Auto-Directed Video Stabilization with Robust L1 Optimal Camera Paths

Matthias Grundmann[1,2] Vivek Kwatra[1] Irfan Essa[2]
grundman@cc.gatech.edu kwatra@google.com irfan@cc.gatech.edu
[1]Google Research, Mountain View, CA, USA [2]Georgia Institute of Technology, Atlanta, GA, USA

Abstract

We present a novel algorithm for automatically applying constrainable, L1-optimal camera paths to generate stabilized videos by removing undesired motions. Our goal is to compute camera paths that are composed of constant, linear and parabolic segments mimicking the camera motions employed by professional cinematographers. To this end, our algorithm is based on a linear programming framework to minimize the first, second, and third derivatives of the resulting camera path. Our method allows for video stabilization beyond the conventional filtering of camera paths that only suppresses high frequency jitter. We incorporate additional constraints on the path of the camera directly in our algorithm, allowing for stabilized and retargeted videos. Our approach accomplishes this without the need of user interaction or costly 3D reconstruction of the scene, and works as a post-process for videos from any camera or from an online source.

1. Introduction

Video stabilization seeks to create stable versions of casually shot video, ideally relying on cinematography principles. A casually shot video is usually filmed on a handheld device, such as a mobile phone or a portable camcorder with very little stabilization equipment. By contrast, professional cinematographers employ a wide variety of stabilization tools, such as tripods, camera dollies and steadycams. Most optical stabilization systems only dampen high-frequency jitter and are unable to remove low-frequency distortions that occur during handheld panning shots, or videos shot by a walking person. To overcome this limitation, we propose an algorithm that produces stable versions of videos by removing undesired motions. Our algorithm works as a post process and can be applied to videos from any camera or from an online source without any knowledge of the capturing device or the scene.

In general, post-process video stabilization [10] consists of the following three main steps: (1) Estimating the original (potentially shaky) camera path, (2) Estimating a new smooth camera path, and (3) Synthesizing the stabilized video using the estimated smooth camera path.

We address all of the above steps in our work. Our key contribution is a novel algorithm to compute the optimal steady camera path. We propose to move a *crop window* of fixed aspect ratio along this path; a path optimized to include salient points and regions, while minimizing an L1-smoothness constraint based on cinematography principles. Our technique finds optimal partitions of smooth paths by breaking the path into segments of either constant, linear, or parabolic motion. It avoids the superposition of these three types, resulting in, for instance, a path that is truly static within a constant segment instead of having small residual motions. Furthermore, it removes low-frequency bounces, *e.g.* those originating from a person walking with a camera. We pose our optimization as a Linear Program (LP) subject to various constraints, such as inclusion of the crop window within the frame rectangle at all times. Consequently, we do not perform additional motion inpainting [10, 3], which is potentially subject to artifacts.

Related work: Current stabilization approaches employ key-point feature tracking and linear motion estimation in the form of 2D transformations, or use Structure from Motion (SfM) to estimate the original camera path. From this original shaky camera path, a new smooth camera path is estimated by either smoothing the linear motion models [10] to suppress high frequency jitter, or fitting linear camera paths [3] augmented with smooth changes in velocity to avoid sudden jerks. If SfM is used to estimate the 3D path of the camera, more sophisticated smoothing and linear fits for the 3D motion may be employed [8].

To rerender the original video as if it had been shot from a smooth camera path, one of the simplest and most robust approaches is to designate a virtual crop window of predefined scale. The update transform between the original camera path and the smooth camera path is applied to the crop window, casting the video as if it would have been shot from the smooth camera path. If the crop window does not fit within the original frame, undefined out-of-bound areas may be visible, requiring motion-inpainting [3, 10]. Addi-

Insta360 Exhibit 1023 Page 1
IPR2024-01186 Insta360 v. GoPro

Figure 1: Five stills from our video stabilization with saliency constraints using a face detector. Original frames on top, our face-directed final result at the bottom. The resulting optimal path is essentially static in y (the up and down motion of camera is completely eliminated) and composed of linear and parabolic segments in x. Our path centers the object of interest (jumping girl) in the middle of the crop window (bottom row) without sacrificing smoothness of the path. Please see accompanying video.

tionally, image-based rendering techniques [1] or light-field rendering (if the video was captured by a camera array [13]) can be used to recast the original video.

While sophisticated methods for 3D camera stabilization [8] have been recently proposed, the question of *how* the optimal camera path is computed is deferred to the user, either by designing the optimal path by hand or selecting a *single* motion model for the whole video (fixed, linear or quadratic), which is then fit to the original path. The work of Gleicher and Liu [3] was the first to our knowledge to use a cinematography-inspired optimization criteria. Beautifully motivated, the authors propose a system that creates a camera path using greedy key-frame insertion (based on a penalty term), with linear interpolation in-between. Their system supports post-process saliency constraints. Our algorithm approximates the input path by *multiple, sparse* motion models in one unified optimization framework including saliency, blur and crop window constraints. Recently, Liu et al. [9] introduced a technique that imposes subspace constraints [5] on feature trajectories when computing the smooth paths. However, their method requires long feature tracks over multiple frames.

Our proposed optimization is related to L1 trend filtering [6], which obtains a least square fit, while minimizing the second derivate in L1 norm, therefore approximating a set of points with linear path segments. However, our algorithm is more general, as we also allow for constant and parabolic paths (via minimizing the first and third derivate). Figure 8 shows that we can achieve L1 trend filtering through a particular weighting for our objective.

2. L1 Optimal Camera Paths

From a cinematographic standpoint, the most pleasant viewing experience is conveyed by the use of either static cameras, panning ones mounted on tripods or cameras placed onto a dolly. Changes between these shot types can be obtained by the introduction of a cut or jerk-free transitions, *i.e.* avoiding sudden changes in acceleration.

We want our computed camera path $P(t)$ to adhere to these cinematographic characteristics, but choose not to introduce additional cuts beyond the ones already contained in the original video. To mimic professional footage, we optimize our paths to be composed of the following path segments:

- A constant path, representing a static camera, *i.e.* $DP(t) = 0$, D being the differential operator.
- A path of constant velocity, representing a panning or a dolly shot, *i.e.* $D^2P(t) = 0$.
- A path of constant acceleration, representing the ease-in and out transition between static and panning cameras, *i.e.* $D^3P(t) = 0$.

To obtain the optimal path composed of distinct constant, linear and parabolic segments, instead of a superposition of them, we cast our optimization as a constrained L1 minimization problem. L1 optimization has the property that the resulting solution is sparse, *i.e.* it will attempt to satisfy many of the above properties along the path *exactly*. The computed path therefore has derivatives which are exactly zero for most segments. On the other hand, L2 minimization will satisfy the above properties *on average* (in a least-squared sense), which results in small but non-zero gradients. Qualitatively, the L2 optimized camera path always has some small non-zero motion (most likely in the direction of the camera shake), while our L1 optimized path is only composed of segments resembling a static camera, (uniform) linear motion, and constant acceleration.

Our goal is to find a camera path $P(t)$ minimizing the above objectives while satisfying specific constraints. We explore a variety of constraints:

Inclusion constraint: A crop window transformed by the path $P(t)$ should always be contained within the frame rectangle transformed by $C(t)$, the original camera path. When modeled as a hard constraint, this allows us to perform video stabilization and retargeting while guaranteeing that all pixels within the crop window contain valid information.

Proximity constraint: The new camera path $P(t)$ should preserve the original intent of the movie. For example,

Insta360 Exhibit 1023 Page 2
IPR2024-01186 Insta360 v. GoPro

if the original path contained segments with the camera zooming in, the optimal path should also follow this motion, but in a smooth manner.

Saliency constraint: Salient points (*e.g.* obtained by a face detector or general mode finding in a saliency map) should be included within all or a specific part of the crop window transformed by $P(t)$. It is advantageous to model this as a soft constraint to prevent tracking of salient points, which in general leads to non-smooth motion of the non-salient regions.

2.1. Solution via Linear Programming

For the following discussion we assume that the camera path $C(t)$ of the original video footage has been computed (*e.g.* from feature tracks) and is described by a parametric linear motion model at each instance of time. Specifically, let the video be a sequence of images $I_1, I_2, \ldots, I_n$, where each frame pair (I_{t-1}, I_t) is associated with a linear motion model $F_t(x)$ modeling the motion of feature points x from I_t to I_{t-1}. From now on, we will consider the *discretized* camera path C_t defined at each frame I_t. C_t is iteratively computed by the matrix multiplication

$$C_{t+1} = C_t F_{t+1} \implies C_t = F_1 F_2 ... F_t. \quad (1)$$

While we focus our discussion on 2D parametric motion models F_t, our system is theoretically applicable to higher dimensional *linear* motions though we do not explore them in this paper.

Given the original path C_t, we express the desired smooth path as

$$P_t = C_t B_t, \quad (2)$$

where $B_t = C_t^{-1} P_t$ is the *update* transform that when applied to the original camera path C_t, yields the optimal path P_t. It can be interpreted as the "stabilization and retargeting transform" (or crop transform) which is applied to the crop window centered at each frame to obtain the final stabilized video. The optimization serves to find the optimal stable camera path $P(t)$ minimizing the objective

$$\mathcal{O}(P) = w_1 |D(P)|_1 + w_2 |D^2(P)|_1 + w_3 |D^3(P)|_1 \quad (3)$$

subject to multiple previously mentioned constraints. Without constraints, the optimal path is constant: $P_t = I, \forall t$.

1. Minimizing $|D(P)|_1$: Using forward differencing; $|D(P)| = \sum_t |P_{t+1} - P_t| = \sum_t |C_{t+1} B_{t+1} - C_t B_t|$ using eq. (2). Applying the decomposition of C_t in eq. (1)

$$\begin{aligned} |D(P)| &= \sum_t |C_t F_{t+1} B_{t+1} - C_t B_t| \\ &\leq \sum_t |C_t| |F_{t+1} B_{t+1} - B_t|. \end{aligned}$$

With C_t known, we therefore seek to minimize the residual

$$\sum_t |R_t|, \quad \text{with } R_t := F_{t+1} B_{t+1} - B_t \quad (4)$$

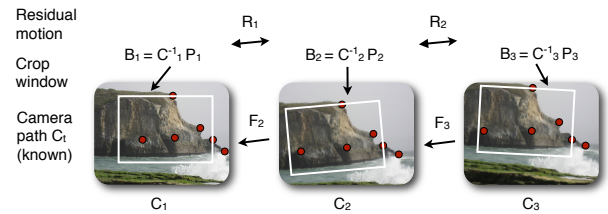


Figure 2: Camera path. We seek to find the update transform B_t for each frame, such that the L1 norm of the residual $|R_t| = |F_{t+1} B_{t+1} - F_t|$ is minimized for all t (static camera). By minimizing the difference of the residuals $|R_{t+1} - R_t|$ as well, we can achieve a path that is composed of static and linear segments only. Refer to text for parabolic segments.

over all B_t[1]. In fig. 2 we visualize the intuition behind this residual. A constant path is achieved when applying the update transform B_2 and feature transform F_2 in succession to frame I_2 yields the same result as applying B_1 to frame I_1, *i.e.* $R_1 = 0$.

2. Minimizing $|D^2(P)|_1$: While forward differencing gives $|D^2(P)| = \sum_t |DP_{t+2} - DP_{t+1}| = \sum_t |P_{t+2} - 2P_{t+1} + P_t|$, care has to be taken, as we model the error as additive instead of compositional. We therefore minimize directly the difference of the residuals

$$|R_{t+1} - R_t| = |F_{t+2} B_{t+2} - (I + F_{t+1}) B_{t+1} + B_t| \quad (5)$$

as indicated in fig. 2.

3. Minimizing $|D^3(P)|_1$: Similarly,

$$\begin{aligned} &|R_{t+2} - 2R_{t+1} + R_t| = \quad (6) \\ &|F_{t+3} B_{t+3} - (I + 2F_{t+2}) B_{t+2} + (2I + F_{t+1}) B_{t+1} - B_t|. \end{aligned}$$

4. Minimizing over B_t: As initially mentioned, the known frame-pair transforms F_t and the unknown update transforms B_t are represented by linear motion models. For example, F_t may be expressed as 6 DOF *affine* transformation

$$F_t = A(x; p_t) = \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} dx_t \\ dy_t \end{pmatrix}$$

with p_t being the parametrization vector $p_t = (dx_t, dy_t, a_t, b_t, c_t, d_t)^T$. Similar a 4 DOF *linear similarity* is obtained by setting $a_t = d_t$ and $b_t = -c_t$.

We seek to minimize the weighted L1 norm of the residuals derived in eqs. (4) to (6) over all update transforms B_t parametrized by their corresponding vector p_t. Then, the residual for the constant path segment in eq. (4) becomes

$$|R_t(p)| = |M(F_{t+1}) p_{t+1} - p_t|,$$

[1]Note, that we chose an additive error here instead of the compositional error $\min |S_t|$ s.t. $F_{t+1} B_{t+1} - B_t S_t = 0$, which is better suited for transformations, but quadratic in the unknowns and requires a costlier solver than LP.

Insta360 Exhibit 1023 Page 3
IPR2024-01186 Insta360 v. GoPro

where $M(F_{t+1})$ is a linear operation representing the matrix multiplication of $F_{t+1}B_{t+1}$ in parameter form.

5. The LP minimizing the L1 norm of the residuals (eqs. (4) to (6)) in parametric form can be obtained by the introduction of *slack* variables. Each residual will require the introduction of N slack variables, where N is the dimension of the underlying parametrization, *e.g.* $N = 6$ in the affine case. For n frames this corresponds to the introduction of roughly $3nN$ slack variables. Specifically, with e being a vector of N *positive* slack variables, we bound each residual from below and above *e.g.* for $|D(P)|$:

$$-e \leq M(F_{t+1})p_{t+1} - p_t \leq e$$

with $e \geq 0$. The objective is to minimize $c^T e$ which corresponds to the minimization of the L1 norm if $c = \mathbf{1}$. By adjusting the weights of c we can steer the minimization towards specific parameters, *e.g.* we can weight the strictly affine part higher than the translational part. This is also necessary as translational and affine parts have different scales, we therefore use a weighting of 100:1 for affine to translational parts.

Using the LP formulation of our problem, it easy to impose constraints on the optimal camera path. Recall, that p_t represents the parametrization of the update transform B_t, which transforms a crop window originally centered in the frame rectangle. In general, we wish to limit how much B_t can deviate from the original path to preserve the intent of the original video[2]. Therefore, we place strict bounds on the affine part of the parametrization p_t: $0.9 \leq a_t, d_t \leq 1.1$, $-0.1 \leq b_t, c_t \leq 0.1$, $-0.05 \leq b_c + c_t \leq 0.05$, and $-0.1 \leq a_t - d_t \leq 0.1$.

The first two constraints limit the range of change in zoom and rotation, while the latter two give the affine transform more rigidity by limiting the amount of skew and non-uniform scale. Therefore in each case, we have an upper (ub) and a lower bound (lb), which can be written as

$$\text{lb} \leq Up_t \leq \text{ub}, \tag{7}$$

for a suitable linear combination over p_t, specified by U. To satisfy the inclusion constraint, we require that the 4 corners $c_i = (c_i^x, c_i^y)$, $i = 1..4$ of the crop rectangle reside inside the frame rectangle, transformed by the linear operation $A(p_t)$, as illustrated in fig. 3. In general, it is feasible to model hard constraints of the form "transformed point in convex shape" in our framework, *e.g.* for an affine parametrization of p_t, we require

$$\begin{pmatrix}0\\0\end{pmatrix} \leq \underbrace{\begin{pmatrix}1 & 0 & c_i^x & c_i^y & 0 & 0\\ 0 & 1 & 0 & & c_i^x & c_i^y\end{pmatrix}}_{:=\text{CR}_i} p_t \leq \begin{pmatrix}w\\h\end{pmatrix}, \tag{8}$$

with w and h being the dimensions of the frame rectangle.

[2]Also for video stabilization extreme choices for scale and rotation might minimize the residual better but discard a lot of information.

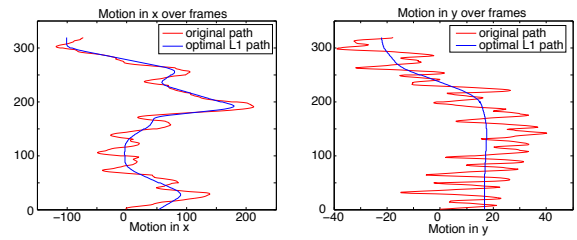


Figure 4: Optimal camera path obtained via our constrained LP formulation for the video in fig. 10. Shown is the motion in x and y over a period of 320 frames, using the inclusion constraint for a crop window of 75% size of the original frame. Note how the optimal path is composed of constant, linear and parabolic arcs. Our method is able to replace the low-frequency bounce in y (person walking with a camera) with a static camera while guaranteeing that all pixels within the crop window are valid.

The complete L1 minimization LP for the optimal camera path with constraints is summarized in Algorithm 1. We show an example of our computed optimal path from the original camera path in fig. 4. Note how the low-frequency bounce in y, originating from a walking person while filming, can be replaced by a static camera model.

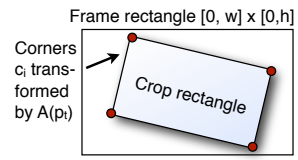


Figure 3: Inclusion constraint.

Algorithm 1: Summarized LP for the optimal camera path.

Input: Frame pair transforms F_t, $t = 1..n$
Output: Optimal camera path $P_t = C_t B_t = C_t A(p_t)$

$$\begin{aligned}
\text{Minimize} \quad & c^T e \\
\text{w.r.t.} \quad & p = (p_1, ..., p_n) \\
\text{where} \quad & e = (e^1, e^2, e^3), e^i = (e_1^i, ..., e_n^i) \\
& c = (w_1, w_2, w_3) \\
\text{subject to} \quad & \\
\text{smoothness} \quad & \begin{cases} -e_t^1 \leq R_t(p) \leq e_t^1 \\ -e_t^2 \leq R_{t+1}(p) - R_t(p) \leq e_t^2 \\ -e_t^3 \leq R_{t+2}(p) - 2R_{t+1}(p) + R_t(p) \leq e_t^3 \\ e_t^i \geq 0 \end{cases} \\
\text{proximity} \quad & \text{lb} \leq Up_t \leq \text{ub} \\
\text{inclusion} \quad & (0,0)^T \leq \text{CR}_i p_t \leq (w,h)^T
\end{aligned}$$

2.2. Adding saliency constraints

While the above formulation is sufficient for video stabilization, we can perform *directed* video stabilization, *automatically* controlled by hard and soft saliency constraints, using a modified feature-based formulation. Optimizing for saliency measures imposes additional constraints on the update transform. Specifically, we require that salient points

Insta360 Exhibit 1023 Page 4
IPR2024-01186 Insta360 v. GoPro

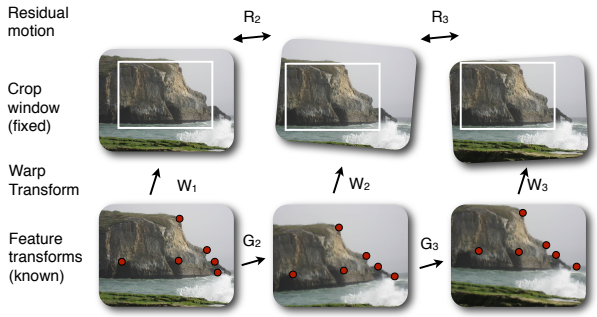


Figure 5: Feature path. Instead of transforming the crop window, we transform original frame such that the feature movement within the static crop window is smooth.

reside *within* the crop window, which is essentially the inverse of our inclusion constraint. We therefore consider optimizing the inverse of the update transform, *i.e.* a warp transform W_t applied to set of features in each frame I_t as indicated in fig. 5. We denote the inverse of F_t by $G_t = F_t^{-1}$. Instead of transforming the crop window by B_t, we seek a transform W_t of the current features, such that their motion within a *fixed* crop window is only composed of static, linear or parabolic motion. The actual update or stabilization transform is then given by $B_t = W_t^{-1}$. We briefly derive the corresponding objectives for $D^i W_t,\ i = 1..3$ based on fig. 5:

1. Minimize $|DW_t|$**:** $|R_t| = |W_{t+1}G_{t+1} - W_t|$,
2. Minimize $|D^2W_t|$**:**
$|R_{t+1} - R_t| = |W_{t+2}G_{t+2} - W_{t+1}(I + G_{t+1}) + W_t|$,
3. Minimize $|D^3W_t|$**:** $|R_{t+2} - 2R_{t+1} + R_t| =$
$|W_{t+3}G_{t+3} - W_{t+2}(I + 2G_{t+2}) + W_{t+1}(2I + G_{t+1}) - W_t|$.

The advantage of this feature path formulation lies in the flexibility it allows for handling saliency constraints. Suppose we want a specific point (*e.g.* mode of a saliency map) or convex region (*e.g.* from a face detector) to be contained within the crop window. We denote the set of salient points in frame I_t by s_i^t. As we are estimating the feature warp transform instead of the crop window transform, we can introduce *one-sided*[3] bounds on s_i^t transformed by $A(p_t)$:

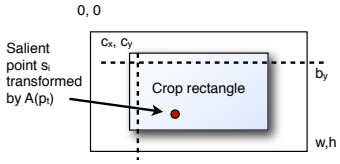


Figure 6: Canonical coordinate system for retargeting.

$$\begin{pmatrix} 1 & 0 & s_i^x & s_i^y & 0 & 0 \\ 0 & 1 & 0 & & s_i^x & s_i^y \end{pmatrix} p_t - \begin{pmatrix} b_x \\ b_y \end{pmatrix} \geq \begin{pmatrix} -\epsilon_x \\ -\epsilon_y \end{pmatrix},$$

with $\epsilon_x, \epsilon_y \geq 0$. The bounds (b_x, b_y) denote how far (at least) from the top-left corner should the saliency points lie, as indicated in the inset fig. 6. A similar constraint is introduced for the bottom-right corner. Choosing $b_x = c_x$ and $c_y = b_y$ will ensure that the salient points lie within the crop window. For $b_x > c_x$ the salient points can be moved to a specific region of the crop rectangle, *e.g.* to the center as demonstrated in fig. 1. Choosing $\epsilon_x, \epsilon_y = 0$ makes it a hard constraint; however with the disadvantage that it might conflict with the inclusion constraint of the frame rectangle and sacrifice path smoothness. We therefore opt to treat ϵ_x, ϵ_y as new slack variables, which are added to the objective of the LP. The associated weight controls the trade off between a smooth path and the retargeting constraint. We used a retargeting weight of 10 in our experiments.

It is clear that the feature path formulation is more powerful than the camera path formulation, as it allows retargeting constraints besides the proximity and inclusion constraints. However, the inclusion constraint needs to be adjusted, as the crop window points are now transformed by the inverse of the optimized feature warp transform, making it a non-linear constraint. A solution is to require the transformed frame corners to lie within a rectangular area around the crop rectangle as indicated in fig. 7, effectively replacing inclusion and proximity constraints.

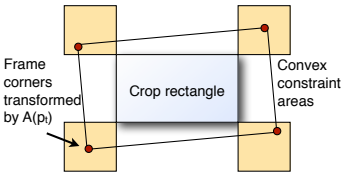


Figure 7: Inclusion constraint for the feature path. The transformed frame corners have to stay within the convex constraint areas (indicated in orange)

An interesting observation is that the estimation of optimal feature paths can be achieved directly from feature points f_k^t in frame I_t, *i.e.* without the need to compute G_t. In this setting, instead of minimizing the L1 norm of the parametrized residual $R(p_t)$, we directly minimize the L1 norm of feature distances. R_t becomes

$$|R_t| = \sum_{f_k:\text{feature matches}} |W(p_t)f_k^t - W(p_{t+1})f_k^{t+1}|_1.$$

As G_t is computed to satisfy $G_{t+1}f_k^t = f_k^{t+1}$ (under some metric), we note that the previously described optimization of feature warps W_t from feature transforms G_t essentially averages the error over all features instead of selecting the best in an L1 sense. We implemented the estimation of the optimal path directly from features for reference, but found it to have little benefit, while being too slow due to its complexity to be usable in practice.

3. Video Stabilization

We perform video stabilization by (1) estimating the per-frame motion transforms F_t, (2) computing the optimal

[3]Compare to two-sided bounds for the inclusion constraint in eq. (8).

Insta360 Exhibit 1023 Page 5
IPR2024-01186 Insta360 v. GoPro

camera path $P_t = C_t B_t$ as described in section 2, and (3) stabilizing the video by warping according to B_t.

For motion estimation, we track features using pyramidal Lucas-Kanade [12]. However, robustness demands good outlier rejection. For dynamic video analysis, global outlier rejection is insufficient, whereas the short baseline between adjacent video frames makes fundamental matrix based outlier rejection unstable. Previous efforts resolve this by undertaking 3D reconstruction of the scene via SfM [8], which is computationally expensive in addition to having stability issues of its own.

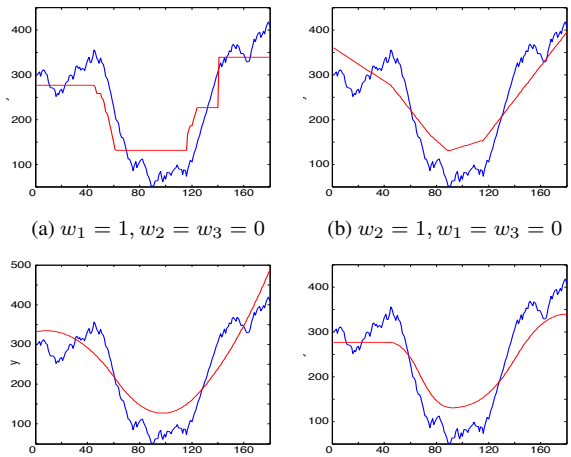
We employ *local* outlier rejection by discretizing features into a grid of 50×50 pixels, applying RANSAC within each grid cell to estimate a translational model, and only retaining those matches that agree with the estimated model up to a threshold distance (< 2 pixels). We also implemented a real-time version of graph-based segmentation [2] in order to apply RANSAC to all features within a segmented region (instead of grid cells), which turns out to be slightly superior. However, we use the grid-based approach for all our results, as it is approximately 40% faster.

Subsequently, we fit several 2D linear motion models (translation, similarity and affine) to the tracked features. While L2 minimization via normal equation with pre-normalization performs well in most cases, we noticed instabilities in case of sudden near-total occlusions. We therefore perform the fit in L1 norm via the LP solver[4], which increases stability in these cases by automatically performing feature selection. To our knowledge, this is a novel application of L1 minimization for camera motion estimation, and gives surprisingly robust results.

Once the camera path is computed as set of linear motion models, we fit the optimal camera path according to our L1 optimization framework subject to proximity and inclusion constraints as described in section 2. A crucial question is how to chose the weights $w_1 - w_3$ in the objective eq. (3)? We explore different weightings for a synthetic path in fig. 8. If only one of the three derivative constraints is minimized, it is evident that the original path is approximated by either constant non-continuous paths (fig. 8a), linear paths with jerks (fig. 8b), or smooth parabolas but always non-zero motion (fig. 8c). A more pleasant viewing experience is conveyed by minimizing all three objectives simultaneously. Though the absolute values of the weights are not too important, we found eliminating jerks to be most important, which is achieved when w_3 is chosen to be an order of magnitude larger than both w_1 and w_2.

The choice of the underlying motion model has a profound effect on the stabilized video. Using affine transforms instead of similarities has the benefit of two added degrees of freedom but suffers from errors in skew which leads to effects of non-rigidity (as observed by [8]). We therefore use

[4]We use the freely available COIN CLP simplex solver.



(a) $w_1 = 1, w_2 = w_3 = 0$ (b) $w_2 = 1, w_1 = w_3 = 0$

(c) $w_3 = 1, w_1 = w_2 = 0$ (d) $w_1 = 10, w_2 = 1, w_3 = 100$

Figure 8: Optimal path (red) for synthetic camera path (blue) shown for various weights of the objective eq. (3).

similarities to construct our optimal path. However similarities (like affine transforms) are unable to model non-linear inter-frame motion or rolling shutter effects, resulting in noticeable residual *wobble*, which we address next.

Residual Motion (Wobble and Rolling Shutter) Suppression: In order to precisely model inter-frame motion, necessary for complete shake-removal, motion models with higher DOF than similarities, *e.g.* homographies, are needed. However, higher DOF tend of overfit even if outlier rejection is employed. Consequently, one can achieve good registration for a few frames but their composition starts to quickly become unstable, *e.g.* homographies start to suffer from excessive skew and perspective. We suggest a robust, hybrid approach, initially using similarities (for frame transforms) $F_t := S_t$ to construct the optimal camera path, thereby ensuring rigidity over all frames. However, we apply the rigid camera path, as computed, only for every $k = 30$ keyframes. For intermediate frames,

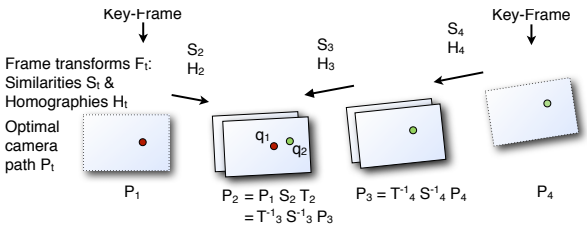


Figure 9: Wobble suppression. The key idea is to decompose the optimal path P_t into the lower-parametric frame transform S_t used as input and a residual T_t (representing the smooth shift added by the optimization to satisfy the constraints). S_t is replaced by a higher parametric model H_t to compute the actual warp. For consistency, the warp is computed forward (red) from previous and backward (green) from next key-frame, and the resulting locations q_1 and q_2 are blended linearly.

Insta360 Exhibit 1023 Page 6
IPR2024-01186 Insta360 v. GoPro

Figure 10: Reducing rolling shutter by our wobble suppression technique. Shown are the result for two frames 1/3 second apart. Top row: Original frames m (left) and $m+1$ (right). Middle row: Stabilization result without wobble suppression. Bottom Row: Stabilization with wobble suppression. Notice, how wobble suppression successfully removes the remaining skew caused by rolling shutter. (The yellow traffic sign is tilted in reality.)

we use higher dimensional homographies $F_t := H_t$ to account for misalignments. As indicated in fig. 9, we decompose the difference between two optimal (and rigid) adjacent camera transforms, $P_1^{-1}P_2$, into the known estimated similarity part S_2 and a smooth residual motion T_2, *i.e.* $P_1^{-1}P_2 = S_2T_2$ ($T_2 = \mathbf{0}$ implies static camera). We then replace the low-dimensional similarity S_2 with the higher-dimensional homography H_2, resulting in $P_1^{-1}P_2 := H_2T_2$. For each intermediate frame, we concatenate these replacements starting from its previous and next keyframes. This effectively results in two sample locations q_1, q_2 per pixel (indicated with red and green in fig. 9), with an average error of around $2-5$ pixels in our experiments. We use linear blending between these two locations to determine a per-pixel warp for the frame.

4. Video Retargeting

Video retargeting aims to change the aspect ratio of a video while preserving salient and visually prominent regions. Recently, a lot of focus has been on "content-aware" approaches that either warp frames based on a saliency map [14] or remove and duplicate non-salient seams [11, 4], both in a temporally coherent manner.

In section 2.2, we showed how we can direct the crop window to include salient points without having to sacrifice smoothness and steadiness of the resulting path. On the other hand, if the input video is already stable, *i.e.* $C(t)$ is smooth, we can explicitly model this property by sidestepping the estimation of each frame transform F_t, and force it to the identity transform $F_t = I$, $\forall t$. This allows us to steer the crop window based on saliency and inclusion constraints alone, achieving video retargeting by automatic pan-and-scan. Simply put, video retargeting falls out as a special case of our saliency based optimization, when the input video is assumed to be stable. In contrast to the work by Liu and Gleicher [7], our camera paths are not constrained to a single pan, allowing more freedom (*e.g.* subtle zoom) and adaptation to complex motion patterns.

While several measures of saliency exist, we primarily focus on motion-driven saliency. We are motivated by the assumption that viewers direct their attention towards moving foreground objects, a reasonable assumption within limitations. Using a fundamental matrix constraint and clustering on KLT feature tracks, we obtain foreground saliency features as shown in fig. 12, which are then used as constraints, as described in section 2.2.

5. Results

We show some results of video-stabilization using our optimal camera paths on a YouTube "Fan-Cam" video in fig. 11. Our optimization conveys a viewing experience very close to professional footage. In fig. 1, we demonstrate the ability to include saliency constraints, here derived from a face detector, to frame the dancing girl at the center of resulting video without sacrificing smoothness. In the accompanying video, the reader will notice occasional blur caused by high motion peaks in the original video. Stabilization techniques pronounce blur, as the stabilized result does not agree with the perceived (blurred) motion. In the video we show our implementation of Matsushita et al.'s [10] blur removal technique; however, the blur is too pronounced and the technique, suffering from temporal inconsistencies, performs poorly. However, our framework allows for the introduction of *motion constraints*, *i.e.* after determining which frames are blurred, we can force the optimal camera path to agree with the motion of the blur. This effectively reduces the perceived blur while still maintaing smooth (but accelerated) camera motion.

We demonstrate the ability to reduce rolling shutter in fig. 10; notice how the skew of the house is removed. Using motion based saliency constraints we can perform video retargeting using a form of automated pan-and-scan in our framework; see fig. 12 for an example. While we effectively *crop* the frame, our technique is extremely robust, avoiding spatial and temporal artifacts caused by other approaches.

As dynamics are impossible to judge from images, we would encourage the reader to watch the accompanying video. We evaluate our approach on wide variety of videos, comparing our video stabilization to both current methods of Liu et al. [8, 9]. We also include an example comparing the light field approach of Brandon et al. [13]. For video retargeting, we show more examples and compare to [11, 14].

Our technique is based on per-frame feature tracks only, without the need of costly 3D reconstruction of the scene. We use robust and iterative estimation of motion models

Insta360 Exhibit 1023 Page 7
IPR2024-01186 Insta360 v. GoPro

Figure 11: Example from YouTube "Fan-Cam" video. Top row: Stabilized result, bottom row: Original with optimal crop window. Our system is able remove jitter as well as low-frequency bounces. Our L1 optimal camera path conveys a viewing experience that is much closer to a professional broadcast than a casual video. Please see video.

Figure 12: Example of video retargeting using our optimization framework. Top row: Original frame (left) and our motion aware saliency (right). Foreground tracks are indicated by red, the derived saliency points used in the optimization by black circles. Bottom row: Our result (left), Wang et al.'s [14] result (middle) and Rubinstein et al.'s [11] result (right).

(from lower to higher dimensional), only using inliers from the previous stage. Our technique is fast; we achieve 20 fps on low-resolution video, while wobble suppression requires grid-based warping and adds a little more overhead. Our unoptimized motion saliency runs at around 10 fps.

6. Conclusion, Limitations, and Future Work

We have proposed a novel solution for video stabilization and retargeting, based on computing camera paths directed by a variety of automatically derived constraints. We achieve state-of-the-art results in video stabilization, while being computational cheaper and applicable to a wider variety of videos. Our L1 optimization based approach admits multiple simultaneous constraints, allowing stabilization and retargeting to be addressed in a unified framework. A stabilizer based on our algorithm, with *real-time* previews, is freely available at youtube.com/editor.

Our technique may not be able to stabilize all videos, *e.g.* low feature count, excessive blur during extremely fast motions, or lack of rigid objects in the scene might make camera path estimation unreliable. Employing heuristics to detect these unreliable segments, we reset the corresponding linear motion models F_t to the identity transform, effectively falling back to the shaky video. Further, the use of cropping discards information, something a viewer might dislike. Our computed path is optimal for a *given* crop window size, which is the only input required for our algorithm. In future work, we wish to address automatic computation of the optimal crop size, as currently we leave this as a choice to the user.

Acknowledgements We thank Tucker Hermans for narrating the video and Kihwan Kim for his selection of fan-cam videos. We are especially grateful to the YouTube Editor team (Rushabh Doshi, Tom Bridgwater, Gavan Kwan, Alan deLespinasse, John Gregg, Eron Steger, John Skidgel and Bob Glickstein) for their critical contributions during deployment, including distributed processing for real-time previews, front-end UI design, and back-end support.

References

[1] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *IEEE CVPR*, 2001. 226

[2] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004. 230

[3] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Mult. Comput. Commun. Appl.*, 2008. 225, 226

[4] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Discontinuous seam-carving for video retargeting. *IEEE CVPR*, 2010. 231

[5] M. Irani. Multi-frame correspondence estimation using subspace constraints. *Int. J. Comput. Vision*, 48:173–194, July 2002. 226

[6] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky. l1 trend filtering. *SIAM Review*, 2009. 226

[7] F. Liu and M. Gleicher. Video retargeting: automating pan and scan. In *ACM MULTIMEDIA*, 2006. 231

[8] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. In *ACM SIGGRAPH*, 2009. 225, 226, 230, 231

[9] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. In *ACM Transactions on Graphics*, volume 30, 2011. 226, 231

[10] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28, July 2006. 225, 231

[11] M. Rubinstein, A. Shamir, and S. Avidan. Improved seam carving for video retargeting. In *ACM SIGGRAPH*, 2008. 231, 232

[12] J. Shi and C. Tomasi. Good features to track. In *IEEE CVPR*, 1994. 230

[13] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala. Light field video stabilization. In *ICCV*, 2009. 226, 231

[14] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel. Motion-aware temporal coherence for video resizing. *ACM SIGGRAPH ASIA*, 2009. 231, 232

Insta360 Exhibit 1023 Page 8
IPR2024-01186 Insta360 v. GoPro