# Video Stabilization using Robust Feature Trajectories

Ken-Yi Lee Yung-Yu Chuang Bing-Yu Chen Ming Ouhyoung
National Taiwan University
{kez|cyy|robin|ming}@cmlab.csie.ntu.edu.tw

## Abstract

*This paper proposes a new approach for video stabilization. Most existing video stabilization methods adopt a framework of three steps, motion estimation, motion compensation and image composition. Camera motion is often estimated based on pairwise registration between frames. Thus, these methods often assume static scenes or distant backgrounds. Furthermore, for scenes with moving objects, robust methods are required for finding the dominant motion. Such assumptions and judgements could lead to errors in motion parameters. Errors are compounded by motion compensation which smoothes motion parameters. This paper proposes a method to directly stabilize a video without explicitly estimating camera motion, thus assuming neither motion models nor dominant motion. The method first extracts robust feature trajectories from the input video. Optimization is then performed to find a set of transformations to smooth out these trajectories and stabilize the video. In addition, the optimization also considers quality of the stabilized video and selects a video with not only smooth camera motion but also less unfilled area after stabilization. Experiments show that our method can deal with complicated videos containing near, large and multiple moving objects.*

## 1. Introduction

Hand-held video capturing devices become more and more accessible today because of lowering prices and reducing sizes. While people enjoy the luxury of capturing interesting moments with more ease, it also leads to a great amount of videos captured casually by amateurs without carefully planning [7]. Such a casual video often exhibits annoying jitter due to shaky motion of an unsteady hand-held camera or a vehicle-mounted camera. Therefore, to enhance user's experiences in watching casual videos, *video stabilization*, removing unwanted video perturbations due to unstable camera motions, plays an important role.

A video can be stabilized during recording or with an offline post-processing. Nowadays, many cameras are equipped with hardware stabilizers. Expensive high-end cameras usually have sophisticated sensors and lens systems to remove camera shake during recording. Cheaper cameras may use sensors to estimate camera motion and built-in firmware to properly crop the acquired images for compensating the jittered motion [10]. Such online solutions are often not sufficient for dealing with sequences with complicated motions. Therefore, even using a camera with an online stabilizer, unwanted camera jerks still happen frequently in videos taken by non-professional users. Thus, offline video stabilization algorithms are still required for making these videos steady.

Most video stabilization methods follow a three-step framework consisting of three steps: *motion estimation*, *motion compensation* (also called *motion smoothing* or *motion filtering*) and *image composition* [15]. Motion estimation estimates the motion between frames. The estimated motion parameters are forwarded to motion compensation, which damps camera motion by removing high-frequency fluctuations and computes the global transformation necessary to stabilize the current frame. Finally, image composition warps the current frame according to that transformation and generates the stabilized sequence. Optionally, image composition could include inpainting and deblurring to further improve the quality of the stabilized video. Most previous video stabilization methods focused on improving components of this framework. For example, many algorithms have been proposed for improving camera compensation by better smoothing out camera motion or recovering the intentional camera motion [1, 6, 11, 16, 20]. Some focused on improving image composition by motion inpainting and deblurring [14]. However, this three-step framework was kept intact in most video stabilization algorithms.

Most previous video stabilization methods need to make assumptions on motion models for motion estimation. Flexible motion models, such as optical flow, could be used but they bring more challenges on smoothing motion parameters for motion compensation. Therefore, simpler models such as planar homography are more popular. Although these models can be fitted more robustly, they have many restrictions on scene and motion properties. For scenes with great depth variations or multiple moving objects, robust

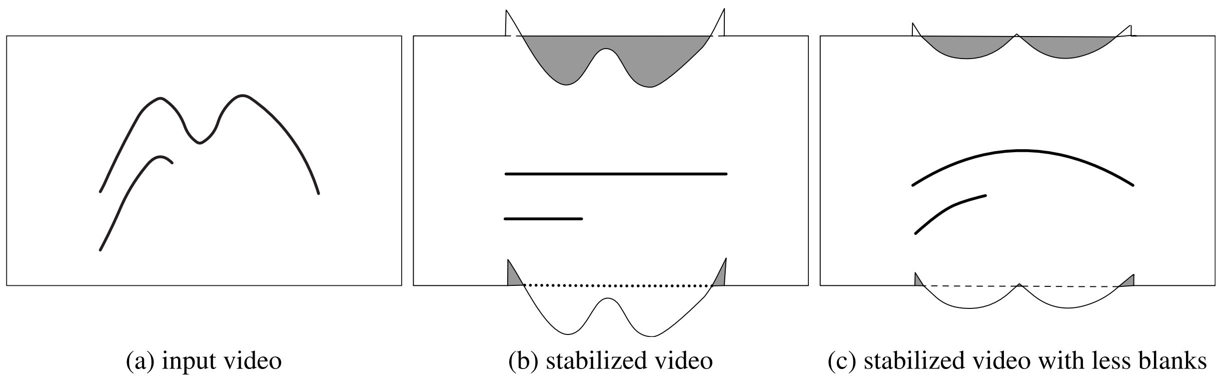


Figure 1. The basic idea of the proposed method. These are 2D illustrations of 3D video volumes. Curves in the input video (a) represent feature trajectories. By deforming the video volume, one can obtain a stabilized video (b). Trajectories become smooth in the deformed image space. The video is perceived smooth because the features move smoothly along time in the video volume. However, some areas become unfilled (as shaded in the figure). Our method seeks for a video which is not only stabilized but also has less uncovered areas (c).

means such as RANSAC have to be used to find the dominant motion. The assumptions on motion properties and judgements of dominant motion often lead to inaccuracy in estimated motion. Motion compensation based on such inaccurate estimation could further aggravate the problem mostly due to the accumulative errors of cascaded transformation chains. Therefore, these methods often have difficulties with videos containing more complicated motion such as multiple large moving foreground objects.

This paper proposes a video stabilization method based on two observations. The first is well-tracked features usually imply structured regions of background or of interest. At the same time, viewer's perception on whether a video is stabilized is often related to whether those regions move smoothly. Therefore, our method extracts a set of robust feature trajectories. Instead of explicitly estimating camera motion, we find a set of transformations to smoothen the trajectories in the transformed image space. Once the feature trajectories become smooth, the video often looks stabilized. One strength of our approach is that it does not make assumptions on motion properties since feature trajectories represent motion in a non-parametric manner. In addition, by looking at long trajectories, it is more likely to recover the true intentional motion than only checking a pair of frames at a time. The second observation is, while most stabilization methods find the best camera motion in terms of the smoothness measurement they used, there are actually many camera motions that viewers feel stabilized enough. To avoid leaving large blank areas for image composition to fill, it is often possible to find a sufficiently smooth motion with less blank areas. The result is a stabilized video with both smooth camera motion and less unfilled areas. It alleviates the task of image composition, thus improving visual quality of the stabilized video.

Figure 1 illustrates the basic idea. Figure 1(a) shows a 2D illustration of the 3D volume for the input video. There are two feature trajectories shown as curves. One can stabilize the video by deforming the video volume for smoothing out the trajectories as shown in Figure 1(b), the smoothest one in terms of smoothness. However, the video in Figure 1(c) could also be perceived as smooth by viewers but has less uncovered areas. Our algorithm seeks for such smooth videos with less uncovered areas.

In summary, our approach directly stabilize a video without explicitly estimating camera motion. In addition, it considers both video motion smoothness and video quality degradation simultaneously. The tradeoffs among video motion smoothness, image distortion and the amount of unfilled areas are explicitly adjustable in our method. Thus, our method essentially unifies both motion estimation and motion compensation into a single step while taking the needs of image composition into account.

## 2. Related work

Most previous video stabilization methods follow the same framework and focus on improving components. Here, we review them by components.

***Motion estimation.*** Motion estimation has long been an important research topic for computer vision. Much work has been done in optical flow [3] and image alignment [18]. A common motion estimation approach for video stabilization is to estimate a dominant planar homography that stabilizes a large planar region in the video [5]. Both direct and feature-based methods have been used for homography estimation. Homography-based schemes generally performs quite well for stabilizing planar scenes or rotational camera motions but suffers from highly non-planar scenes. Jin *et al*. proposed a 2.5D motion model to handle sequences with significant depth changes [9]. Chang *et al*. estimated optical flow and then fitted the computed flow field to a simplified affine motion model [6].

***Motion compensation.*** The goal of motion compensation is to remove high-frequency jitters from the estimated camera motion. It is the component that most video stabilization algorithms attempt to improve and many methods have been proposed, such as motion vector integration [1], particle filter [20], variational method [16], regularization [6], Kalman filter [11] and local parabolic fitting [8].

***Image composition.*** Some focus on improving video quality degradation caused by stabilization. Hu *et al.* used dynamic programming to reduce the visual artifacts in the boundary of defined areas and mosaics [8]. Matsushita *et al.* used motion inpainting to fill in the unfilled area for full-frame stabilization and deblurring for further improving video quality [14]. Gleicher *et al.* compensated estimated camera motions and considered video quality loss at the same time [7].

## 3. Robust feature trajectories

Extracting robust feature trajectories from a video is a crucial step for our method. It is certainly possible to concatenate frame-by-frame feature matches to obtain long-range trajectories. However, feature matching is best suited to successive pairs of frames, not to long sequences. A false match often severely messes up the whole feature trajectory. Thus, trajectories must be concatenated and pruned carefully to avoid false matches' terrible impacts.

Our robust feature trajectory extraction algorithm incorporates spatial motion consistency to reduce false matches and temporal motion similarity of trajectories for long-range tracking. It is a marriage of particle video [17] and SIFT flow [12]. The overall flow of our algorithm is similar to particle video, a framework for estimating dense particle-level long-range motion of a video based on optical flow. For efficiency and robustness, we adapt this framework but using reliable feature tracking. On the other hand, although the workflow is similar to particle video, our optimization is more similar to SIFT flow as it is formulated as a discrete optimization problem.

Our algorithm retrieves feature trajectories by making a forward sweep across the video. Here, a trajectory $\xi_j$ is a set of 2D points $\{\mathbf{p}_j^t | t_s \leq t \leq t_e\}$ across time defined between its start time $t_s$ and end time $t_e$. Assume that, after finishing frame $t$, we have a set of live trajectories $\Gamma = \{\xi_j\}$. For the next frame $t+1$, the following four steps are performed to reliably extend the live trajectories to the next frame.

***Addition.*** We use feature detection to generate feature points $\{\mathbf{f}_i^{t+1}\}$ for frame $t+1$. Our current implementation uses SIFT [13] for its accuracy and robustness in different lighting and blurring conditions.

***Linking.*** For incorporating motion consistency to spatially neighboring features, we need to assign neighbor relationships to features. Delaunay triangulation is applied on the detected feature points. If any pair of feature points share an edge, they are considered neighbors. Such neighborhood information propagates with the trajectories through the entire video until the trajectories are pruned. In other words, two trajectories are neighbors as long as they share an edge in some frame.

***Propagation by optimization.*** Feature points are propagated by using feature matching. If there is a similar feature in the next frame, the feature point will be propagated. However, there might be many similar features in the next frame, an optimization step helps choose the best match to propagate by considering feature similarity and neighbors' motion consistency. If temporal movement of two trajectories exhibits a similar pattern in the past, it is more likely that they will move similarly at the current frame as they are often on the same surface patch. Thus, the algorithm prefers to move two trajectories with high temporal motion similarity in a similar manner.

Let $\Phi^t$ be a match assignment for frame $t$, which assigns feature points at frame $t+1$ to trajectories at time $t$. For example, $\Phi^t(i) = j$ means to extend the $i$-th trajectory $\xi_i$ to the $j$-th feature point $\mathbf{f}_j^{t+1}$ at frame $t+1$. In other words, it means to assign $\mathbf{f}_j^{t+1}$ as $\mathbf{p}_i^{t+1}$, the point at frame $t+1$ for trajectory $\xi_i$. Note that $\Phi^t$ is injective as it describes correspondences. It also means that $\Phi^t(i)$ could be null if there are more trajectories than detected features. With such an assignment, the motion flow $\mathbf{u}_i^t$ for the $i$-th trajectory at frame $t$ for this assignment $\Phi^t$ can then be define as $\mathbf{u}_i^t(\Phi^t) = \mathbf{f}_{\Phi^t(i)}^{t+1} - \mathbf{p}_i^t$ since $\mathbf{f}_{\Phi^t(i)}^{t+1}$ is frame $t+1$'s correspondence for $\mathbf{p}_i^t$.

Our goal is to find the optimal assignment which assigns good matches but also maintains good motion similarity for neighboring features by minimizing the following cost function:

$$\begin{aligned} E(\Phi^t) = & \sum_i \parallel s(\mathbf{p}_i^t) - s(\mathbf{f}_{\Phi^t(i)}^{t+1}) \parallel^2 \\ & + \lambda_t \sum_{(i,j)\in\mathbf{N}} w_{ij} \parallel \mathbf{u}_i^t(\Phi^t) - \mathbf{u}_j^t(\Phi^t) \parallel^2, \end{aligned} \quad (1)$$

where $s(\mathbf{p})$ is the SIFT descriptor of point $\mathbf{p}$, $\mathbf{N}$ is the set of all neighbor pairs and $\lambda_t$ controls the tradeoff between two terms. The first term penalizes feature mismatches and the second term punishes motion inconsistency of neighboring features. Similar to particle video, link weights $w_{ij}$ is measured by a weighted sum of past motion similarity as

$$\begin{aligned} \mathbf{v}_i^t = & \mathbf{p}_i^{t+1} - \mathbf{p}_i^t, \\ D_{ij} = & \frac{1}{|\tau|} \sum_{t\in\tau} \parallel \mathbf{v}_i^t - \mathbf{v}_j^t \parallel^2, \\ w_{ij} = & G(\sqrt{D_{ij}}; \sigma_d), \end{aligned}$$

where $\tau$ is a set of previous frames and $G(\cdot;\sigma)$ is a zero-mean Gaussian with the standard deviation $\sigma$. $D_{ij}$ measures the motion similarity between trajectory $i$ and trajectory $j$ by averaging their flow differences for a number of frames.

Similar to SIFT flow, the problem of robust feature correspondence is formulated as a discrete optimization problem as shown in Equation 1. We use a greedy best-match approach to speed up the optimization. Although there is no guarantee for global optimum, it works pretty well in practice. In addition, even if the greedy approach may discard some possible alternatives and lead to mismatches, this problem can be corrected in the next step.

***Pruning.*** After propagation, all trajectories at the current frame are stretched to correspondent positions in the next frame. If there are more trajectories than detected features, unassigned trajectories are terminated, removed from the live trajectory set $\Gamma$ and added to a retired trajectory set $\Pi$. Similarly, if there are more features than trajectories, the unmapped features will start new trajectories and are added to $\Gamma$ for the next frame. In addition, extended trajectories with low feature similarity and/or bad neighbor consistency will also be pruned and added to the retired trajectory set $\Pi$. The cost of trajectory $i$ is defined as

$$\parallel s(\mathbf{p}_i^t) - s(\mathbf{f}^{t+1}_{\Phi^t(i)}) \parallel^2 + \lambda_t \sum_{j \in \mathbf{N}(i)} w_{ij} \parallel \mathbf{u}_i^t(\Phi^t) - \mathbf{u}_j^t(\Phi^t) \parallel^2$$

and trajectories with costs higher than a threshold are pruned. Remaining matches are used to extend the trajectories by letting $\xi_i = \xi_i \cup \{\mathbf{f}^{t+1}_{\Phi^t(i)}\}$, *i.e.*, assigning $\mathbf{f}^{t+1}_{\Phi^t(i)}$ to $\mathbf{p}_i^{t+1}$. These trajectories are kept in $\Gamma$ for the next frame.

By repeating the above four steps frame by frame through the entire video, we can retrieve a set of robust feature trajectories $\Pi$ from the input video. It is worthy noted that, since neighbor information is considered during tracking, features tend to move as a patch but not as single points. This is helpful for our stabilization.

## 4. Video stabilization by optimization

Smoothness of a curve is often measured by the sum of squared values of the second derivative (acceleration). Therefore, in a perfectly stabilized video, features should move in a constant velocity in image space. Thus, instead of estimating and compensating real camera motions, we estimate a set of geometric transformations so that features move along their trajectories approximately in a constant velocity in the image space after being transformed.

Another factor we take into account is image quality degradation after transformation. When a frame is warped by a transformation, the image quality may be degraded. This degradation comes from two sources. First, after transformation, some areas become uncovered in the warped frame. Second, a transformation with a scale factor larger than one stretches the image. Although these problems could be alleviated by inpainting and super-resolution algorithms, these algorithms are not perfect and artifacts may still remain in the stabilized video. Our solution is to consider image degradation during the search of transformations for smoothing transformed trajectories.

In the following, we first present how trajectories are weighted for robustness (Section 4.1), then explain the construction of the objective function (Section 4.2), and finally describe how to find the optimal solution (Section 4.3).

### 4.1. Trajectory weights

Section 3 retrieves a set of robust trajectories $\Pi = \{\xi_i\}$ from the input video. Although there are good trajectories that do relate to real camera shakes, some include wrong matches and some belong to self-moving objects unrelated to camera motion. Therefore, not every trajectory has equal importance and we need a mechanism for emphasizing impacts of good trajectories. When a trajectory comes from a region of background or of interest, it should have higher importance. In general, longer trajectories are more important as they often represent objects of interest in motion or static background due to the moving camera. Both cases are useful for stabilizing a video. On the other hand, fast motions often lead to short trajectories as the feature matching is less reliable. As fast foreground motion is often not related to camera shakes, we would like to depreciate short trajectories when stabilizing cameras. Therefore, we measure the importance of a trajectory by its temporal persistence and give higher weights to longer trajectories. In addition, the weight should be a function of time as well. That is, a trajectory could contribute differently for different frames. A trajectory should have more impact to a frame if the frame is on the middle of the trajectory. On the other hand, an emerging or vanishing trajectory should have less impact to the frame. Thus, we used a triangle function to give higher weights toward the middle of the trajectory. Let's denote the contribution of the $i$-th trajectory $\xi_i$ to frame $t$ as $w_i^t$ which is proportional to $l_i^t = \min(t - t_s(\xi_i), t_e(\xi_i) - t)$, where $t_s(\xi_i)$ and $t_e(\xi_i)$ denote the start time and end time of $\xi_i$.

Finally, a couple of normalization steps are required. A spatial normalization is used to avoid dominant spots if many trajectories gather around some spots (Equation 2). This normalization has the effects to consider trajectories more evenly in space. A temporal normalization is applied to normalize trajectory weights of a frame by the number of trajectories going through the frame (Equation 3).

$$\tilde{w}_i^t = \frac{l_i^t}{\sum_{j \in \Pi(t)} G(|\mathbf{p}_j^t - \mathbf{p}_i^t|; \sigma_s)}, \qquad (spatial) \quad (2)$$

$$w_i^t = \frac{\tilde{w}_i^t}{\sum_{j \in \Pi(t)} \tilde{w}_j^t}, \qquad (normalization) \quad (3)$$

where $\Pi(t)$ is the set of trajectories going through frame $t$.

Note that, in Equation 2, if there are more trajectories close to $\xi_i$, the denominator of $\tilde{w}_i^t$ becomes larger, thus reducing the impact of $\xi_i$ and similarly for trajectories in that cluster.

## 4.2. Objective function

To stabilize a video by considering feature trajectory smoothness and video quality degradation simultaneously, our objective function has two terms, one for roughness of trajectories and the other for the image quality loss after being transformed. Our goal is to find a geometric transformation $T_t$ for each frame $t$ to minimize the objective function. To be more precise, for a video of $n$ frames, the goal is to select a set of transformations, $\mathbf{T} = \{T_t | 1 \leq t \leq n\}$ satisfying the following criterion:

$$\arg\min_{\mathbf{T}} E_{roughness}(\mathbf{T}) + \lambda_d E_{degradation}(\mathbf{T}), \quad (4)$$

where $\lambda_d$ is a parameter for adjusting the tradeoff between trajectory roughness and video quality degradation.

In Equation 4, $E_{roughness}$ represents the roughness of feature trajectories. The roughness is relate to accelerations of a trajectory. If trajectory $\xi_i$ goes through frames $t-1$, $t$, and $t+1$, then the acceleration $\mathbf{a'}_i^t(\mathbf{T})$ of $\xi_i$ in the transformed frame $t$ can be defined as

$$\mathbf{a'}_i^t(\mathbf{T}) = T_{t+1}\mathbf{p}_i^{t+1} - 2T_t\mathbf{p}_i^t + T_{t-1}\mathbf{p}_i^{t-1}.$$

Note that, the roughness should be measured relative to the original frame, not the transformed one. Thus, we measure the roughness of a trajectory $\xi_i$ in frame $t$ as $\| T_t^{-1}\mathbf{a'}_i^t(\mathbf{T}) \|^2$ instead of $\| \mathbf{a'}_i^t(\mathbf{T}) \|^2$. Otherwise, transformations mapping all points to a single point will have the best smoothness. By summing up weighted roughness values of all trajectories, we obtain the roughness cost term,

$$E_{roughness}(\mathbf{T}) = \sum_{\xi_i \in \Pi} \sum_{t=2}^{n-1} w_i^t \| T_t^{-1}\mathbf{a'}_i^t(\mathbf{T}) \|^2 .$$

Note that $w_i^t = 0$ if $\xi_i$ does not go through frame $t$.

$E_{degradation}$ in Equation 4 represents the quality loss of the transformed frames, which consists of three components $E_{distortion}$, $E_{scale}$, and $E_{uncovered}$,

$$E_{degradation}(\mathbf{T}) = \sum_{t=1}^{n}(E_{distortion}(T_t) + \lambda_s E_{scale}(T_t) + \lambda_u E_{uncovered}(T_t)),$$

where $\lambda_s$ and $\lambda_u$ are parameters for adjusting the tradeoffs among different types of quality loss.

The definition of $E_{distortion}(T_t)$ depends on the type of the geometric transformation. It is usually measured by the difference between itself and a correspondent similarity transformation. $E_{scale}(T_t)$ is related to the scale/zoom factor of the transformation. When the scale factor is larger than 1.0, $E_{scale}$ is non-zero as scale-up will introduce quality degradation. $E_{uncovered}(T_t)$ measures the area where might be unfilled after the transformation is applied.

Our current implementation uses both similarity and affine transformations for video stabilization. The choice is made by users. For reducing optimization complexity and increasing efficiency, the use of similarity transformations is a better choice and it often produces good results in most cases. Here, we take similarity transformations as an example. A similarity transformation,

$$T_t = s_t\mathbf{R}(\theta_t) + \mathbf{\Delta}_t = \begin{bmatrix} s_t\cos(\theta_t) & -s_t\sin(\theta_t) & \Delta_{x,t} \\ s_t\sin(\theta_t) & s_t\cos(\theta_t) & \Delta_{y,t} \end{bmatrix},$$

has four parameters. As it is required that $T_t^{-1}$ exists (implying $s_t \neq 0$), and mirror transformations are not favorite (implying $s_t \geq 0$), we add a constraint, $s_t > 0$ for every transformation $T_t$ into the optimization.

When warping an image by a similarity transformation, any rotating angle $\theta_t$ or translation $\mathbf{\Delta}_t$ will not decrease image quality. But a scale factor $s_t$ larger than 1 may result in loss of image details. A larger scale factor will result in greater quality degradation. Hence, we use the following costs for similarity transformations,

$$E_{distortion}(T_t) = 0$$
$$E_{scale}(T_t) = \begin{cases} (s_t - 1)^4 & \text{if } s_t > 1 \\ 0 & \text{otherwise} \end{cases}$$

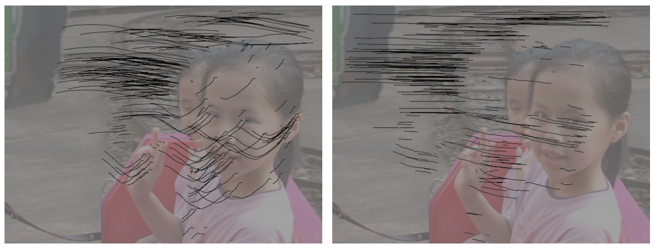
As to $E_{uncovered}(T_t)$, a naïve solution would be to calculate the uncovered area after transformations. Since this is time-consuming, an approximation is used to speed up.

$$E_{uncovered}(T_t) = \sum_{v\in\mathcal{V}, e\in\mathcal{E}} c(T_t^{-1}p, e)^2$$
$$c(v,e) = \begin{cases} len(e)\Psi(dist(v,e)) & \text{if } v \text{ is outside of } e, \\ 0 & \text{otherwise.} \end{cases}$$
$$\Psi(d) = \sqrt{d^2 + \delta^2} - \delta,$$

where $\mathcal{V}$ is the set of corner points of a frame; $\mathcal{E}$ is the set of contour edges of a frame; $len(e)$ is the length of $e$; and $dist(v, e)$ is the distance between a point $v$ and an edge $e$. Here, the use of $\Psi$ is in spirit of Brox's work [4] and $\delta$ is set to a small number, 0.001.

## 4.3. Optimization

We have formulated the video stabilization problem as a constrained nonlinear least squares fitting problem (Equation 4). We solve it by Levenberg-Marquardt algorithm. A good initial guess is often crucial for solving such a nonlinear optimization. In our case, we used slightly scaled identity transformations with scale factor 1.01 as initial guesses



(a) before stabilization (b) after stabilization

Figure 2. Feature trajectories before stabilization (a) and after stabilization (b). Before stabilization, feature trajectories are not smooth in the image space. Note that many long trajectories were extracted from the background. Trajectories related to foreground are less and often shorter. Thus, background has more impacts to the stabilization process. After stabilization, feature trajectories become much smoother and the video is more stabilized.

and found it works quite well in practice. In addition, for processing a long video, the optimization problem could be broken into several smaller subproblems and solved by an overlapped sliding window approach.

## 5. Results

We have applied the proposed algorithm to a number of video sequences. The final results are best viewed in video form (as in the supplementary material), but we present stills for several examples. Figure 2 displays trajectories before and after stabilization. As the background has more and longer trajectories, it has greater impacts on the stabilization. After stabilization, trajectories are much smoother and the video is stabilized. Although there is a large moving object in the foreground, our method does not have problem with finding the right object to stabilize. series of temporal transformations. several frames for two stabilized sequences. The top one is considered easier as it only contains small moving objects.

We compare our method with a traditional approach and a video stabilization program, *Deshaker* [19]. For the traditional approach, we used SIFT and RANSAC for motion estimation, and a low-pass filter for motion compensation. It is similar to motion estimation of Matsushita *et al.*'s work [14] in spirit. We have also tested other commercial programs, such as *iStabilize* and *iMovie*, but their results are not as good as Deshaker and not listed here. None of these methods implements inpainting and deblurring for fair comparisons. Figures 4 and 5 show stabilized frames of these methods on two sequences. Both sequences have significant camera motion and complicated foreground object movement. Table 1 shows the ratio of uncovered areas for these three methods. Overall, our method gives the most stabilized videos and leaves the least uncovered areas. Please see the supplementary video for comparisons.

| sequence | Traditional | Deshaker | Proposed |
|---|---|---|---|
| themepark (Fig 4) | 4.2809% | 5.1455% | 3.0312% |
| skiing (Fig 5) | 3.9608% | 4.0559% | 2.3641% |

Table 1. Un-coverage ratio for three methods on two sequences.

All results used the same set of parameters, $\lambda_d = 1$, $\lambda_s = 50$ and $\lambda_u = 1$. Users can also adjust them to pay more attention to what they care more. The adjustments are intuitive: a smaller $\lambda_d$ gives smoother motion; a larger $\lambda_s$ produces less blur and a larger $\lambda_u$ reduces the unfilled area. As for computation time, on average, our method took 2.58 seconds per frame on the examples in the paper. (51% for feature detection, 26% for robust trajectories and 23% for optimization and frame warping.) The time is comparable to our implemented traditional approach. Our method is faster than some commercial programs, but slower than Deshaker which took roughly 1 second per frame.

## 6. Conclusions and future work

We have proposed a novel approach for video stabilization. By using robust feature trajectories, we can estimate a set of transformations to stabilize a video by considering user intention and video quality degradation simultaneously. In contrast to most existing approaches, our approach doesn't require estimation of camera motion. Hence, our approach does not suffer from the problems with insufficient motion models and inaccurate motion classification. It can be applied to a large variety of videos and experiments show it is more robust than previous methods.

Our method inherits most limitations from the conventional 2D-transform-based methods. For example, it underperforms when there are significant depth variations. 3D methods [2, 21] could avoid the problem but often rely on unreliable depth estimation. For reliable depth estimation, they often put restrictions on input videos by, for example, assuming majority of the scene is static. Therefore, these approaches often fail terribly to handle videos with large moving foreground objects. Thus, 2D methods are still more preferable in practice. One additional limitation of our method is that it requires enough detected and tracked features. Our current implementation requires at least five feature trajectories passing through a frame for reliable results. However, for videos with insufficient features, traditional approaches suffer from inaccurate motion estimation as well. Robust motion estimation for plain backgrounds is worth of further exploration. In addition, SIFT detection is more accurate but considerably slower. With the robust feature match procedure, other less accurate but faster features could be used. Finally, incorporation of video enhancement techniques such as motion inpainting [14] would greatly improve the visual quality of the stabilized video.

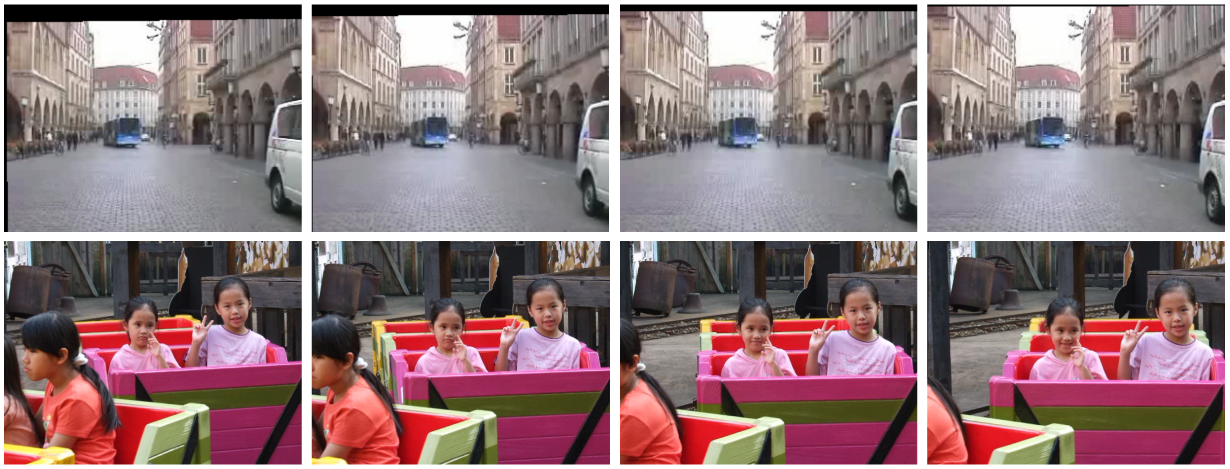


Figure 3. Results of our video stabilization algorithm on two sequences. Top: *street* (http://www.pixlock.com/). Bottom: *toycar*
.

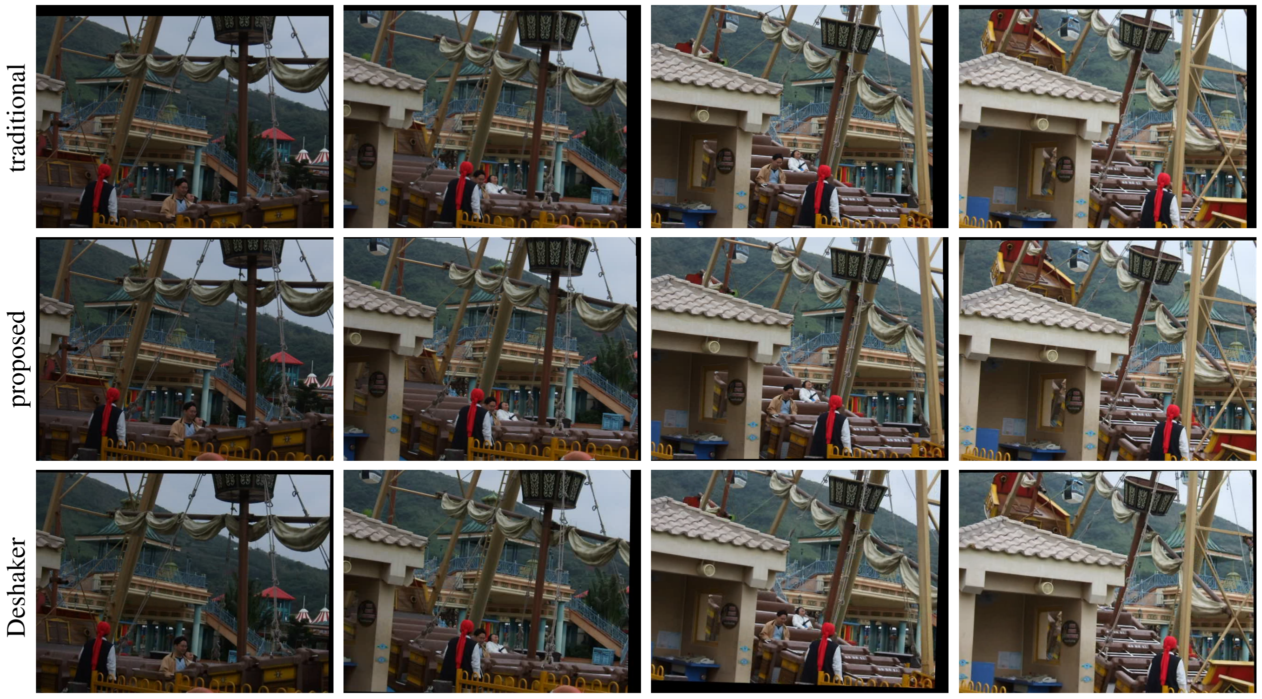


Figure 4. Comparisons to a traditional approach and Deshaker on the sequence, *themepark*. Top row: a traditional approach. Middle row: the proposed method. Bottom row: Deshaker. In general, our method gives the most stabilized video and least unfilled area.

## Acknowledgments

The authors would like to thank collaborators from CyberLink Corp. for their suggestions and video clips. This work was partly supported by grants NSC97-2622-E-002-010-CC2 and NTU-98R0062-04.

## References

[1] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato. SIFT features tracking for video stabilization. In *Proceedings of Image Analysis and Processing (ICIAP) 2007*, pages 825–830, September 2007.

[2] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, B. Curless, M. Cohen, and S. B. Kang. Using photographs to enhance videos of a static scene. In *Proceedings Eurographics Symposium on Rendering 2007*, pages 327–338, 2007.

[3] M. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

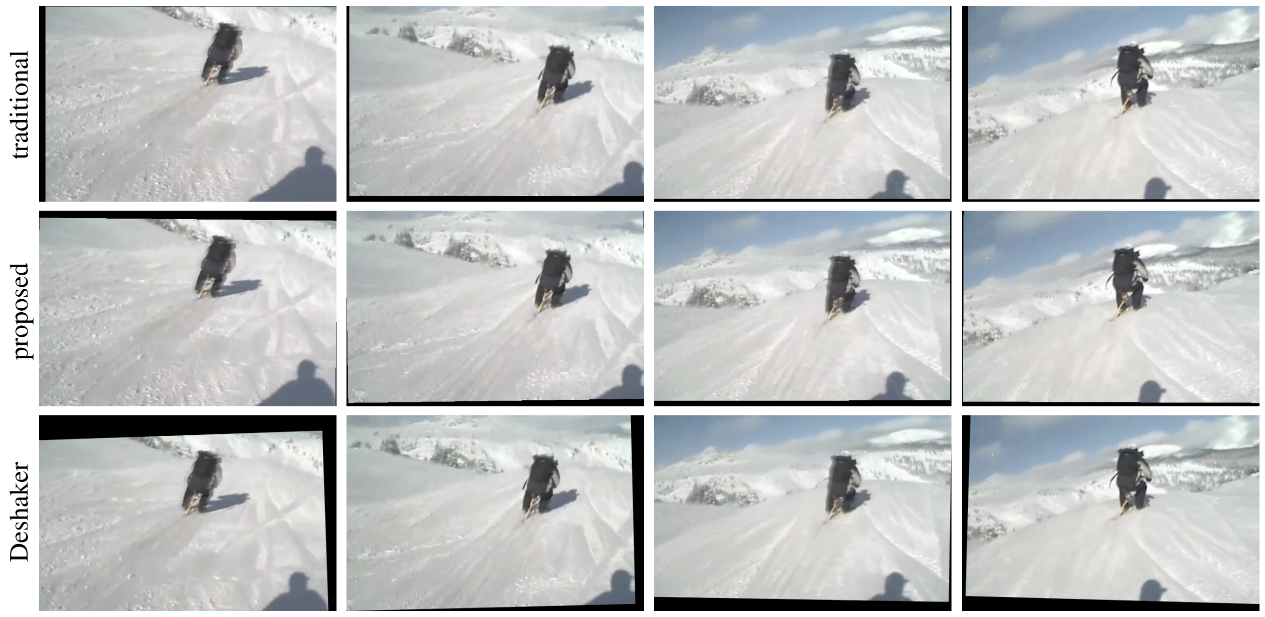


Figure 5. Comparisons to a traditional approach and Deshaker on the sequence, *skiing* (http://www.pixlock.com/). Top row: a traditional approach. Middle row: the proposed method. Bottom row: Deshaker. Our method gives the most stabilized video and least unfilled area.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proceedings of European Conference on Computer Vision 2004*, pages 25–36, 2004.

[5] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *Proceedings of Computer Vision and Pattern Recognition 2001*, pages II.609—614, 2001.

[6] H.-C. Chang, S.-H. Lai, and K.-R. Lu. A robust and efficient video stabilization algorithm. In *Proceedings of IEEE International Conference on Multimedia and Expo 2004*, pages I.29–32, 2004.

[7] M. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 5(1):1–28, 2008.

[8] R. Hu, R. Shi, I. fan Shen, and W. Chen. Video stabilization using scale-invariant features. In *Proceedings of Information Visualization, 2007*, pages 871–877, July 2007.

[9] J. Jin, Z. Zhu, and G. Xu. Digital video sequence stabilization based on 2.5D motion estimation and inertial motion filtering. *Real-Time Imaging*, 7(4):357–365, 2001.

[10] S.-J. Ko, S.-H. Lee, S.-W. Jeon, and E.-S. Kang. Fast digital image stabilizer based on gray-coded bit-plane matching. *IEEE Transactions on Consumer Electronics*, 45(3):598–603, Aug 1999.

[11] A. Litvin, J. Konrad, and W. Karl. Probabilistic video stabilization using Kalman filtering and mosaicking. In *Proceesings of IS&T/SPIE Symposium on Electronic Imaging, Image and Video Communications and Proc*, pages 663–674, 2003.

[12] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT flow: dense correspondence across difference scenes. In *Proceedings of European Conference on Computer Vision 2008*, pages 28–42, 2008.

[13] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[14] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1150–1163, 2006.

[15] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing 1998*, pages V.2789–2792, 1998.

[16] M. Pilu. Video stabilization as a variational problem and numerical solution with the viterbi method. In *Proceedings of Computer Vision and Pattern Recognition 2004*, pages I.625–630, 2004.

[17] P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72–91, 2008.

[18] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(1):1–104, 2006.

[19] G. Thalin. Deshaker, 2008. `http://www.guthspot.se/video/deshaker.htm`.

[20] J. Yang, D. Schonfeld, C. Chen, and M. Mohamed. Online video stabilization based on particle filters. In *Proceedings of IEEE International Conference on Image Processing 2006*, pages 1545–1548, 2006.

[21] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao. Video stabilization based on a 3D perspective camera model. *The Visual Computer*, to appear.