Simultaneous Camera Path Optimization and Distraction Removal for Improving Amateur Video

Fang-Lue Zhang, *Member, IEEE,* Jue Wang, *Senior Member, IEEE,*
Han Zhao, Ralph R. Martin, *Member, IEEE,* Shi-Min Hu, *Member, IEEE*

Abstract—A major difference between amateur and professional video lies in the quality of camera paths. Previous work on video stabilization has considered how to improve amateur video by smoothing the camera path. In this paper, we show that additional changes to the camera path can further improve video aesthetics. Our new optimization method achieves multiple simultaneous goals: (i) stabilizing video content over short time scales, (ii) ensuring simple and consistent camera paths over longer time scales, and (iii) improving scene composition by automatically removing distractions, a common occurrence in amateur video. Our approach uses an L^1 camera path optimization framework, extended to handle multiple constraints. Two-passes of optimization are used to address both low-level and high-level constraints on the camera path. Experimental and user study results show that our approach outputs video which is perceptually better than the input, or the results of using stabilization only.

Index Terms—Camera path, cinematography, distraction

1 Introduction

Consumer camera hardware has developed rapidly, and the built-in cameras on recent mobile devices are capable of capturing high resolution video with high frame rates; this previously required high-end professional cameras. However, a significant gap remains between many amateur videos and professional ones, in the quality of the camera paths achieved (i.e. how the camera is moved and zoomed to capture the scene). The camera path of a professional video is usually carefully planned beforehand, and precisely executed with the support of hardware such as dollies and tracks. In contrast, many amateur videos are taken spontaneously without planning, and without hardware support. Most amateurs lack the skills to carefully design a camera path for a specific scene. Furthermore, even with planning, unexpected, unwanted events can occur within the scene.

Such amateur camera paths can detract from the output video. Firstly, hand shake can cause video content to jitter, making it hard to watch, a problem that has been extensively studied: various video stabilization approaches have been proposed [1], [2], [3]. Secondly, the camera path may contain motions over longer timescales that are undesirable these are often caused by low-frequency body motions of the cameraman such as walking. However, current stabilization techniques generally preserve low-frequency motions, as these may correspond to actions such as panning: see Gleicher and Liu [4]. Finally, amateur camera paths often exhibit sub-optimal scene composition. For instance, the main subject of the video may drift off-center as the camera moves. It is also not uncommon for unwanted objects to enter the scene unnoticed by the videographer, but causing a distraction to the viewer, e.g. an irrelevant dog may run by in the background when filming a child playing.

While video stabilization has been extensively studied, little work has considered how to resolve a wider range of issues by re-planning the camera path, especially in terms of improving scene composition and removing distractions. Furthermore, such issues have been considered in isolation. In this paper, we give an *integrated* solution for resolving all these issues simultaneously, using a novel camera path optimization approach which incorporates multiple constraints. Our system automatically detects distractions that draw the viewers' attention from the main objects in the video, allowing distractions to be eliminated. To improve motion quality, we segment the camera path into coherent pieces and fit a high quality motion element of the kind used in professional video to each segment. We significantly extend the original L^1 optimization framework in [2] to incorporate these additional constraints as well as stabilization.

Performing this task requires consideration of the camera path at different levels. Distraction removal and content stabilization can be addressed locally by examining sequences of consecutive frames. Optimizing the complete set of motion segments requires a global analysis of all camera motions. We thus use a two-pass optimization approach which handles constraints at different levels. We first apply low-level optimization to perform content stabilization, and distraction detection and removal, yielding a modified camera path. We then further analyze this modified path, dividing it into segments and fitting a motion model to each segment. Finally, we incorporate all constraints and the fitted models in a further optimization pass. This produces a final, steady, high quality camera path which avoids distractions and improves scene composition, at the same time as maximizing final scene coverage.

We have conducted a user study to analyze the perceptual quality of our results, by comparing them to (i) the input video, (ii) the results of a stabilization approach, and (iii) the results of a reduced version of our approach which omits the second optimization

- *Fang-Lue Zhang is with TNList, Dept. Computer Science, Tsinghua University, FIT 3-524, 100084.*
  http://cg.cs.tsinghua.edu.cn/people/ fanglue/
- *Fang-Lue Zhang, Han Zhao and Shi-Min Hu are with Tsinghua University. Jue Wang is with Adobe Research. Ralph R. Martin is with Cardiff University. Shi-Min Hu is the corresponding author.*

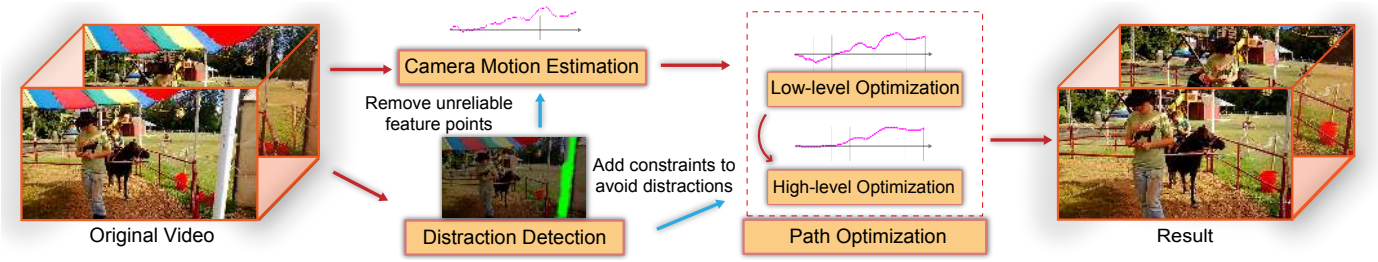


Fig. 1. Algorithm overview.

pass. This study showed that our method significantly improves the aesthetic quality of video, and produces better results with higher quality camera paths than either stabilization or low-level optimization alone.

2 Related Work

We now briefly review related methods concerning appearance enhancement and camera path improvement.

Appearance enhancement. Video enhancement methods mainly focus on manipulating existing visual content in the video, without paying attention to the camera motion. Algorithms have been proposed for several tasks. Wexler et al. [5] and Stengel et al. [6] give a video completion method to remove undesirable objects in video by texture synthesis, but do not suggest how to detect distracting objects to remove. Many low-level methods exist, such as the one in [7] [8], which refines blurry frames of input video using a patch-based method, but they cannot improve video quality in terms of composition, unlike our method. Other work on video stylization [9] and abstraction [10] also provide enhanced appearance, but such methods just keep the structure of the original video, and do not improve the planning of the video.

Video stabilization. A second category of techniques focuses solely on camera path recovery and re-planning for visual quality improvement; our work belongs to this category. Video stabilization concerns removal of high-frequency motions typical of hand-held cameras, and has been extensively studied. Early approaches used temporal filters to smooth the 2D motion recovered from the trajectories of feature points [11]; inpainting is often needed to fill unrecorded content due to changes in the camera path [12]. Robust feature tracking can improve stabilization results [13]. Recent approaches, such as subspace stabilization [1], L^1-optimization [2], and bundled camera path optimization [3] can handle complex cases involving significant scene parallax and rolling shutter effects. However, these methods solely smooth the camera path to remove high-frequency camera jitter. They do not consider removing undesirable lower-frequency motion components of the original camera path such as up-and-down motions caused by the cameraman walking.

Re-cinematography. To improve the camera path beyond low-pass filtering, Gleicher et al. [4] fit camera motion models of the kinds used in professional cinematography to camera paths extracted from amateur video. Liu et al. explicitly reconstruct and optimization the camera path in the 3D space. Our goals are similar in terms of path optimization, but their approach is not flexible enough to allow other additional constraints, such as avoiding distracting objects. Grundmann et al. [2] use an L^1-optimization framework to compute a smoothed camera path composed of steady segments with smooth in-between transitions. Their focus is still on stabilization. They do not analyze the visual content and quality of the resulting paths, which is our main concern. Preserving visually salient content in the original video has been given limited consideration in previous stabilization methods [4], [14], and these approaches do not consider distraction detection and removal, a further goal of our method.

Video aesthetics. We aim not just to stabilize the video, but also to improve its aesthetic qualities by removing distractions and undesirable low-frequency motions. Salient object detection is a basic step for many object-level aesthetics improvement approaches, pioneered by the work of Itti et al. [15]. Image saliency methods use either heuristic methods [16], [17] or learned models [18] to predict the objects in an input image that can potentially draw viewers' attention. One of the representative methods is developed by Cheng et al. [19]. These methods are extended to videos by incorporating additional features such as motion, flickering, optical flow and spatio-temporal interest points [20][21]. However they only focus on detecting visually important regions, but do not evaluate their aesthetics. In other work on aesthetic improvement of video, Luo et al. [22] proposed an aesthetic quality measure for images and video based on spatial composition. Yeh et al. [23] proposed an evaluation method for temporal aesthetic quality that considers the directions, magnitudes and positions of object motions in video. These methods, however, only measure visual quality, without providing methods for improving it. Xiang and Kankanhalli [24] optimized visual quality by improving the motion of the foreground object by re-projecting the good motions to frames with low motion quality. This method just focuses on foreground objects and does not give good camera paths for dynamic scenes. Berthouzoz et al. [25] provided tools for placing cuts and transitions at appropriate positions in interview videos. These methods require stabilized videos, and aim to preserve as much content possible. In contrast, our method selects content to avoid distractions. More recently, Arev et al. [26] presented a system to generate a single video of a scene from multiple videos captured by different cameras. Our method aims to improve the visual quality of a single input video.

3 Overview

The improvements to the output video should help to keep the viewer's attention on the main subject which the videographer intended to capture. In addition to removing unwanted distractions, and stabilizing the video, the camera path should thus follow

some basic rules of cinematography—for example, camera motion should generally be monotonic, and not oscillate unexpectedly. Also, a simple camera motion should be used for each separate segment of the video.

We use a sequence of transformation matrices to encode the changes between each adjacent pair of frames in a video as a proxy for the camera motion. The camera motion can then be described by a sequence of parameters which are the elements of these transformation matrices. Elementary cinematographic camera operations such as pushing in and pulling out, panning, tilting, and staying correspond to segments with zero, constant or smoothly changing values of these parameters (see Fig. 1). The overall objective is to optimize the path represented as a matrix sequence so that it comprises simple segments of the above kind while satisfying additional constraints, particularly to avoid distractions.

As it is easier to analyze and segment a stabilized camera path rather than the original unstable input video, we use a two-pass optimization framework, as shown in Fig. 1. The aim of the first pass is to find an initial camera path which avoids any distractions, and at the same time is stabilized with respect to high frequency jitter. The aim of the second pass is to then ensure that each motion segment has a simple model, while respecting any constraints generated during the first pass. In detail, we firstly detect any objects which may distract users, and determine hard constraints to ensure that the output frames avoid these objects, while being contained entirely within the input frames. An L^1-optimization framework is then used to generate an initial path in which the parameters representing the camera path are simple functions of time. In the high-level pass, the zoom, rotation angle and translation are then analysed to split the camera path into segments, and a piecewise linear model is fitted to each of these quantities, after eliminating any unnatural motions over short time scales, such as the camera moving up immediately followed by moving down. Finally, L^1-optimization gives the output frames, again using the same hard constraints.

4 Distraction Detection

4.1 Principles

Distractions are objects that attract the viewer's attention away from the main subject. To remove them from the output, we must first detect them. Distractions typically have the following properties:

High saliency The visual saliency of distractions is usually significantly higher than that of their surroundings (which is why they are noticeable). Video saliency is related to both appearance and motion. As in still images, regions with high color or texture contrast to adjacent regions have high *appearance saliency*. More importantly, objects whose motions differ significantly from those of nearby regions have high *motion saliency*. Both kinds of saliency are significant when determining if a region contains a distraction: a moving object with low appearance saliency is less noticeable.

Off-center location and short duration Amateurs typically try to keep the main object near the center of each frame when shooting a video, whereas distractions often appear near edges. Furthermore, they often are only present for a short time. They may arise either due to camera motion, or the distraction's own motion.

We automatically detect distractions in an input video by using these properties. To determine the presence of distractions by tracking local regions in video, we use temporal super-pixels (TSP) [27], which provide good spatial localisation and have good temporal stability. For each TSP in each frame, we compute its local appearance contrast and motion contrast relative to adjacent regions to produce a time-dependent saliency value. Distractions are identified by considering region saliency, spatial location, and temporal duration, as we now explain.

4.2 Computing video saliency

A TSP is a set of contiguous video pixels with similar color and motion parameters and can be found using the method in [27]; TSPs do not overlap. We define the set of all TSPs in the video as Φ. The i-th TSP is denoted $\Phi_i = \Phi_i^s, \Phi_i^{s+1}, \ldots, \Phi_i^{s+n-1}$, where s is the first frame in which this TSP appears, and n is the number of frames for which it lasts. Φ_i^j comprises the pixels that the i-th TSP covers in frame j.

For each TSP, we compute its saliency for each frame in which it exists, as its saliency may change over its lifespan. For example, a dog sitting near the main subject in a video can be static for a while before starting to move around. The viewer's attention may not be distracted at first, but may be drawn away when the dog starts moving.

Saliency is determined by local appearance contrast and motion contrast. We use the RContrast [19] saliency detection method to compute the appearance saliency value $S_C(\Phi_k^j)$ for Φ_k^j, the region covered by TSP Φ_k in frame j. We use this method as it is suited to calculating saliency for small regions.

For motion saliency, we compare the mean optical flow in Φ_k^j to that of nearby regions:

$$S_M(\Phi_k^j) = |F(\Phi_k^j) - \frac{1}{L} \sum_{\Phi_l^j \in N_k^j} F(\Phi_l^j)|, \tag{1}$$

where

$$F(\Phi_k^j) = \frac{1}{N} \sum_{p \in \Phi_k^j}^{N} f_p^j.$$

and f_p^j is the optical flow vector of pixel p in frame j, computed using Sun et al.'s approach [28]. $N_k(t)$ is the neighborhood region set which contains all TSP regions whose centroids are closer to the centroid of $\Phi_k(t))$ than a threshold τ (set to 0.3 in normalized coordinates in our implementation).

Combining these two terms, the video saliency of a region is:

$$S(\Phi_k(j)) = S_C(\Phi_k(j)) + \alpha S_M(\Phi_k(j)), \tag{2}$$

where α controls the relative importance of visual and motion saliency. We consider the latter to be more important, so set $\alpha = 0.75$. Finally, saliency values are normalized to $[0, 1]$ relative to the maximum saliency for each frame. Various saliency maps calculated by our method are shown in Fig. 2(b).

Fig. 2. Video saliency maps for selected frames: distractions are shown as green masks.

4.3 Distraction labeling

After computing saliency values for each TSP in each frame, we now label regions with high saliency values and which lie close to the frame border as potential distractions D_P:

$$D_P(\Phi_i(j)) = (S(\Phi_i(j)) \geq T) \cap (P(\Phi_i(j)) \in \Omega), \quad (3)$$

where T is a threshold, and Ω is the border area of each frame outside the rectangle w_-, h_-, w_+, h_+. In our implementation, we set $T = 0.4$, and $\{w_-, h_-, w_+, h_+\} = \{0.2w, 0.1h, 0.8w, 0.9h\}$, where w, h are the width and height of the frame.

We next count how many times a TSP is labeled as a potential distraction during its lifespan, and determine the length of its lifetime. If the following conditions are all met, we treat its whole lifespan as a distraction:

- the proportion of its frames in which it is labelled as a distraction is greater than τ,
- its first or last frame lies within the border region Ω_t,
- its duration is shorter than a threshold D.

Our implementation sets $\tau = 0.5$ and $D = 3$ seconds. To avoid missing neighboring TSPs which belong to the same distracting objects, we propagate distraction labels to neighboring TSPs with similar motion vector and mean color. Examples are shown in Fig. 2(c). To exclude these distractions in the output video, we add constraints controlling cropping to the motion path.

Implementation details We use publicly available source code[1] for TSP extraction, using the default parameters, while optical flows are calculated using the method in [28]. We downsample the video to 320×240 when finding distractions, to accelerate computation. After finding the locations of the distractions, they are up-sampled and used to process the full-resolution video.

1. http://people.csail.mit.edu/jchang7/

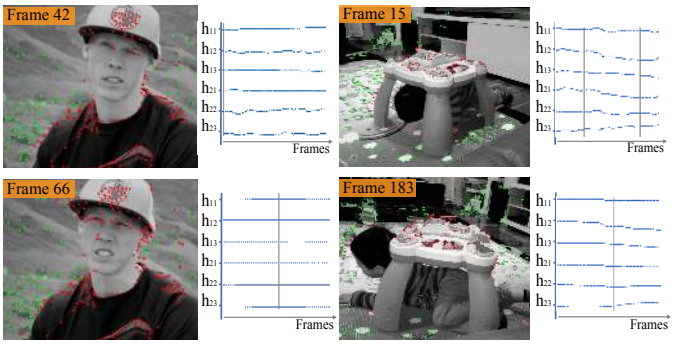


Fig. 3. Feature selection for motion estimation. Successfully tracked feature points are shown on the frames. Feature points with higher saliency values are shown in red, and are not used in motion estimation. Estimated motion transformation parameters using the method in [2] are shown beside the frames: (top) using all feature points, (bottom) without the blue feature points.

5 Camera Path Optimization

Given the constraints to remove the distractions, we can now generate a desired camera path in terms of the transformation matrix sequence relating adjacent frames. The overall objective is to find a set of update matrices relating pairs of adjacent frames. By finding suitable smoothly varying parameters for these matrices, the output video will have a smooth path made of elementary segments, like one used in cinematography.

Our computations are performed in the 2D image plane. The output window has fixed dimensions $W_c = \{0, 0, w, h\}$ in each frame; pixels from the original frames are used to fill this window. We optimize the transformation matrices relating each original frame in this plane, so that after transformation, the content shown in W_c varies smoothly and monotonically over the medium term, and satisfies the constraints needed to avoid distractions.

5.1 Original path estimation

Before optimizing the camera path, we must first recover the original scene's motion parameters. We adopt the discretized piecewise linear camera motion representation that has been extensively used in previous approaches [2], [4]. Specifically, an input video is a sequence of images $\{I_0, \ldots, I_n\}$. An affine transform matrix H^{t+1} relates each successive pair of frames via $I^{t+1} = H^{t+1} I^t$. A proxy for the camera path can thus be represented by the sequence of matrices $H^1, \ldots, H^n$. These transformations can be concatenated so that:

$$I^t = H^t \ldots H^1 I^0. \quad (4)$$

To efficiently estimate H^t for each frame pair, we detect sparse Harris corner feature points and track them using an implementation[2] of the Kanade-Lucas-Tomasi feature tracker [29].

A common approach to estimating H^t uses RANSAC to exclude unreliable feature points: see for example [2]. This approach works well for static scenes, but often fails when large moving objects present in the scene. In such cases, a large number of feature points may belong to dynamic foreground objects and cannot be completely removed by RANSAC. The remaining foreground feature points may cause serious problems when estimating the

2. http://www.ces.clemson.edu/stb/klt/

camera motion model, as shown in Fig. 3 and the supplementary material.

Our approach uses the saliency detection results already computed to avoid this problem. Feature points whose motion saliency values are higher than a threshold belong to dynamic objects, and so are directly excluded before applying the RANSAC process. This improves the robustness and accuracy of H^t estimation. As Fig. 3 shows, using all tracked feature points including ones from the moving foreground person results in noisy motion parameters. Using the restricted set of features provides a much more stable result with better parameter estimates.

Note that this method is not limited to videos captured by static cameras: it also works for some dynamic scenes captured by moving cameras such as the example shown on the right in Fig. 3. In this case, the TSPs belonging to the moving object have larger motion saliency according to Eqn. (1). This is because they have a different motion direction and speed relative to the background TSPs nearby, while the background TSPs have a relatively coherent motion with respect to their neighbors. Unfortunately, this simple strategy does not always work. When the foreground object is very large nd its parts have similar motion (e.g. a large bus drives past in front of the camera), the saliency of the TSPs belonging to the moving foreground will be lower. The method will then fail to estimate the correct camera motion, as the background feature points will be excluded due to their high saliency. High-level semantic scene understanding is probably necessary to correctly handle cases of this kind.

5.2 Two-pass optimization

Our expectations for the improved camera path are twofold. At a low level, we expect the new path to be smooth and stable; it should also avoid distractions while keeping the main objects in the frame. At a high level, we expect the camera motion to comprise a series of smooth, monotonic movements like those performed by a professional videographer, such as pull out, push in, panning, etc. [30], [31]. We use a two-pass optimization framework to meet these expectations at both levels.

We first apply L^1 optimization to achieve our low-level goals, including stabilization and scene recomposition. A similar approach has already been used for stabilization [2], and we extend it to include multiple objectives.

Without imposing higher level constraints, the new camera path generated by this pass often contains visually contradictory elements. Consider Fig. 4. To avoid the distracting white pole on the right, the optimized path includes a counterclockwise rotation followed immediately by a clockwise rotation (Fig. 4(b)), which looks poor. Our second optimization pass produces a final camera path that avoids such oscillations, giving a path composed of more natural and professional-looking motion segments.

5.2.1 Low-level optimization

For the low-level pass, from the original video, we want to produce a camera path composed of a series segments that avoid distractions while keeping as much significant content as possible. The objective of this stage is thus a smooth path with the hard constraints that the distractions should lie outside the output window W_c, and soft constraints that as much original content should be retained as possible. Given the original camera path $\{H^t\}$ based on a full affine transformation model, we seek to find an update transformation sequence $\{P^t\}$. In the result, each original video frame is now transformed by the updated proxy camera path $\{H'^t\} = \{P^t H^t\}$ and cropped to the cropping window. The video content remaining satisfies various constraints, as illustrated in Fig. 5.

Following the approach in [2], to achieve a smooth and stable path, we aim to minimize the first, second and third order derivatives of the resulting sequence $\{H'^t\}$, which can be measured using *residual motion* $\triangle^t$:

$$\begin{aligned} \triangle_1^t &= P^{t+1}H^{t+1} - H^t, \\ \triangle_2^t &= \triangle_1^{t+1} - \triangle_1^t, \\ \triangle_3^t &= \triangle_2^{t+1} - \triangle_2^t. \end{aligned} \tag{5}$$

We also wish to completely avoid TSPs that are marked as distractions. We treat these as hard constraints: in frame t, after applying the update transform P^t, the position of a distracting TSP should lie outside the cropping window W_c. For speed, we enforce this using the bounding box of each TSP rather than the TSP itself. Because distracting TSPs are usually located near the frame border, if the corners of the bounding box are all outside W_c after transformation, we may assume the whole box will be outside W_c. We thus only need to record these points as C_k for the k-th distracting TSP.

The L^1 optimization framework expresses all constraints concerning inclusion and exclusion of points p^t as inequalities of the form:

$$(x_{\min}, y_{\min})^T \leq P^t p^t \leq (x_{\max}, y_{\max})^T.$$

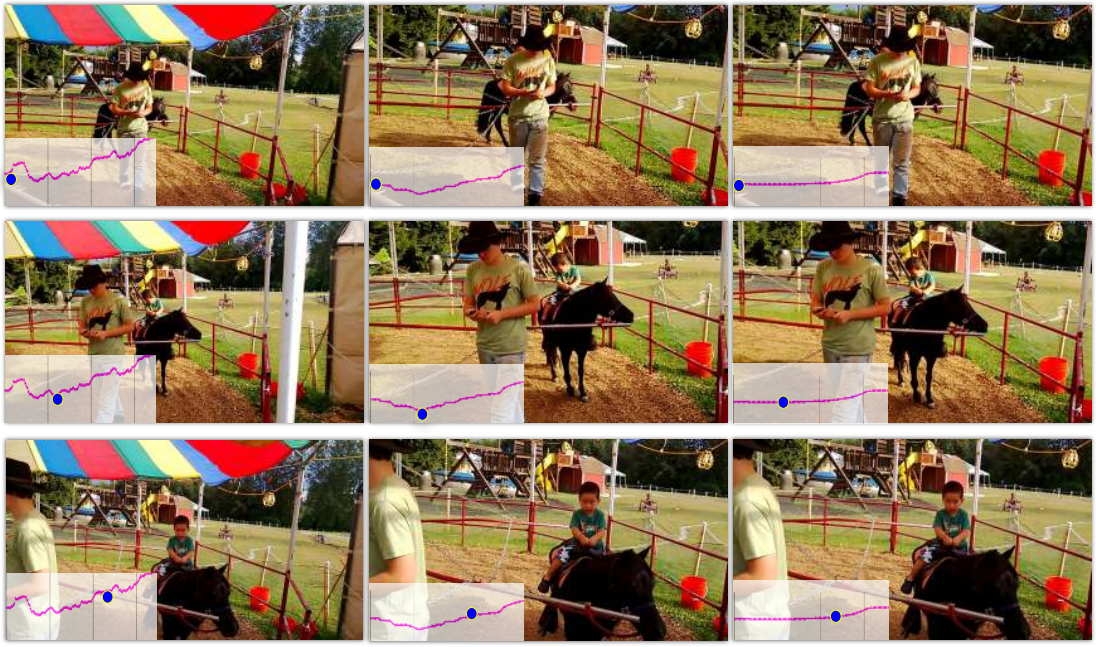
If a constraint is one-sided, bounds may be infinite: e.g. if the x-value should be smaller than zero, then x_min is set to negative infinity. Consider the distracting TSPs in the left-bottom region of Fig. 5 as an example. To ensure those located closer to the vertical boundary of the cropped frame (shown in orange), are removed, we must ensure that the x coordinates of all 4 corners of their bounding boxes satisfy $C'_k(x) = (P^t C_k)(x) < 0$. Similarly, those that are closer to the horizontal boundary (shown in green) must satisfy $C'_k(y) < 0$. The constraints for other distracting TSPs in other regions can be set in a similar way. Compared to restricting both $C'_k(x) < 0$ and $C'_k(y) < 0$ in this case, our one-variable constraint is looser, thus allowing more original content to be preserved in the final video.

The other constraint is inclusion of the main target object in the final video. Following [2], to make sure the cropping window lives inside the original video frames, we constrain the transformed corners of the original frames to lie outside the cropping window. For instance, the top-left corner c_{tl}^t of frame t must satisfy $P^t c_{tl}^t \leq (0,0)^T$.

Assuming that the most salient non-distraction region is likely to be the main subject of the video, we wish to ensure that it appears in the cropping window. We thus add inequality constraints for the corners of its bounding box b_i^t:

$$(0,0)^T \leq P^t b_i^t \leq (w,h)^T.$$

Given that the cropping window W_c is fixed, the content coverage of the final video is controlled largely by the scaling terms in P^ts.



(a) Original sequence (b) Low-level optimized (c) High-level optimized

Fig. 4. Two-pass optimization. The pink curves show rotation of the scene relative to the first frame, in the original sequence, after low-level optimization, and after high-level optimization.

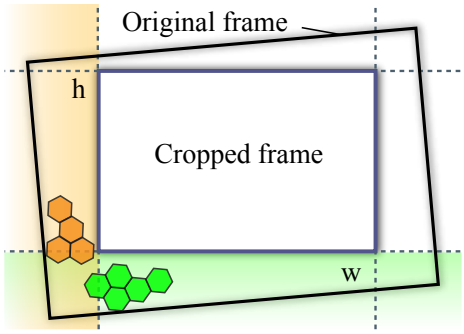


Fig. 5. Removing distracting TSPs by setting constraints in the L^1 optimization framework. For orange TSPs, x coordinates are constrained; for green TSPs, the y coordinates are constrained.

To maximize coverage in the output video, extra terms are added to the optimization objective based on the distances between the updated frame corners and the original ones. For each corner c_i^t, we introduce two slack variables to be minimized, $\sigma_{ix}^t, \sigma_{iy}^t$ via the inequalities:

$$P^t c_i^t - c_i^t \leq (\sigma_{ix}^t, \sigma_{iy}^t)^T$$

or, depending on the location of the corner,

$$P^t c_i^t - c_i^t \geq (\sigma_{ix}^t, \sigma_{iy}^t)^T.$$

Finally, slack variables are introduced as the bounds of the residual motion values in Eqn. (5):

$$-S_i^t < \triangle_i^t < S_i^t, \tag{6}$$

where S_i^t and $\triangle_i^t$ are both matrices containing the same number of entries as the transformation matrix H^t.

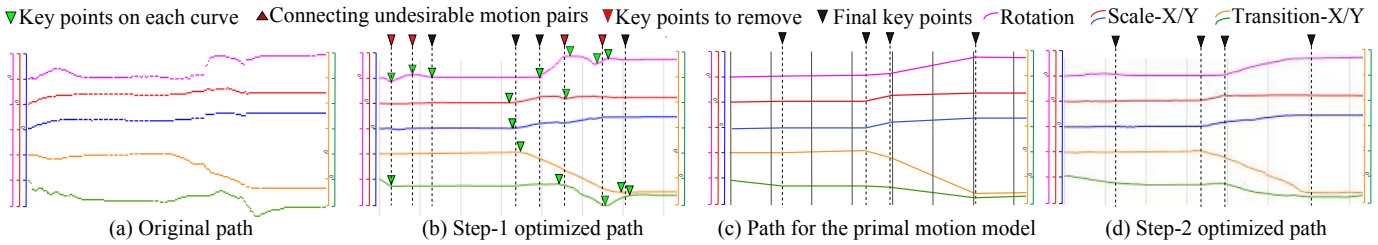
The overall optimization objective is to minimize a weighted sum of the slack variables contained in $\{S_i^t, \sigma_i^t\}$, constrained by the above inequalities:

$$E = \arg\min_s W^T s \quad \text{subject to} \quad P^0, \ldots, P^t, \tag{7}$$

where s represents the vector formed by the slack variables in $\{S_i^t, \sigma_i^t\}$, and W contains the weights for each slack variable. As in [2], we set the weights for slack variables of parameters related to scaling and rotation to 50 times those of translation parameters because an equal amount of change to the former parameters will cause much larger variations than changes to the latter. The default weights for σ_i^t are equal to those of the translation parameters. This problem can be effectively solved using linear programming. Finding the minimum value for the weighted sum of the slack variables gives the optimal P^t.

5.2.2 *High-level path refinement*

Low-level optimization is performed directly in the space of transformation parameters when determining P^t. Since the motion components are not determined by a single parameter, the camera path produced by the initial optimization can only satisfy low-level constraints on the original path, but cannot guarantee high perceptual quality. The second pass of optimization further refines the camera path so that it is composed of commonly used cinematographic camera motions such as panning, zooming, push-in and pull-out. At the same time, we remove unreasonable combinations of motion segments, such as panning one way and then immediately panning the opposite way. To do this, we first analyse the initially optimized camera transformation matrix sequence by decomposing it into its motion components of scaling, rotation and translation. In the motion component space (see e.g. Fig. 6), we can clearly see any undesirable motion segments such



(a) Original path (b) Step-1 optimized path (c) Path for the primal motion model (d) Step-2 optimized path

Fig. 6. Camera paths shown as motion parameter functions. Low-level optimization of the original path (a) yields the modified path (b). Key points in this path are detected and filtered. The remaining key points divide the path into segments. A simple motion model is fitted to each segment (c), and used as a reference in the high-level optimization pass, together with constraints, to produce the final result (d).

as moving left then immediately right. We detect segments (as explained later) and fitting a new motion curve representing a simple motion to each, combining contradictory adjacent segments where necessary, to give an output based on simple smooth movements. Unfortunately, doing so does not always satisfy the constraints previously determined, concerning inclusion of the cropping window in the source, and avoiding distractions. We overcome this problem by using the desired path to guide another optimization pass to provide the final camera path.

A full affine transform H:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{pmatrix}, \tag{8}$$

may be decomposed into scaling, rotation, translation and skewing. Although skewing will be very small after low-level optimization between frames, we still model it for accuracy. The components are:

$$M_x = \sqrt{h_{11}^2 + h_{12}^2}, \quad M_y = \frac{h_{11}h_{22} - h_{12}h_{21}}{M_x},$$
$$T_x = h_{13}, \quad T_y = h_{23},$$
$$\theta = \text{atan}(h_{12}/h_{11}),$$
$$S = \frac{h_{11}h_{21} + h_{12}h_{22}}{h_{11}h_{22} - h_{12}h_{21}},$$

where M_x and M_y are scaling coefficients in x and y directions, (T_x, T_y) is the translation, θ is the rotation angle, and S is the skew. Using Eqn. (4), we can compute the accumulated transform H'^t from frame 0 to frame t as follows. Firstly we apply the update matrix P^t from Eqn. (7) to all tracked feature points, then estimate H'^t between frame 0 and frame t using the method in Section 5.1. We then decompose H'^t into rotation, translation and scaling components. These component values $M_x(t)$, $M_y(t)$, $\theta(t)$, $T_x(t)$, $T_y(t)$, $S(t)$, are varying functions of time, as shown in the example in Fig. 6. After the first optimization pass, shearing is close to zero, so we do not consider it further.

We now explain certain steps in further detail:

Resolving motion conflict The low-level optimization produces high quality motion segments, but can produce aesthetically undesirable results: consecutive segments can have opposing motions, e.g. zoom-in immediately followed by zoom-out, or panning left immediately followed by panning right. To eliminate such cases, for each motion component function f, we first remove noise using a low-pass filter. We then find the key points where the first order derivatives change sign, or become zero, or stop being zero, which indicate the changes of the motion status. We merge the neighboring key points in all motion component function if they are too close in time (set to 6 frames in our experiments), see the top row of Fig. 6(b). We record all key points extracted from different motion descriptors in a single chronological sequence.

To eliminate consecutive opposing motions, we first identify key points that connect such pairs of motions. As shown in Fig. 6(b), since two opposing motions tend to cancel out each other, the overall motion change after such a pair of motion segments is close to zero. Thus, on the derivative curve $\triangle f(x)$, the sum of the values should be zero for such segments. We thus use a box filter on the derivative curve to detect them:

$$F(x) = B(x) * \triangle f(x),$$

where

$$B(x) = \begin{cases} 1 & -r \le x \le r, \\ 0 & \text{otherwise;} \end{cases} \tag{9}$$

r controls the temporal span of the filter ($r = 15$ frames by default). If $F(x_0) = 0$ and the values of $f(x)$ are not all zero, any key points closer than r are removed. We replace the function between the neighboring key points on either side by a linear segment connecting them on each curve. We then iteratively perform this filtering process until no further key points can be removed. Fig. 6(b) shows examples of key points removed due to opposing motions. The remaining key point set is denoted $\mathcal{Q} = \{q_k\}, k = 1, \ldots, m$.

Fitting the motion model We wish to represent the output video using a set of standard camera motions commonly used in cinematography: (i) *zoom-in* and *zoom-out*, simultaneous scaling in x and y, (ii) *push-in* and *pull-out*, combinations of scaling and translation in one direction, (iii) *panning* and *tilting*, which can be approximated as horizontal and vertical translations, but if the main scene is not parallel to the picture plane, there will also be scaling. To achieve smooth motion between each pair of adjacent key points, we fit a piecewise linear model to the motion component functions taking q_k as the split points. Formally, taking the curve $\theta(t)$ as an example, denoting the segment between q_k and q_{k+1} ($q_k \in \mathcal{Q}$) as L_k, we fit a linear function $a^{L_k}x_k + b^{L_k}$ when $x_k \in L_k$, which is continuous with the adjacent function at the intersection point q_k.

Our overall objective is to solve the following minimisation problem:

$$\min \textstyle\sum_{k=1}^{m} \parallel a^{L_k}x_k + b^{L_k} - \theta(x_k) \parallel$$
$$\text{such that } \; a^{L_k}q_k + b^{L_k} = a^{L_{k+1}}q_k + b^{L_{k+1}}. \tag{10}$$

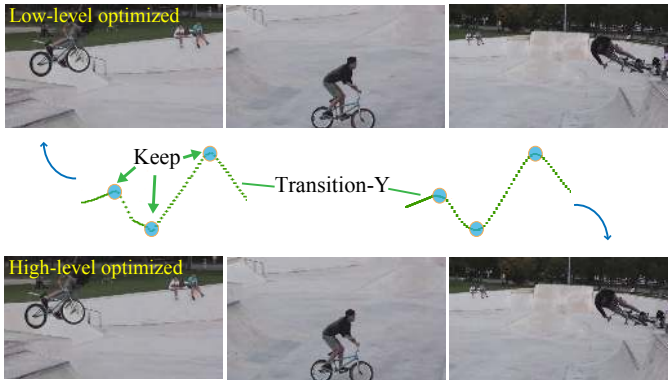


Fig. 7. A video in which the camera follows the main object. Main object position detection ensures we retain all important key points.

We must further constrain the fitting problem to ensure that $M_x(t)$ and $M_y(t)$ are identical, to avoid distortion. To do so, we just replace the objective of each frame by:

$$\| a^{L_k} x_k + b^{L_k} - M_x(x_k) \| + \| a^{L_k} x_k + b^{L_k} - M_y(x_k) \| .$$

The resulting model is denoted f' and an example is illustrated in Fig. 6(c).

Final optimization This model now contains high quality motion segments, but cannot be directly used as the final camera path for two reasons. Firstly, as the camera path has been modified, it may no longer exclude distractions from all frames, nor can it guarantee that the cropping frame remains within the original video. Secondly, only first order continuity is enforced between motion segments, but higher order continuity is desirable. To address these issues, using the above model as a reference, we perform L^1 optimization again. We again include all the hard constraints from the initial optimization pass, but change the optimization objective to be that the final camera path is close to the desired smoothed path.

Specifically, the transformation matrix H'^t from frame 0 to frame t is calculated from the fitted parameter curves in Eqn. (10). Let the final camera motion from frame 0 to frame t be H''^t. To make H''^t similar to H'^t, we introduce a new set of slack variables S_R^t which bounds the differences of the matrix elements relating them:

$$-S_R^t < H_R''^t - H'^t < S_R^t,$$

where

$$H_R''^t = P_R^t H''^t - P_R^0,$$

and P_R^t is the update matrix to be computed for each frame. We add the new slack variables in S_t^R to the slack variable set in Eqn. (7) to form a new vector s^h, and use it in the new optimization objective:

$$E = \arg\min_{s_h} W_h^T s_h \quad \text{subject to} \quad P^0, \ldots, P^t, \qquad (11)$$

where W_h includes the weights W in Eqn. (5) and the weights for the new slack variables. The weights for parameters related to scaling, rotation and translation for the new slack variables are set in the same way as the corresponding original slack variables in the low-level optimization pass. Linear programming is again used to produce the parameters of the final update transform P^Rs, as shown in Fig. 6(d). This lead to a new frame update transformation matrix sequence used to transform all frames to the cropping window W_c, giving the final output video.

Special case—subject tracking The high-level optimization is designed to remove oscillatory motion segments. However, not all such motions are undesirable, especially if the camera is trying to follow the main subject. Consider the example in Fig. 7. The camera moves down and then immediately up to follow the fast moving biker, which is an appropriate camera path in this case. To ensure that such cases are handled properly in videos with fast moving backgrounds, we further check whether the most salient objects stay near the frame center. If the average background optical flow magnitude over a 20-frame window is larger than 10 pixels for some frames, we keep any key points belonging to such frames to avoid the background motion being smoothed out. Results for this example can be seen in the supplementary video.

6 Results

Our method transforms video inputs captured by amateur videographers into video outputs with high-quality camera paths and fewer distractions, as we now show.

In our experiments, we tested the ability of the method to detect various commonplace distractions and remove them from the final video. We also considered how well our two-level optimization avoids unnatural camera paths which might otherwise be caused by avoiding distractions or low-level stabilization. We further carried out a user study to assess whether our method can improve the visual quality of amateur video, and whether its results are better than those provided by stabilization alone.

6.1 Performance

We implemented our method in C in a single thread on a PC with a 2.5 GHz 8-core Xeon CPU and 16 GB memory. On average, distraction detection takes 3.5 s per frame, including 3.1 s for TSP extraction and 0.025 s for optical flow computation. Each optimization pass takes 0.1–0.2 s per frame, depending on the number of constraints. The speed of the algorithm could be readily improved in various straightforward ways. Firstly, as the TSP implementation is the bottleneck, a parallelized version could make the whole algorithm significantly faster. Secondly, temporal downsampling could be applied without significantly affecting the output quality too—the locations of distractions do not have to be accurately determined to exclude them from the video, and a conservative bounding box could be used.

6.2 Experiments

Distraction detection We conducted an experiment to determine how well our distraction detection method works. We downloaded 10 amateur videos from the Internet which were associated with comments that they contained distracting or annoying objects or people. To provide ground truth, we then manually labeled the distractions, by sampling the video every 10 frames and manually marking the distraction regions. After dividing the video into TSPs, any in these marked regions were taken as ground truth distractions. We then carried out distraction detection as described in Section 4, using the default settings to automatically label the

Fig. 8. Automatically detected distractions (shown as green masks), and final output frames avoiding them. In (c) the orange regions are undetected distractions. In (d) the red regions are incorrectly detected as distractions.

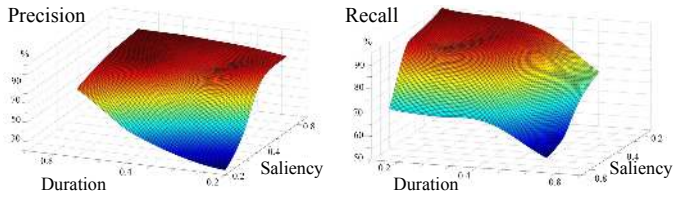


Fig. 9. Variation of precision and recall with different parameter settings.

distractions. The number of false positive and false negative labels on the TSPs gave a recall rate of 87% and precision rate of 75%. Increasing the saliency threshold T and threshold time τ gave a higher precision and a lower recall; variations in recall and precision with different parameter settings are shown in Fig. 9.

In examples like Fig. 8(a), moving people are highly salient, but are irrelevant to the main video content. Our method can successfully label them as distractions and avoid them in the final results. False positives—regions incorrectly marked as distractions—are mostly TSPs on the background, adjacent to real distractions (see Fig. 8(d)). If the background is fairly constant, one interpretation of the video is that parts of the background are moving along with the foreground: this is indistinguishable from a smaller foreground object moving against a static background. False negatives—distractions which are not detected—are usually TSPs which belong to objects moving slowly relative to the background (seeFig. 8(c). False positives are reltively harmless, as they simply cause a little overcropping. False negatives are more problematic as they result in failure to remove some distractions. Thus, we set default parameters to prefer high recall performance, ensuring that we can effectively detect and avoid most distractions.

Two-pass path optimization To test whether the high-level refinement pass improves the path as intended, we performed two-pass optimization on the same 10 videos, including the constraints to avoid distractions, and considered whether we effectively decreased the number of contradictory motions (see Section 5.2.2). In the 10 videos, our method detected 34 contradictory motions in various motion components. After optimization, only 5 contradictory motions remained, and most were removed, as shown in Fig. 10(a). The main reason that the others were not removed is that, on the one hand there is a goal to keep as much content as possible, and on the other, those motions are the only way to satisfy the constraints determining distraction removal and inclusion of the cropping window in the original frame. Such cases typically have the frame edges close to the cropping window edges. An example is shown in Fig. 10(b), where the successive rotation up-and-down is detected, but the final optimization failed to remove it because, for the middle frame, the upper edge of the cropping window is already close to the original frame edge. In the left frame, a salient object is also close to to the right edge of the cropping window.

6.3 User study

To verify whether our algorithm has the desired effect of subjectively improving the aesthetic quality of a video, we designed a user study. Its objectives were to determine:

- whether our method can generate video result with better aesthetic quality than simpler alternatives, such as stabilization only and stabilization followed by cropping;
- whether distraction detection and removal can improve visual quality;
- how the low-level and high-level path optimization steps affect visual quality.

As a basis for comparison with stabilization alone, we chose two widely-used commercial stabilization solutions: (1) the stabilizer currently used in YouTube[3], a refined version of the method introduced in [2]; (2) the subspace stabilizer in Adobe After Effects, which is based on Liu et al.'s work [1]. We also compare with a straightforward sequential approach for achieving both

3. https://www.youtube.com/editor

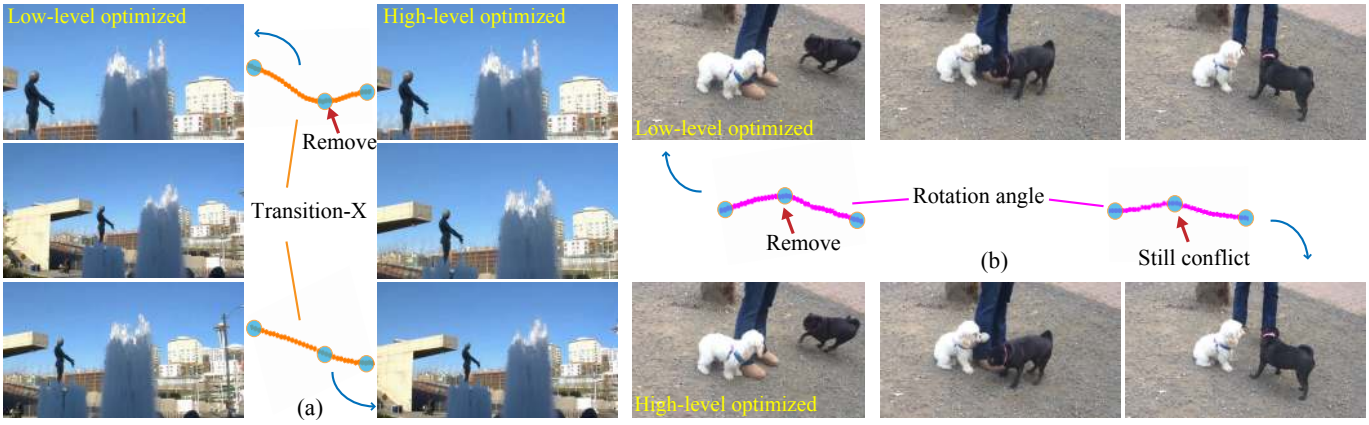


Fig. 10. Two-level optimization. (a) An undesirable path resulting from low-level optimization is successfully corrected in the final result. (b) Consecutive rotation anticlockwise then clockwise is only partially corrected by high-level optimization.

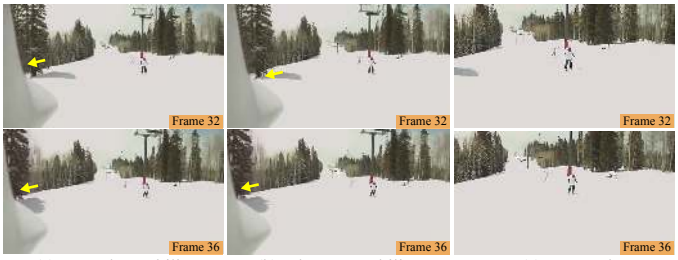


Fig. 11. Comparing our method with video stabilization methods. (a)(b) The results of the YouTube stabilizer and Subspace stabilizer in Adobe After Effects [1] both contain a distraction object (highlighted by yellow arrows) that is visually jittering. (c) Our approach naturally avoids this problem by removing the distraction.

stable camera path and distraction removal: we first apply video stabilization to smooth the camera path, then apply our distraction detection and removal method to produce the final video.

For the study, we prepared six versions of each video considered: a) the original video, b) stabilized video by YouTube, c) stabilized video by Adobe After Effects stabilizer (AE), d) stabilization followed by cropping for distraction removal, e) our intermediate result just using low-level optimization, f) our final result after high-level optimization. For each example and each participate, we showed the five derived videos in a randomized order, and asked the participant to compare each video to the original one according to the following criteria:

1) stability of the video content,
2) if there are distracting objects in the result video,
3) quality of the camera motion,
4) the severeness of content loss due to cropping.

Subjects gave an integer score between -4 and $+4$ for each question, -4 meaning much worse, and $+4$ meaning much better, than the original. The only exception is for the last question on severeness of content loss, we only allow negative scores since the original videos contain the most amount of content.

Our study used 16 amateur videos downloaded from the Internet, all of them contain some amount of distractions. They were shown to 25 participants, 15 male and 10 female, age from 20 to 30.

TABLE 1
Average quality scores in the user study. For details about t-test applied to the scores pairwise, please refer to the supplementary materials.

	Low-level only	Both passes	YouTube stabilizer	AE stabilizer	Crop after stabilized
Stability	2.06	2.94	2.13	1.89	2.43
Distraction	2.94	2.98	0.70	0.82	2.95
Camera action	1.64	2.39	1.93	1.90	1.92
Content	-0.30	-0.33	-0.10	-0.26	-1.26

They included university students, engineers and designers. The statistics of the study are shown in Table 1.

The quantitative results indicate that our method generates results with higher aesthetic quality than stabilization alone: distraction removal also improves the visual quality of the videos. We now consider each criterion in detail.

Stability The stability results are consistent with the observation that, to avoid distractions, our low-level optimization pass introduces a little jitter: the results are not as stable as the YouTube's and After Affects stabilizer's results. However, after the high-level pass, the results become well stabilized, while also having the benefits addressed in the other criteria. For example, consider Fig. 11. Distractions remain after stabilization, and their unsteady motion causes participants to rate this as an unsatisfactory stabilization result. In contrast, our method avoids this distraction and so produces a smoother-looking video.

Distraction removal Because the stabilizers do not perform distraction detection, it can only avoid distractions serendipitously when cropping the transformed frame. It is clear that our second pass preserves the distraction removal performed by the first pass, and its results are presumably considered less distracting because of smoother overall motion—for example, there will be fewer changes in content at the edges of the frame. A t-test shows that the scores of low-level optimization and two-pass optimization do not significantly differ. This is because they both remove the same distractions.

Camera action The Low-level optimization and the two stabilization methods receive relatively lower scores than our complete system, because they do not focus on how to refine the camera

Fig. 12. Comparison with Gleicher and Liu's method [2008]. (Top) Three frames from their result video, where the camera zooms out immediately after zooming in, and does not remove distractions near the frame border (indicated by the yellow arrow). (Bottom) Our result has a more natural camera path with the distractions removed.

actions and just stabilize it. Furthermore, the score for low-level optimization is slightly lower than that of both stabilizers because it introduces complex motions to avoid distractions. In comparison, our high-level optimization method produces simpler motion and avoids contradictory motions, thus achieves the highest score.

Content retention and quality In terms of the severeness of content loss, the results from cropping after stabilization received the lowest score, indicating that more important content has been cropped out by this method than others. This is because both steps apply cropping independently. We will provide more detailed comparison between our method and this simple strategy in the next section. The scores for other four methods are close, indicating no significant difference according to this criterion. We have found that in extreme cases where distractions are too large, our method could remove some other important content of the video and make video frames to be blurry by excessive zooming in. Such an example is shown in Fig. 14(b), where our final result only achieved a low average score of -1.75. We will discuss how to avoid excessive cropping in the next section.

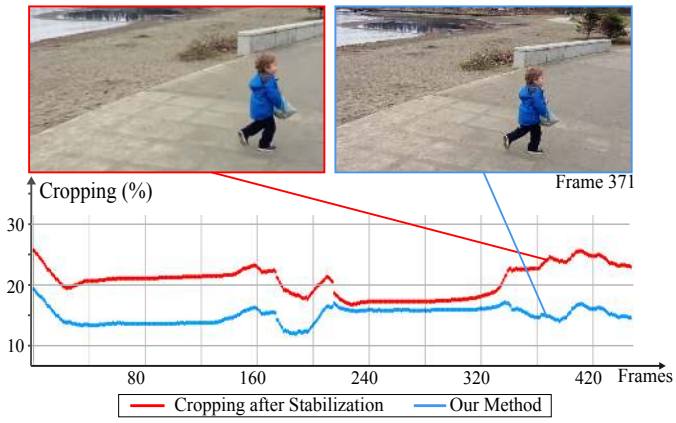
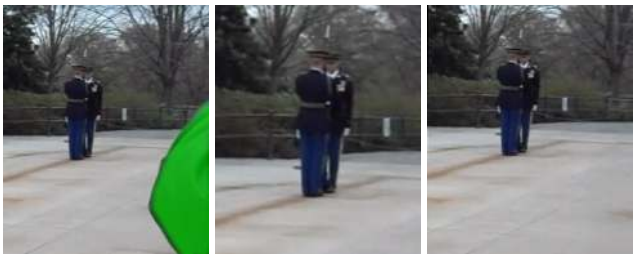


Fig. 13. Comparing our approach with a simple cropping after stabilization strategy, in terms of the percentage of cropped content on each frame. The top row shows one frame with a large cropping difference.



(a) Distractions (b) No completion (c) Partial completed

Fig. 14. Using video completion can potentially avoid too much cropping.

7 Discussion

7.1 Comparisons

Comparison to Gleicher and Liu's Method Gleicher and Liu [4] proposed a method to break a video into shorter segments with static scenes and directed motions following the rules of cinematography. Their method is based on finding the four corners for cropping windows in detected keyframes and controlling the path between them. This makes it hard to satisfy per-frame constraints (as needed to avoid distractions) except at the keyframes: see Fig. 12. Compared to this work and other video stabilization methods such as [2], [32], our output has improved aesthetic quality for several reasons. Firstly, our results avoid distractions, significantly improving visual quality in ways not considered by simple stabilization. Secondly, our camera paths avoid contradictory movements like zooming out immediately after zooming in: see Fig. 12. We also ensure that simple camera paths are used.

Comparison to cropping after stabilization Applying video stabilization and distraction cropping sequentially is a straightforward strategy to achieve both goals. However, the main issue of this approach is excessive cropping, as cropping has been done in both steps independently. In Fig. 13, we compare this strategy and our approach by plotting the amount of content cropping for each frame of the same input video, which suggests that our method can keep more original video content by simultaneously addressing both stabilization and distraction removal. In our experiments, we have found that the removed regions by cropping after stabilization strategy are usually 20%-50% larger using that of our method.

7.2 Avoiding Excessive Cropping

If a distracting object covers a large portion of a video frame, our method can lead to excessive cropping, which is unacceptable to most users when compared with the original video (an example is shown in Fig. 14(b)). Furthermore, too much cropping will also introduce blurry video frames that have low visual quality, especially when the input video is already low-res. To avoid this problem, our system can optionally apply hole filling techniques to remove distractions. Specifically, if the system detects strong zooming-in in the optimized camera path, i.e. the perimeter of the final video is smaller than 60% of the perimeter of the original video, it then applies the hole filling method proposed in [33] to remove those distractions for which corresponding background regions can be found in other frames. We then only need to exclude remaining pixels classified as distractions that cannot be completed. As shown in Fig. 14(c), by using partial background

Fig. 15. Our system cannot produce good results in the case where the distraction overlaps with the main object. (left) Original videos. (Right) Our results. The distracting object is marked in green. The guitar and the distracting pedestrian overlap for several frames, causing part of the guitar to be removed by our method.

reconstruction inside the occlusion region, our system can keep more video content and avoid excessive cropping.

In more extreme cases where hole filling also fails, we can relax the hard constraints on distraction removal to allow distractions to partially remain in the video, in order to avoid excessive cropping. Further work is needed to determine a suitable user interface.

7.3 Other Discussions and Limitations

Our approach is based on the same camera path model and optimization framework proposed in [2]. While this framework is seemingly simpler and more restrictive than more recent approaches such as the mesh homography model [3], it has several advantages over more complicated models in practice. First, it is robust and can be applied on a wide variety of examples, while more recent 3D or 2.5D approaches typically have more assumptions on the scene structure, such as the applicability of 3D reconstruction or long-range feature tracking. Secondly, this framework is computationally very efficient, while more complicated models often come with much higher computational cost. Finally, this framework is flexible enough to incorporate additional constraints, which is much harder to do with more complicated models. For all these reasons we choose this camera path representation model as a basis of our algorithm. Our evaluation results in Table 1 also show that based on this framework, our system achieves similar, if not better quality of video stabilization than more recent approaches that use more complex camera motion models.

Our method has several other limitations. Firstly, we can only avoid distractions which do not overlap the main objects. If we cannot find a cropping window which can separate the main object from distractions, our method will fail as the various constraints will conflict, as shown in Fig. 15. Secondly, our method is also limited by the global linear motion model it uses. As a recent study has shown [3], a single global motion matrix is insufficient for stabilizing certain types of video. In some cases, distractions cannot be detected automatically or reliably; user assistance may be needed to correctly identify the distractions. Fig. 8(c) shows such an example, where the orange region was added by the user.

8 Conclusions

We have presented a method to improve the visual quality of amateur video. We use a video distraction detection method and a two-pass optimization framework to provide a camera path which avoids distractions, and gives smooth and reasonable camera actions. Experiments and a user study have shown that distractions can be effectively detected, and removing them improves the aesthetic quality of video. We also significantly improve the visual quality by refining the output camera motion path. We hope in future to improve the computational efficiency as discussed in Section 6.1, and improve upon these results by using scene reconstruction methods to allow us to perform high-level path optimization and distraction avoidance in 3D space.

Acknowledgments

This work was supported by the National High Technology Research and Development Program of China (Project Number 2013AA013903), the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Number 61272226, 61120106007), Research Grant of Beijing Higher Institution Engineering Research Center, Tsinghua University Initiative Scientific Research Program, and an EPSRC travel grant.

References

[1] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Trans. Graph.*, vol. 30, no. 1, pp. 70:1–70:10, 2011.

[2] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *IEEE CVPR*, 2011, pp. 225–232.

[3] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM Trans. Graphics*, vol. 32, no. 4, pp. 78:1–78:10, 2013.

[4] M. L. Gleicher and F. Liu, "Re-cinematography: Improving the camerawork of casual video," *ACM Trans. Multimedia Computing, Communications, and Applications*, vol. 5, no. 1, pp. 2–11, 2008.

[5] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 463–476, 2007.

[6] J. Herling and W. Broll, "High-quality real-time video inpainting with pixmix," *IEEE TVCG*, vol. 20, no. 6, pp. 866–879, June 2014.

[7] S. Cho, J. Wang, and S. Lee, "Video deblurring for hand-held cameras using patch-based synthesis," *ACM Trans. Graphics*, vol. 31, no. 4, pp. 64:1–64:12, 2012.

[8] M. Stengel, P. Bauszat, M. Eisemann, E. Eisemann, and M. Magnor, "Temporal video filtering and exposure control for perceptual motion blur," *IEEE TVCG*, vol. 21, no. 5, pp. 663–671, May 2015.

[9] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez, "Intrinsic video and applications," *ACM Trans. Graphics*, vol. 33, no. 4, pp. 80:1–80:11, 2014.

[10] W. Qu, Y. Zhang, D. Wang, S. Feng, and G. Yu, "Semantic movie summarization based on string of ie-rolenets," *Computational Visual Media*, vol. 1, no. 2, 2015.

[11] A. Litvin, J. Konrad, and W. C. Karl, "Probabilistic video stabilization using kalman filtering and mosaicing," in *Electronic Imaging 2003*, 2003, pp. 663–674.

[12] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150–1163, 2006.

[13] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato, "Sift features tracking for video stabilization," in *14th Int. Conf. Image Analysis and Processing, ICIAP 2007*, 2007, pp. 825–830.

[14] B.-Y. Chen, K.-Y. Lee, W.-T. Huang, and J.-S. Lin, "Capturing intention-based full-frame video stabilization," *Computer Graphics Forum*, vol. 27, no. 7, pp. 1805–1814, 2008.

[15] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE TPAMI*, no. 11, pp. 1254–1259, 1998.

[16] A. Borji, "What is a salient object? a dataset and a baseline model for salient object detection," *IEEE TIP*, vol. 24, no. 2, pp. 742–756, 2015.

[17] H. Li and K. N. Ngan, "A co-saliency model of image pairs," *IEEE TIP*, vol. 20, no. 12, pp. 3365–3375, 2011.

[18] P. Siva, C. Russell, T. Xiang, and L. Agapito, "Looking beyond the image: Unsupervised learning for object saliency and detection," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 3238–3245.

[19] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, "Global contrast based salient region detection," in *IEEE CVPR 2011*, 2011, pp. 409–416.

[20] S. Marat, T. H. Phuoc, L. Granjon, N. Guyader, D. Pellerin, and A. Guérin-Dugué, "Modelling spatio-temporal saliency to predict gaze direction for short videos," *International journal of computer vision*, vol. 82, no. 3, pp. 231–243, 2009.

[21] E. Vig, M. Dorr, T. Martinetz, and E. Barth, "Intrinsic dimensionality predicts the saliency of natural dynamic scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 6, pp. 1080–1091, 2012.

[22] Y. Luo and X. Tang, "Photo and video quality evaluation: Focusing on the subject," in *Proc. ECCV 2008*. Springer, 2008, pp. 386–399.

[23] H.-H. Yeh, C.-Y. Yang, M.-S. Lee, and C.-S. Chen, "Video aesthetic quality assessment by temporal integration of photo-and motion-based features," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1944–1953, 2013.

[24] Y. Y. Xiang and M. S. Kankanhalli, "Automated aesthetic enhancement of videos," in *Proc. Int. Conf. Multimedia*. ACM, 2010, pp. 281–290.

[25] F. Berthouzoz, W. Li, and M. Agrawala, "Tools for placing cuts and transitions in interview video," *ACM Trans. Graphics*, vol. 31, no. 4, pp. 67:1–67:10, 2012.

[26] I. Arev, H. S. Park, Y. Sheikh, J. Hodgins, and A. Shamir, "Automatic editing of footage from multiple social cameras," *ACM Trans. Graphics*, vol. 33, no. 4, pp. 81:1–81:10, 2014.

[27] J. Chang, D. Wei, and J. W. Fisher III, "A video representation using temporal superpixels," in *Proc. IEEE CVPR 2013*, 2013, pp. 2051–2058.

[28] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Proc. IEEE CVPR 2010*. IEEE, 2010, pp. 2432–2439.

[29] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Int. J. Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[30] B. Brown, *Cinematography: Theory and Practice*. Focal Press, Elsevier, 2012.

[31] R. Bresson, *Notes on Cinematography*. Urizen Books, New York, 1958.

[32] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Trans. Graphics*, vol. 28, no. 3, pp. 44:1–44:12, 2009.

[33] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Prez, "Video inpainting of complex scenes," *SIAM Journal on Imaging Sciences, Society for Industrial and Applied Mathematics*, vol. 7, no. 4, pp. 1993–2019, 2014.

Fang-Lue Zhang is a post doctor in in Tsinghua University. He received his BS degree from the Zhejiang University in 2009 and Ph.D degree from Tsinghua University in 2015. His research interests include computer graphics, image processing and enhancement, image and video analysis and computer vision.

Jue Wang is a Principle Research Scientist at Adobe Research. He received his B.E. (2000) and M.Sc. (2003) from Department of Automation, Tsinghua University, Beijing, China, and his Ph.D (2007) in Electrical Engineering from the University of Washington, Seattle, WA, USA. He received Microsoft Research Fellowship and Yang Research Award from University of Washington in 2006. He joined Adobe Research in 2007 as a research scientist. His research interests include image and video processing, computational photography, computer graphics and vision. He is a senior member of IEEE and a member of ACM.

Han Zhao is an undergraduate student at the Tsinghua University. He is currently interested in computer graphics, including image/video processing and animation.

Ralph R. Martin is currently a Professor at Cardiff University. He obtained his PhD degree in 1983 from Cambridge University. He has published more than 200 papers and 12 books, covering such topics as solid and surface modeling, intelligent sketch input, geometric reasoning, reverse engineering, and various aspects of computer graphics. He is a Fellow of: the Learned Society of Wales, the Institute of Mathematics and its Applications, and the British Computer Society. He is on the editorial boards of Computer Aided Design, Computer Aided Geometric Design, Geometric Models, the International Journal of Shape Modeling, CAD and Applications, and the International Journal of CADCAM.

Shi-Min Hu is currently a professor in the department of Computer Science and Technology, Tsinghua University, Beijing. He received the PhD degree from Zhejiang University in 1996. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conference. He is Editor-in-Chief of Computational Visual media, and on editorial board of several journals, including IEEE Transactions on Visualization and Computer Graphics, Computer Aided Design and Computer & Graphics.