g$  based on the motion vectors in the *x* and *y* directions. The original grid vertices generate the target-stabilized grid based on  $H_g$ , which ultimately synthesizes the stable frames. Only the coordinates within 0-*H* and 0-*W* are the RGB values of the original frames. All other locations greater than *W* and *H*, and less than 0, are black edge locations, from which a specific binary image can be calculated.



**Figure 8.** Special binary images for the generated frames: (**a**) shows the generated stable frame with black edges and (**b**) shows the special binary map obtained according to (**a**).

Step2: The "binary image " obtained from *step*<sub>1</sub> is eroded according to Equation (2). Step3: Contouring the white area of the "binary image". This step involves finding the connected domain in the "binary image" with a pixel value of 255, and the same idea is used in the work of Satoshi et al. [25]. In a binary image, when the pixel values between adjacent pixels are different for the first time, this isconsidered to be the start of the boundary *i<sub>s</sub>* and is uniquely encoded to deal with the problem of creating multiple boundaries. When you

find the the start-point  $i_s$ , then you look for the end-point  $i_c$ , which is used to determine whether the contour has ended. When the current point is found to belong to the contour, it is currently the center point  $i_c$ . In the process of tracking, we look for the next point belonging to this contour in the 8-neighbourhood of the center point, which is the latest point,  $i_n$ .  $i_c$  and  $i_n$  are continuously iterated and replaced to complete the generation of the boundary, as shown in Figure 9.



**Figure 9.** Diagram of the contour tracking process: where  $i_s$  denotes the starting point,  $i_e$  the endpoint,  $i_c$  the central point, and  $i_n$  the latest point. (a) shows the distribution of the points in the current state. (b) shows the result of the first trace.

**Step4:** The process of finding the maximum inner rectangle for the contour. The outline of the white region and the set of points of that outline can be obtained from  $step_3$ . Then, according to the characteristics of the rectangle, we define the sum value of the coordinates of the upper left corner of the rectangle as the minimum, and the sum value of the coordinates of the lower right corner as the maximum. Using this rule, we find the coordinates of the two points diagonal to the rectangle using point set of the outline, so we can find the maximum inner rectangle of the outline.

The advantage of adaptive cropping is the ability to avoid the problem of the black edge in generating stable frames and to retain the maximum field of view of the original frames.

## 4. Experiments

We evaluated our model on the public benchmark dataset, the NUS dataset [5], and compared it with the current state-of-the-art models, both DNN-based methods and traditional methods. In this section we provide comprehensive quantitative and qualitative assessment results. Furthermore, during processing, it was found that most models could not handle high-resolution video. The model presented in this paper can handle video of any resolution adaptively.

## 4.1. Datasets

The NUS dataset includes a total of 174 unstable videos of varying duration (10–60 s), which are divided into six categories: Regular, Zooming, Running, QuickRotation, Crowd, and Parallax. The resolutions in the dataset are  $640 \times 360$ ,  $640 \times 480$ , and  $1280 \times 720$ .

## 4.2. Experimental Settings

Since the experimental model in this paper was an unsupervised model, there was no need for a training dataset. We just input the unstable video data, implemented the model, and output stable video. The test set used for the model was obtained from the NUS dataset. We ran the model for this paper on a Tesla V100-SXM2 with a 32G GPU.

#### 4.3. Subjective Results

The visual results for Meshflow [1], DUT [14], StabNet [16], DIFRINT [21], Yu et al. [15], and the model presented in this paper are shown in Figure 10.



**Figure 10.** Visual results of different methods, including Meshflow [1], DUT [14], StabNet [16], DIFRINT [21], Yu et al. [15] and the method in this paper. Red boxes indicate distortion, and green circles indicate the presence of black edges.

The results from the visualization can be decsribed as follows.

**Video1:** Large untextured area (sky in the video). The MeshFlow [1] and Yu et al. [15] methods produced redundant black edges due to improper cropping. DUT [14] and Yu et al. [15] did not take into account the key points in the untextured area and the optical flow's inconsistency with the overall optical flow, resulting in significant distortion of the tower in the stable frame, as well as around the buildings. StabNet [16] excessively cropped the image and at the same time introduced more distortion as this method only handles regular two-dimensional motion.

**Video2:** Motion foreground. MeshFlow [1] produced too few key points, resulting in distortion in and around the moving object, as well as excess black edges. DUT [14] and Yu et al. [15] both produced excessive black edges due to improper cropping. DIFRINT [21] produced artifacts around the moving foreground target. DIFRINT [21] resulted in artifacts due to the use of multiple interpolations. StabNet [16] left only a small field of view due to excessive cropping.

**Video3:** QuickRotation. MeshFlow [1], DUT [14], and Yu et al. [15] all produced redundant black edges due to improper cropping. MeshFlow [1] did not adapt to fast rotation using only homography transforms and out-of-place warping. The DIFRINT [21] method uses interpolation and does not adapt to fast-moving objects—neither the camera itself nor the objects in the video. The motion trajectory of the feature points of this type of video is complex, so StabNet [16] did not adapt well and this led to distortion.

**Video4:** Crowd. MeshFlow [1] and Yu et al. [15] produced excessive black edges due to improper cropping. MeshFlow [1] produced distortion, resulting from inaccurate estimation due to the use of homography transformations alone, whereas the results of the method of Yu et al. [15] showed distortion due to the inaccurate optical flow at the filled

edges. DIFRINT [21] still produced artifacts due to interpolation. The results produced by StabNet [16] were still distorted due to the fact that this method is not able to adapt well to complex motion.

## 4.4. Quantitative Results

## 4.4.1. Evaluation Metrics

To fairly assess the quality of the model, three evaluation metrics are introduced here: the cropping ratio, distortion ratio, and stability score.

The cropping ratio is the area of the generated frame remaining after the black border has been completely removed. The larger the ratio, the better. The calculation is based on the homography matrix **M**, obtained by matching the key points of the original and stable frames. The cropping ratio **cp** of each frame is obtained from the scale component of **M**. The final crop ratio is equal to the average of all **cp** values, according to Equation (11):

$$\mathbf{c}\mathbf{p} = 1/\sqrt{\mathbf{M}[0,1]^2 + \mathbf{M}[0,0]^2}$$
  
cropping\_ratio =  $\frac{1}{C}\sum_{c=1}^{C}(\mathbf{c}\mathbf{p})$  (11)

The **distortion ratio** is the degree of distortion of the stable frame. A higher value means that a better shape of the original frame has been maintained. The distortion rate is equal to the minimum value of the ratio of the two smallest eigenvalues  $\lambda_1$  and  $\lambda_2$  among all the eigenvalues of **M** in all frames, and the distortion fraction is equal to the minimum of all  $\lambda_1 \lambda_2$  ratios, according to Equation (12):

$$\lambda_{1}, \lambda_{2}, \lambda_{3}, \dots, \lambda_{n} = \text{linalg.eig}(\mathbf{M})$$
  

$$\lambda_{1}, \lambda_{2} = \min((\lambda_{1}, \lambda_{2}, \lambda_{3}, \dots, \lambda_{n}), 2)$$
  
Distation\_ratio =  $\min\left(\frac{\lambda_{1}}{\lambda_{2}}\right)$ 
(12)

where linalg.eig() represents the eigenvalue operation.

The stability score is the degree of smoothness of the stabilized video—the larger the better. We select the average of the percentage of translational components and the average of the percentage of rotational components over the frame, the minimum of both is called stability score, according to Equation (13):

stability\_score = min 
$$\left(\frac{1}{C}\sum_{c=1}^{c} fft(\text{ Trans }), \frac{1}{C}\sum_{c=1}^{c} fft(\text{ Theta })\right)$$
 (13)

where Theta denotes the rotational component, Trans denotes the translational component, and *fft*() denotes the Fourier transform function.

#### 4.4.2. Quantitative Results

We evaluated our model and state-of-the-art video stabilization methods on the public benchmark dataset NUS, where the models being compared were Meshflow [1], Liu et al. [5], StabNet [16], and DUT [14], Yu et al. [15]. The results of the evaluation of these quality metrics in the demonstrated approach are shown in Tables 1–3.

Table 1 shows a comparison of the cropping ratio. In the NUS dataset, the method presented in this paper maintained a similar cropping ratio to that obtained using the methods of Liu et al. [5] and Yu et al. [15]. Meanwhile, compared with the methods of Meshflow [1], DUT [14] and StabNet [16] based on the "one size fits all" cropping method, the cropping ratio obtained using the method presented in this paper has already improved upon the cropping ratio of many other methods.

						~
	Meshflow [1]	Liu et al. [5]	StabNet [16]	DUT [14]	Yu et al. [15]	Ours
Regular	0.69	0.82	0.54	0.74	0.88	0.88
Running	0.59	0.79	0.51	0.69	0.70	0.70
Zooming	0.60	0.79	0.52		0.77	0.75
Crowd	0.55	0.82	0.50	0.71	0.82	0.80
QuickRotation	0.38	0.84	0.42	0.67	0.84	0.70
Parallax	0.54	0.83	0.50	0.71	0.82	0.78

Table 1. Comparison of cropping ratio.

Table 2 shows a comparison of the distortion ratio. Our model, using the NUS dataset, resulted in less distortion. Compared with MeshFlow's transform based on global homography [1], the transform based on the foreground and background separation introduced less distortion. Compared with DUT [14], which also uses a DNN-based key-point detector, the problem of key-point redundancy caused by a large untextured area was considered in our experimental process, and the key points in a large untextured area were removed. Furthermore, PCAFlow was used to fit the optical flow, which introduced less distortion. Compared to StabNet [16] which also generates stable frames based on grids, StabNet [16] was not unsupervised and DeepSta datasets were used in the training. This dataset is too small and contains a single scene, so the training was not accurate enough. Compared with the method of Yu et al. [15], which also filled the flat area, Yu et al. [15] filled the boundary between the untextured area and the textured area for a second time, which was overfilled and also produced deformation and distortion. When the object in the video was too close to the camera or had a large occlusion or there were too many people in the video, the motion trajectory optimization method of Liu et al. [5] did not follow its motion trend, resulting in poor results, including distortion and artifacts.

Table 2. Comparison of distortion ratio.

	Meshflow [1]	Liu et al. [5]	StabNet [16]	DUT [14]	Yu et al. [15]	Ours
Regular	0.90	0.92	0.70	0.98	0.97	0.98
Running	0.76	0.83	0.75	0.84	0.90	0.92
Zooming	0.80	0.74	0.72		0.90	0.92
Crowd	0.76	0.84	0.76	0.86	0.92	0.92
QuickRotation	0.76	0.76	0.57	0.82	0.82	0.92
Parallax	0.72	0.84	0.57	0.85	0.91	0.93

Table 3 shows the stability score obtained in our experiments. In the NUS dataset, our model obtained a high stability score compared to those of the other models. Our model was the best on the whole, especially in Running, Zooming, QuickRotation, and Parallax categories, and in Regular and Crowd the method was ranked second. Our model performed better than Meshflow [1], StabNet [16], and DUT [14], which are also based on the use of a grid. Compared with DUT [14], which also adopts unsupervised learning, the stability score of our model exceeded that of DUT [14] on the NUS datasets. Compared with the method of Liu et al. [5], in regard to smoothing the path of the bundled camera, our model showed superior performance. Compared to the model of Yu et al. [15], which uses optical flow as its input, our model showed superior performance in the Running, Zooming, QuickRotation, and Parallax categories. However, in the Crowd and Regular categories, our model performed slightly worse than that of Yu et al. [15], because, for some videos, the flat area was complicated. For example, when the flat area in the video was very fragmented, our model did not deal with such videos well.

	Meshflow [1]	Liu et al. [5]	StabNet [16]	DUT [14]	Yu et al. [15]	Ours	
Regular	0.84	0.81	0.84	0.84	0.89	0.87	
Running	0.84	0.77	0.82	0.84	0.86	0.88	
Zooming	0.74	0.85	0.81		0.88	0.89	
Crowd	0.77	0.82	0.74	0.79	0.88	0.85	
QuickRotation	0.80	0.86	0.79	0.88	0.89	0.91	
Parallax	0.79	0.77	0.77	0.81	0.83	0.84	

Table 3. Comparison of stability score.

The data in Table 3 are the average values of different models obtained for different classes. If videos were compared on a one-to-one basis, there would be videos with large gaps in their stability scores in each category for the different models. Because the stability score of the gap is relatively small, it is generally not obvious in the subjective results. The stability score is related to many factors, including video content, video brightness and darkness, etc. Thus, if the difference between the stability scores obtained for a single video is small, this is not visually obvious, but if the difference between the stability scores of a video is large, it is visually obvious. So the average represents the average situation of a certain category.

#### 4.5. Comparison of Runtimes

Table 4 shows a comparison of the different state-of-the-art methods in terms of their runtime, including traditional and DNN methods. Under the same experimental conditions, the runtime of the model presented in this paper was the shortest compared to the other models. Our model includes fewer operations, so it has a lower runtime. It was found through our experiments that the most time-consuming part of the process the stage involving the generation of the trajectory and obtaining the optical flow. During the trajectory generation process, each key point should be processed separately, and the key points should be propagated to the grid vertices and processed on the grid vertices. The overall time required was as follows:

Grid-based time consumption can be expressed as in Equation (14):

$$T_g = A \times B \times C \times n \tag{14}$$

where *n* is the number of key points.

Pixel-based time consumption can be expressed as in Equation (15):

$$T_p = H \times W \times C \times n \tag{15}$$

where *H* and *W* are the resolutions of the videos.

Table 4. Comparison of runtimes of different methods of stabilization.

Method	Runtime per Frame			
MeshFlow [1]	2576 ms			
Liu et al. [5]	1360 ms			
DUT [14]	198 ms			
Yu et al. [15]	570 ms			
ours	134 ms			

In the trajectory generation stage, we constructed a Boolean matrix by matricizing all operations so that the index could be used to determine whether the current point was required and to extract the corresponding movement deviation based on the index. The extraction of the optical flow was also found to be a time-consuming step when using RAFT. RAFT itself was found to be a means of generating optical flow via iteration, The accuracy of optical flow had little influence on the experimental results between 20 and 10 iterations, so the number of iterations was adjusted to 10, as shown in Figure 11.

14 of 15



**Figure 11.** RAFT visualization results for different iterations: (**a**) shows the visualization for 20 iterations, (**b**) shows the visualization for 10 iterations.

## 5. Conclusions

In this paper we present a method for video stabilization using an unsupervised approach. The accurate motion trajectory of key points was obtained by post-processing the key points and the optical flow in the maximum flat area. The trajectory was also smoothed using a deep learning approach. Finally, for the generated frames with black edges, an adaptive cropping method was used to crop them so that the original frame details were preserved as much as possible. On the benchmark dataset, we verified the superiority of our model over other state-of-the-art methods, finding that it had less distortion and larger stability scores, minimizing black edges while maximizing the preservation of the original frames. In terms of computational efficiency, our method was more efficient than the other methods tested.

The model presented in this paper is not yet an end-to-end model, so we hope to improve it into an end-to-end model. Moreover, to further improve the performance of the model presented in this paper, we hope that online stabilization can be implemented.

**Author Contributions:** Analysis, C.H. and Y.L.; methodology, Y.L.; software, Y.L. and B.W.; supervision, C.H.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and C.H.; data processing, B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank all the anonymous reviewers for their constructive comments and also thank all the editors for their careful proofreading.

Conflicts of Interest: The authors declare no conflict of interest.

## References

- Liu, S.; Tan, P.; Yuan, L.; Sun, J.; Zeng, B. Meshflow: Minimum Latency Online Video Stabilization. In Proceedings of the European Conference on Computer Vision; Springer: Berlin, Germany, 2016; pp. 800–815.
- Liu, S.; Wang, Y.; Yuan, L.; Bu, J.; Tan, P.; Sun, J. Video stabilization with a depth camera. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 89–95.
- Hu, R.; Shi, R.; Shen, I.; Chen, W. Video stabilization using scale-invariant features. In Proceedings of the 2007 11th International Conference Information Visualization (IV'07), IEEE, Zurich, Switzerland, 4–6 July 2007; pp. 871–877.
- 4. Yu, J.; Ramamoorthi, R. Selfie video stabilization. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 551–566.
- 5. Liu, S.; Yuan, L.; Tan, P.; Sun, J. Bundled camera paths for video stabilization. ACM Trans. Graph. (TOG) 2013, 32, 1–10. [CrossRef]
- Wang, Y.S.; Liu, F.; Hsu, P.S.; Lee, T.Y. Spatially and temporally optimized video stabilization. *IEEE Trans. Vis. Comput. Graph.* 2013, 19, 1354–1361. [CrossRef] [PubMed]
- Liu, S.; Yuan, L.; Tan, P.; Sun, J. Steadyflow: Spatially smooth optical flow for video stabilization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Online Conference, Columbus, OH, USA, 23–28 June 2014; pp. 4209–4216.
- 8. Matsushita, Y.; Ofek, E.; Ge, W.; Tang, X.; Shum, H.Y. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1150–1163. [CrossRef] [PubMed]

- 9. Grundmann, M.; Kwatra, V.; Essa, I. Auto-directed video stabilization with robust 11 optimal camera paths. *CVPR IEEE* 2011, 2011, 225–232.
- 10. Liu, F.; Gleicher, M.; Jin, H.; Agarwala, A. Content-preserving warps for 3D video stabilization. *ACM Trans. Graph. (TOG)* **2009**, 28, 1–9.
- 11. Wang, M.; Yang; G.-Y.; Lin, J.-K.; Zhang, S.-H.; Shamir, A.; Lu, S.-P.; Hu, S.-M. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Trans. Image Process.* **2018**, *28*, 2283–2292. [CrossRef] [PubMed]
- 12. Zhao, M.; Ling, Q. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE Trans. Image Process.* 2020, 29, 3582–3595. [CrossRef] [PubMed]
- Xu, S.Z.; Hu, J.; Wang, M.; Mu, T.J.; Hu, S.M. Deep video stabilization using adversarial networks. *Comput. Graph. Forum.* 2018, 37, 267–276. [CrossRef]
- 14. Xu, Y.; Zhang, J.; Maybank, S.J.; Tao, D. Dut: Learning Video Stabilization by Simply Watching Unstable Videos. *arXiv* 2020, arXiv:2011.14574.
- Yu, J.; Ramamoorthi, R. Learning video stabilization using optical flow. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online Conference, Seattle, WA, USA, 13–19 June 2020; pp. 8159–8167.
- 16. Huang, C.H; Yin, H; Tai, Y.W; Tang, C.K. Stablenet: Semi-online, multi-scale deep video stabilization. arXiv 2019, arXiv:1907.10283.
- 17. Kalman, R.E. A new approach to linear filtering and prediction problems. J. Basic Eng. Mar 1960, 82, 35–45. [CrossRef]
- 18. Karpenko, A.; Jacobs, D.; Baek, J.; Levoy, M. Digital video stabilization and rolling shutter correction using gyroscopes. *CSTR* **2011**, *1*, 13.
- 19. Zhang, G.; Hua, W.; Qin, X.; Shao, Y.; Bao, H. Video stabilization based on a 3D perspective camera model. *Visual Comput.* **2009**, 25, 997–1008. [CrossRef]
- Bhat, P.; Zitnick, C.L.; Snavely, N.; Agarwala, A.; Agrawala, M.; Cohen, M.; Curless, B.; Kang, S.B. Using photographs to enhance videos of a static scene. In Proceedings of the 18th Eurographics Conference on Rendering Techniques, Grenoble, France, 25–27 June 2007; pp. 327–338.
- 21. Choi, J.; Kweon, I.S. Deep iterative frame interpolation for full-frame video stabilization. *ACM Trans. Graph. (TOG)* **2020**, *39*, 1–9. [CrossRef]
- DeTone, D.; Malisiewicz, T.; Rabinovich, A. Superpoint: Self-supervised interest point detection and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 224–236.
- 23. Teed, Z.; Deng, J. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2020; pp. 402–419.
- Wulff, J.; Black, M.J. Efficient Sparse-to-Dense Optical Flow Estimation Using a Learned Basis and Layers. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Online Conference, Boston, MA, USA, 7–12 June 2015; pp. 120–130.
- Suzuki, S. Topological structural analysis of digitized binary images by border following. *Comput. Vision Graph. Image Process.* 1985, 30, 32–46. [CrossRef]