Re-Cinematography: Improving the Camera Dynamics of Casual Video

Michael L. Gleicher
Department of Computer Sciences
University of Wisconsin-Madison
gleicher@cs.wisc.edu

Feng Liu
Department of Computer Sciences
University of Wisconsin-Madison
fliu@cs.wisc.edu

ABSTRACT

This paper presents an approach to post-processing casually captured videos to improve apparent camera movement. *Re-cinematography* transforms each frame of a video such that the video better follows cinematic conventions. The approach breaks videos into shorter segments. For segments of the source video where the camera is relatively static, re-cinematography uses image stabilization to make the result look locked-down. For segments with camera motions, camera paths are keyframed automatically and interpolated with matrix logarithms to give velocity-profiled movements that appear intentional and directed. The approach automatically balances the tradeoff between motion smoothness and distortion to the original imagery. Results from our prototype show improvements to poor quality home videos.

Categories and Subject Descriptors

H.5.1 [**Information Presentation**]: Multimedia—*Video*

General Terms: Design, Experimentation, Human Factors
Keywords: image stabilization, casual video, cinematography

1. INTRODUCTION

The increasing availability of video capture devices means that one is always on hand to capture an interesting moment. It also leads to a growing amount of video that is created without the planning and artistry required to make *good* video: when a baby takes his first steps, a proud parent is lucky to remember to take off the lens cap, never mind set up a fluid-head tripod or remember what they learned in film school. Such casually captured videos often record precious memories but are also difficult to watch. Our goal is to create tools that can post-process these videos to improve them. In this paper we introduce *Re-Cinematography,* an approach that processes recorded video clips to improve one aspect of their quality: the apparent camera movements.

Casual video is often made with unplanned and poorly executed camera motions. The best known problem is shakiness from a hand-held camera. These undesirable high-frequency movements can be removed by *image stabilization.* Image stabilization adjusts each frame to change the apparent movement between frames. These transformations typically damage the individual frames through distortions, lost portions of the frame, and loss of detail. Removing unwanted shakiness usually makes this an acceptable tradeoff. Almost all video cameras implement some image stabilization, and widely available software solutions can provide even more aggressive stabilization (i.e. larger changes).

Re-Cinematography extends image stabilization, adjusting video frames so that the resulting apparent camera motion follows cinematic conventions. Rather than simply removing high frequencies, the video is adjusted such that it appears as if the camera were on a properly damped tripod: either holding still, or moving with directed, velocity-profiled movements. Like stabilization, transforming the frames involves tradeoffs between image and motion quality. Re-Cinematography considers these tradeoffs explicitly.

1.1 Overview

Consider a video clip of a skier (Fig 1) taken with a hand-held digital camera panning to follow the skier. Unlike a video made by a professional (with considerable camera experience and a fluid-head tripod), this casual video has small jitters (from being hand-held) and does not have a fluid, goal-directed motion. Image stabilization (including what is built into the camera) addresses the former problem; Re-cinematography addresses both.

Re-cinematography can automatically turn this video clip into a pan, similar to what one would obtain using a good tripod. Conceptually, the method uses inter-frame motion estimation to build an image mosaic that completely stabilizes the camera movement to create a panoramic image, and then animates a virtual camera that views this mosaic. In practice, an implementation need not form the panoramic image (except to create diagrams like Fig 1). More dramatic examples of the contrast between image stabilization (that reduces jitter) and re-cinematography (that creates directed motions) are given in §5.1, e.g. Fig 9.

Re-cinematography avoids the limitations of image mosaicing by breaking video into a series of shorter segments, and choosing appropriate virtual camera motions for each. These motions are created automatically for each video segment based on the observed camera motions; nearly static source video results in no motion, while larger camera movements are replaced with goal-directed, velocity-profiled paths. The motions are created by automatically keyframing the camera paths. Problems in direct paths are addressed by

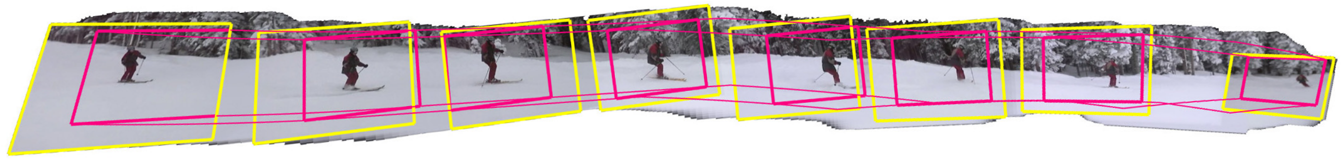


Figure 1: Schematic of the Re-cinematography process. Conceptually, an image mosaic is constructed for the video clip and a virtual camera viewing this mosaic is keyframed. Yellow denotes the source camera path, magenta (dark) the keyframed virtual camera.

inserting keyframes. For example, our approach uses *motion salience* to determine what is likely to be important to the viewer and insures that these objects stay on screen. Other constraints include limitations on the scene analysis and amount of distortion introduced by the transformations.

Re-cinematography is a novel approach to improving the apparent camera motion in casual videos. Specific contributions include:

- A local mosaic approach that allows mosaic-based techniques to be applied to a wider range of video content.
- A model for creating camera motions directly as 2D transformations that map cinematographic ideals onto implementable mathematical foundations such as matrix exponential interpolation.
- An approach that explicitly considers the tradeoffs in video motion alteration.
- Methods that plan new virtual camera motions that optimize the tradeoff between motion quality and image quality.
- Methods that use motion salience to avoid trimming important aspects of the video.

1.2 Good Video

Re-cinematography aims to create good apparent camera motions, not simply remove shakiness. Cinematography is the art of choosing what the viewer in a film or video sees through camera positioning and movement. Good video uses the control of viewpoint effectively. Guidelines and conventions developed over the past century suggest what camera work is effective. Skilled filmmakers may bend these rules to achieve a desired effect, but this is different than a causal videographer ignoring the rules.

Literature on film helps indirectly. General books on the film arts (such as [6]), or more specifically about cinematography (such as [22] or [2]), discuss cinematography and the aesthetic considerations of camera movement. Texts such as [5] and [7] give more technical discussion and a relationship to low-level perceptual issues. These latter discussions are more readily translated into computational frameworks.

To borrow terminology from [7], camera movements should be motivated and intentional. If the filmmaker exerts control over the viewer's viewpoint they should have a reason so the viewer can more easily follow. This suggests that small adjustments should be avoided, i.e. that the camera is "locked down" unless it makes a significant movement. When there are movements, they should be goal-directed (since they have an intent), rather than wandering.

To create these carefully planned camera motions, filmmakers usually use camera supports. A good support, like a fluid-head tripod, not only makes sure that non-moving shots are stable, but also helps insure that motions have continuity in velocity and direction. See Brown [7] for an array of supports used by filmmakers to create camera movements. For casual videography the guidelines still apply. Web references for videography tips have a consistent message: use a tripod (the most common kind of support).

For re-cinematography, we translate these guidelines into more specific, implementable goals. Small movements are better replaced with a static camera. Larger movements should follow directed paths and move with velocity-profiled paths between goals. By velocity-profiled, we mean that it moves with a constant velocity, except possibly for an initial acceleration or deceleration (§3.4). Re-cinematography achieves these goals by estimating the camera motion in the source video and adjusting it. Other details of cinematography, such as size constancy and leading, are also considered in our algorithm design.

Without an understanding of the motivation for the original camera motion, re-cinematography relies on the source movement as a basis for a new one, effectively predicting what the movement might have been if the videographer had a properly damped camera support. This reliance on the source motion is convenient as practical considerations limit how far re-cinematography can deviate from it.

2. RELATED WORK

The use of 2D image transformations to change apparent camera motion has a long tradition. Lab shots that moved the film during optical printing [6] evolved into the complex effects used in animation where flying is simulated by moving the animation camera over a panoramic background [21]. Re-cinematography is inspired by [36] that introduced the camera-over-panorama idea in computer graphics. Unlike their work, we use real video rather than 3D models, a full projective transformation, and do not build or distort panoramas.

Image or video stabilization, of which re-cinematography is a generalization, uses image movement to change apparent camera motions. The methods are ubiquitous and implemented in almost all consumer video cameras, and the literature on the subject is too vast to survey here. Image stabilization comprises three main components: motion estimation, motion smoothing and image production. Our innovation is in the second step. Prior work on motion smoothing falls into three categories: causal (or online) low-pass filters (c.f. [24]), general low-pass filters (c.f. [27] and [11]), and mosaic-based approaches.

Mosaic-based video stabilization is impractical for cleanup of general casual videos [27]. Our approach reaps its benefits by applying it when it can stabilize a shot and extending it to create good camera movements. Other applications of mosaicing have inspired us. For example, Irani et al. [19] provide numerous applications. They introduce "dynamic mosaics" that include some of the original camera motion. Proscenium [3] uses mosaic-based stabilization to provide a video editing tool. Salient Stills [33] creates single frame summaries of video clips. Irani and Anandan [18] use mosaics to summarize videos for indexing. Dony et. al [12] use mosaics to create storyboards. We adopt their approach of breaking video into shorter segments to avoid the limits of mosaicing.

Prior work uses knowledge of cinematography to plan camera movements. These works inspired us by demonstrating different domains and applications. Camera motion in 3D environments has been considered by many; see [10] for a survey. Virtual videography [16] determines virtual camera movements given a static camera source video in the specific case of a chalkboard lecture. We attempt to apply similar reasoning to a very different class of source material. Video retargeting [25] also considers choosing new camera movements for existing video. However it considers transforming feature films, not casual video, and therefore must assume that the source camera motion is the intentional work of a skilled filmmaker and should be preserved.

Re-cinematography does not consider editing: the resulting video clips are the same length as the source. Several prior systems have considered automating the editing of home video. Notable examples include Hitchcock [13] and AVE [17]. Video editing involves selecting and assembling clips of video, whereas re-cinematography aims to improve the quality of the clips. The two approaches could complement one another nicely. For example, Hitchcock uses motion estimation to find bad camera motions that it discards. Instead, it could use re-cinematography to repair them.

3. TECHNICAL FOUNDATIONS

3.1 Motion Estimation

This section reviews some of the basic mathematics of image and video stabilization in order to clarify our notation.

Consider two images a and b from a video. The stabilization transformation $\mathbf{S_b}(a) \in \mathcal{R}^2 \rightarrow \mathcal{R}^2$ maps the positions of points in frame a to where they appear in frame b, if those points were in the same position in the world. If we apply $\mathbf{S_b}(a)$ to the image of frame a, the result would be the view of the world of a taken from the viewpoint of b. Were we to apply the corresponding transform to each frame of the video, the result would be a video taken from the single viewpoint of b. There would be no camera motion, the images have been stabilized.

An image created by combining a collection of images all transformed to a common coordinate system is called a *mosaic*. We refer to the common image as the *base frame.*

For arbitrary scenes and arbitrary camera movements, the stabilization transformation can be complex. However, in two important cases, the transformation will be a 2D projection: if the world is a single plane or the camera rotates around its optical center [31]. A 2D projection

$$x' = \frac{s_1x + s_2y + s_3}{s_7x + s_8y + 1}, y' = \frac{s_4x + s_5y + s_6}{s_7x + s_8y + 1}$$

is an 8 parameter transformation that can be expressed as a linear transformation in homogeneous coordinates, i.e. a $3x3$ matrix. A planar projective transformation is also known as a *homography.*

In the event that the restrictions to camera rotation or a flat world do not apply, a 2D projection is only an approximation to the real stabilization transformation. This

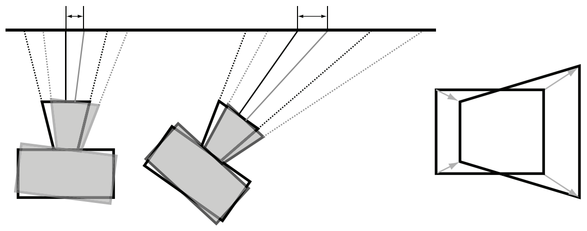


Figure 2: Identical camera rotations can lead to very different motions on the base plane. The movements are identical in local coordinates. To measure the local transformation, we measure the distance moved by each corner of the identity image. This would be the same for both cameras to the left.

simplified model is preferred because it is mathematically convenient (matrix operations allow for manipulation), has proven to be a reasonable approximation to many real situations, and a plethora of techniques exist for motion estimation (that is, estimating $\mathbf{S_b}(a)$ from the images it relates). Historically, all image stabilization methods (with the notable exception of [8]) use either homographies, or a more restricted class of transformations that are a subset of them.

Motion estimation typically performs best between similar images. Most compute transformations between adjacent video frames, i.e. $\mathbf{S}_i(i+1)$, and determine other transformations between other pairs by composition,

$$\mathbf{S}_b(a) = \text{if } (a > b) \prod_{i=b}^{a-1} \mathbf{S}_i(i+1), \text{if } (a < b) \prod_{i=b}^{a+1} \mathbf{S}_{i-1}^{-1}(i). \quad (1)$$

Motion estimation is common to a number of image, video, and vision applications and has been studied extensively. Szeliski's recent survey [32] is encyclopedic and insightful. Our implementation is a variant of the feature-based approaches he describes. Specifically, we use SIFT [26] to identify points and associate them with descriptors, and then match these descriptors between frames. SIFT seems to be robust against lighting changes, blurring, and compression artifacts.

The parameters of the homographies are difficult to interpret directly [14]. A four-point representation, which measures the displacement of the four corners of the source image, gives more meaningful metrics in units of pixels. Making these measurements in local coordinates (Fig. 2), i.e. between pairs of frames, avoids issues from the non-linearity and scene dependence of the transformations.

The sum of the corner vectors (the arrows in Fig 2) gives a measurement of apparent camera movement, while the sum of their magnitudes measures the amount of change between frames (including zoom). The motion measurements are added over a duration of video to form a path, the ratio of arc-length to the net displacement of this path estimates its "curviness."

3.2 Local Mosaics

Mosaic image stabilization is attractive because it gives the appearance of a locked down camera. Unfortunately, it is impractical for longer videos and complex motions [27] for a number of reasons. Therefore, we limit our use of mosaics to short durations of video. We call this a *local mosaic* as we consider a mosaic around the locality of a given frame.

One limit of mosaics for longer video segments is error accumulation in Eqn. 1. Even the best homography estimation will have errors, due to either mis-registration or a poor fit to the planar projective model. As the sequence grows longer, small interframe errors are amplified as they are added in Eqn. 1. Therefore, for each homography we compute an *unreliability score*. The score is used in two ways: first, homographies with very high scores are considered too unreliable to use. Second, the length of a local mosaic is limited such that the sum of the unreliability is below a threshold.

The unreliability score is computed by "triple-frame checking." Our system estimates both inter-frame homographies $\mathbf{S}_i(i+1)$ and homographies between spaced frames $\mathbf{S}_i(i+2)$. The checking score confirms that the result of two interframe homographies is similar to the result of the wider spacing by computing the difference matrix $\mathbf{S}_{i-1}(i)\mathbf{S}_i(i+1)\mathbf{S}_{i-1}^{-1}(i+1)$ and measuring the maximum displacement it makes to the four corners of the frame. Unreliability scores are velocity corrected based on the estimated velocities (magnitude of corner arc-length (§3.1)) of a window of neighboring frames. This correction is applied because errors in homographies are less noticeable when they are small relative to the overall motion.

Our approach does not actually form the mosaic image. This avoids issues in merging images. The transformations $\mathbf{S}_i(j)$ give us a common coordinate system, conceptually the coordinate system of the local mosaic, for the duration of the video. When the video is segmented into local mosaics, we denote the base frame for a particular video frame as $b(t)$.

3.3 Apparent Camera Motion

If we consider the mosaic to be the "world", then the viewpoint or camera is a transformation $\mathbf{C}$ that maps from the mosaic to the video (i.e. the camera's film plane). Assuming a standard camera model, $\mathbf{C}$ is a homography [31].

To view a frame of a video from a particular camera $\mathbf{C}$, the video frame using the stabilization transformation is then transformed to the screen by the camera:

$$\mathbf{M}(t) = \mathbf{C}(t)\mathbf{S}_{b(t)}(t).$$

If $\mathbf{M}(t)$ is the identity matrix ($\mathbf{I}$) the viewer sees exactly the source video frame. If $\mathbf{C}(t) = \mathbf{I}$ then the viewer is looking at the "center" of the mosaic. The inverse of the camera transform can be viewed as a quadrilateral on the base frame. Intuitively, this is the part of the world the camera "sees."

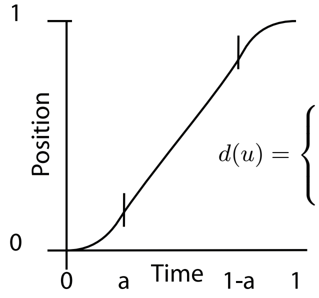
A video appears to have camera motion when static objects in the scene move. As the mosaic defines the static objects, the motion of $\mathbf{C}(t)$ creates a camera movement. Note that if we are viewing the original video, $\mathbf{M}(t) = \mathbf{I}$, so that $\mathbf{C}(t) = \mathbf{S_{b(t)}}^{-1}(t)$. In this case, the virtual camera motion is following the motion of the source video.

It is important to recognize that the result is $\mathbf{M}(t)$. The intermediate transformations, $\mathbf{C}(t)$ and $\mathbf{S}_{b(t)}(t)$, might be wild distortions, providing their composition produces a reasonable transformation.

The problem of re-cinematography is to determine a good $\mathbf{C}(t)$. Standard image stabilization chooses it to be a low-pass filtered version of $\mathbf{S}^{-1}(t)$ (although it is rarely implemented this way). Mosaic-based stabilization chooses $\mathbf{C}(t)$ to be constant.

3.4 Camera Motion

Rather than model the 3D geometry of the virtual camera (as in [20]), we create camera motions directly in the



$$d(u) = \begin{cases} \frac{u^2}{2a-2a^2} & u < a \\ \frac{a}{2-2a} + \frac{u-a}{1-a} & a \le u \le 1-a \\ 1 - \frac{(1-u)^2}{2a-2a^2} & u > 1-a \end{cases}$$

Figure 3: The ease curve accelerates with a constant acceleration, maintains a constant velocity, then decelerates.

space of the homography matrices $\mathbf{C}(t)$. The desire to control apparent velocity profiles of a 3D camera (Fig. 2) suggests a local linearization of the transformations using an exponential map. The idea, introduced to the graphics community by Alexa [1], takes the logarithm of the transformation matrices before applying linear operations (such as interpolation).

An intuition behind the use of the exponential map considers a camera moving with constant velocity. Between every frame, the transformation in the camera's local coordinate system is the same, for example 1 degree to the right. The inter-frame transform is applied by matrix multiplication, so if $\mathbf{A}$ is the initial transformation and $\mathbf{T}$ is the transformation between steps, after n steps the transformation would be $\mathbf{AT}^n$. To interpolate between transformations $\mathbf{A}$ and $\mathbf{B}$ from $t = 0$ to 1 with a constant velocity (in terms of camera rotation), we compute $\mathbf{A}(\mathbf{A}^{-1}\mathbf{B})^t$. Taking the logarithm of this expression would allow it to be re-written as:

$$e^{(1-t)\log \mathbf{A} + t \log \mathbf{B}}. \quad (2)$$

In other words, constant-velocity interpolation of transformations can be achieved by linear interpolation of their logarithms. However, because $\mathbf{A}$ and $\mathbf{B}$ are matrices, multiplication does not commute, and $e^{\mathbf{A}}e^{\mathbf{B}} \neq e^{\mathbf{A}+\mathbf{B}}$ [15], so this is only an approximation. [23] shows that although interpolation of matrix logarithms is not constant velocity, the non-constancy is small, bounded, and correctable if need be. In practice, the effect is too small to be visible in re-cinematography (except in numerical displays of the results such as Fig. 7).

Abrupt accelerations and decelerations of the camera are jarring to viewers. An instant acceleration to a constant velocity creates a noticeable artifact. To avoid this, we use an "ease-in/ease-out" function as given by [25] to warp time so that the interpolations begin with a period of constant acceleration, move with a period of constant velocity, and end with a period of constant deceleration. This function is shown in Figure 3. Alternatively, smoother splines can be used for interpolating the matrix logarithms. We will discuss this in (§4.5).

3.5 Lost Image Information

When we transform the image by $\mathbf{M}(t)$, we change it in potentially damaging ways. Here we consider the four types of damage to quantify the tradeoff between motion improvement and image damage. In general, "larger" transformations, i.e. farther from the identity, cause larger penalties. Identity transformations cause no penalty.

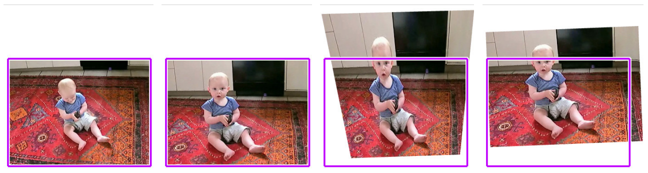


Figure 4: The projective transformation can stabilize the carpet between two frames (left 2 images), but stretches objects out of this plane (middle right). The rightmost image shows the similarity transform closest to the projective homography.

Offscreen Penalty: The transformation may move portions of the source image outside of the frame of the result. If we know that this portion of the image is not important, the loss is not significant. Conversely, if a portion of the source frame is identified as important, it should not be moved offscreen. The numerical value for this penalty (used in §4.3) is the percentage of the important area of the frame that is offscreen.

Truly measuring importance would require understanding of semantics and intent. To estimate importance, we use heuristic rules from image and video retargeting [30, 9, 25]: identifiable objects (especially faces) and visually salient regions are likely to be important. Our implementation detects faces using the method of [34]. For salience, research suggests that motion salience is a strong cue [29], corresponding with our intuition that moving objects (i.e. with motion relative to the camera motion) are likely to be interesting. We compute motion saliency as the contrast between the local motion vectors computed by optical flow, and the global motion vectors given by the homographies.

Uncoverage Penalty: The transformation might map places outside of the source image to inside the frame of the result, leading to empty areas. Left empty (usually black), not only are these areas ugly, but the edge moves with the high frequencies that have been removed from the camera motion. In-camera stabilization avoids these effects by capturing "extra" pixels as a buffer and restricting the movement. Software methods can use *inpainting* to fill these regions. Simple inpainting methods [24] copy pixels from neighboring frames, while newer methods such as [35] and [27] are more sophisticated. Our implementation provides simple inpainting. Neighboring frames (a 2 second window in either direction) are stabilized to the current frame and alpha-blended with more distant frames drawn first.

An alternative approach to address uncovered areas is to enlarge the source image, effectively creating a buffer (like the in-camera method). This approach has a cost as it creates more offscreen area. Determining appropriate scaling is considered in our algorithms in §4.

The uncoverage penalty associated with a transition is the percentage of the resulting frame that is uncovered.

Distortion Penalty: Non-uniform transforms may introduce unpleasant distortions. They may appear as unnatural viewing directions or will stretch portions of the scene that violate the projective assumptions as shown in Figure 4. To quantify the distortion of a transform, we compute the distance to the closest similarity transform, where the distance is defined by the corner displacements.

Resampling: The resampling used to implement the transformation may result in a loss of detail, especially if some amount of zoom is used. Given that much casual video is often presented in a small format (e.g. on a video sharing site, a portable media player, etc.), these details might be lost anyway. Our implementation does not explicitly consider resampling losses.

4. RE-CINEMATOGRAPHY

A re-cinematography system first pre-processes input video to perform motion estimation and important object identification. Given these three inputs, re-cinematography proceeds in the following steps. First, the video is broken into short segments, each corresponding to a local mosaic. Second, initial virtual camera motions are created for these segments that insure continuity between them. Third, the segments' camera motions are each optimized independently to best trade off image and motion quality. Finally, the camera motions are used to transform each frame, and in-painting is applied to fill uncovered regions. These steps are detailed in this section.

Tunable parameters in our system are denoted by $p_?$, with the standard values given. Experimenting with parameter settings is easy as the planning process is very fast (a few seconds for a two minute video). In practice, we have found that little empirical tuning has been necessary.

Because the source video footage tends to be of low quality, the information that can be extracted automatically is limited. Issues such as poor lighting, bad sensors and lenses, severe compression, and camera shake make casual video difficult for computer vision algorithms. Re-cinematography relies on limited information about the source video: in addition to the motion estimation used by stabilization, it also uses an assessment of the motion and motion salience as a predictor of what may be important in each frame.

4.1 Motion Analysis and Segmentation

The first step in the planning process is to break the source video into segments such that each is short enough that a single local mosaic can be applied. Segments are classified into three types: static segments, where there is little source camera movement; moving segments, where there is source camera movement; and bad segments, where the motion estimation is too unreliable to be used. The segmentation process tries to create groups of similar frames such that coherent camera motions can be used for each segment.

Bad homography segments are identified first by finding frames whose un-reliability scores (§3.2) are above a threshold ($p_{bh} = 5$ pixels). Frames within a short window ($p_{bhw} = \pm 2$ frames) are also considered bad. Bad frames are grouped into bad homography segments. For bad segments, re-cinematography sets $\mathbf{M}(t)$ equal to the identity as we cannot alter the bad frames reliably.

Next, the planner identifies static segments. The magnitude of the camera motion at each frame is measured by summing the corner displacements (§3.1) over a one second window. If there are no bad frames within that duration (for reasons explained below), the frame is considered static if its cumulative corner displacement and cumulative corner arc-length are both below thresholds ($p_{zxy} = .6$ and $p_{zs} = 1.2$ pixels per frame respectively). Consecutive static frames are grouped into static segments. Remaining frames are grouped into consecutive moving segments.

At this point, the video has been broken into a sequence of non-overlapping segments. Note that the sequence is constructed such that between any two static or bad seg-

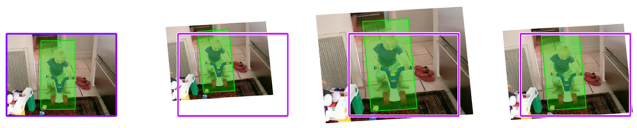


Figure 5: (a) The initial frame with its salience bounding box shown in green. (b) The proposed camera motion has a large uncovered region (on many of its frames). (c) Scaling to fill the uncovered region removes some of the salient image part. (d) The salience onscreen constraint prevents too large of a scale factor.

ments, there is a moving segment. This is important as there must be a transition between each of these segment types to maintain continuity. Our system imposes a limit (p_{ts} = 30 frames) on how short a moving segment can be to prevent motions that are too abrupt. Static segments are shortened if necessary to accommodate this restriction. Moving segments that connect to static ones use ease-in and -out (§3.4) to avoid abrupt starting and stopping.

The resulting segments may be too long to use a single local mosaic. The criteria of (§3.2) are applied: segments whose cumulative homography unreliability scores are above a threshold (p_{bhc} = 80), or whose cumulative corner displacement or arc lengths are greater than a threshold (p_{gbdx} = .6 and p_{gbds} = 3 respectively, units of screen width), are broken into shorter subsegments.

For each resulting segment, the base frame is the middle frame.

4.2 Static Segments

The next phase of the planning process determines the camera motions for each segment. Static and bad segments are considered first. Moving segments are determined subsequently as they must consider adjacent static and bad segments to create continuity.

For each static segment, our algorithm computes a single constant value for $\mathbf{C}(t)$ to be used over the segment's duration. The constant camera creates a "locked down" look for these segments where the camera was close to being static.

The camera matrix used, called the *framing*, is chosen to minimize the amount of uncovered area, even at the expense of causing offscreen areas (§3.5). The scaling is constrained to prevent important objects to be lost off screen. To compute the scaling, the system computes the minimum scale required to fully cover the segment by determining the intersection of the covered areas of all of the frames and computing the scale required to make this area fill the frame. Then, for each frame, a salience bounding box is determined. The scaling is reduced to insure that none of these boxes leave the frame. This process is shown in Fig 5.

To compute the salience bounding box for a frame t, we consider a window of frames $t \pm p_{sbw}$ (p_{sbw} = 8 frames). This implements the cinematographic principle of leading: it gives a moving object space so that the viewer can see where it is going. For each frame i in the window, its salience object box is transformed to t via $\mathbf{S}_t(i)$. A histogram counts the number of frames with salience in each region of the image. The salience bounding box is determined from the histogram cells with counts above a threshold.

Because the total amount of motion in a static segment is small (which is why the segment was identified as static), the amount of distortion applied to any frame is necessarily small, so the system need not consider distortion on static segments. Similarly, offscreen penalties are not an issue for static segments because the only reason that an important object would be moved offscreen is if it were close to the image border.

4.3 Keyframing Moving Segments

For moving segments, we create $\mathbf{C}(t)$ by keyframing and using matrix logarithm interpolation, possibly using ease curves, to provide velocity-profiled paths. A first step determines keyframes for the beginning and end of the segment that provide continuity with neighboring segments. The boundary keys are interpolated directly to provide an initial path that will be refined in the next section.

Let the frame index b_i be the base frame of the moving segment, b_{i+1} be the base frame of the next segment, t_b be the beginning of the moving segment, and t_e be the end of the moving segment (so $t_e + 1$ is the beginning of the next segment). We explain how the end of the segment is chosen; the beginning is done similarly.

If the next segment is a bad segment, then $\mathbf{M}(t_e+1) = \mathbf{I}$. We choose $\mathbf{M}(t) = \mathbf{I}$ because without reliable homography information, the best we can do for the boundary is to mimic the motion of the source video.

If the next segment is a static segment, then we need to choose the camera keyframe to be in the same position as it will be during that static segment. Let the framing of the next segment be $\mathbf{F}_{i+1}$. Since this framing is relative to $b+1$, it must be transformed

$$\mathbf{C}(t_e) = \mathbf{F}_{i+1}\ \mathbf{S}_{b+1}(t_e)\ \mathbf{S}_b^{-1}(t_e). \tag{3}$$

This expression computes the key by transforming the goal framing into the video coordinate system, then transforming it into the coordinate system of the segment.

Insuring continuity with a neighboring moving segment is similar to the static case, except that rather than considering a static framing, we must connect to the keyframed path. The latter of the neighboring segment is keyframed first, without regard to the prior one. Then the end keyframe of the prior one is chosen by first extrapolating the camera for the latter segment to determine where the camera would have been on frame t_e if t_e were in that segment. This extrapolated transform is used in place of $\mathbf{F}_{i+1}$ in Eqn. 3.

4.4 Placing Keys to Optimize Movement

Interpolating between beginning and ending keys creates camera movements with direct, velocity-profiled paths. Such paths are ideal in terms of our goals for motion (§1.2), but may deviate considerably from the actual camera motion. These large deviations lead to transformations that are far from the identity and therefore may have large per-frame image loss penalties (§3.5). We insert keyframes into the segment's camera motion to balance between image and motion quality. Figure 6 shows an example.

Inserting a keyframe has a benefit to image quality, but a potential cost to motion quality. For example, inserting a new keyframe at time j $\mathbf{C}(j) = \mathbf{S}_b^{-1}(j)$ makes $\mathbf{M}(j) = \mathbf{I}$ so the penalty on the frame is zero. Frames close to j (between j and the adjacent keyframes) will also have their penalties reduced. On the other hand, inserting the key frame breaks the directness of the camera's path.

Our strategy for choosing which keys to insert is greedy: our algorithm identifies the frame with the worst image loss and inserts a key, repeating until their are no more

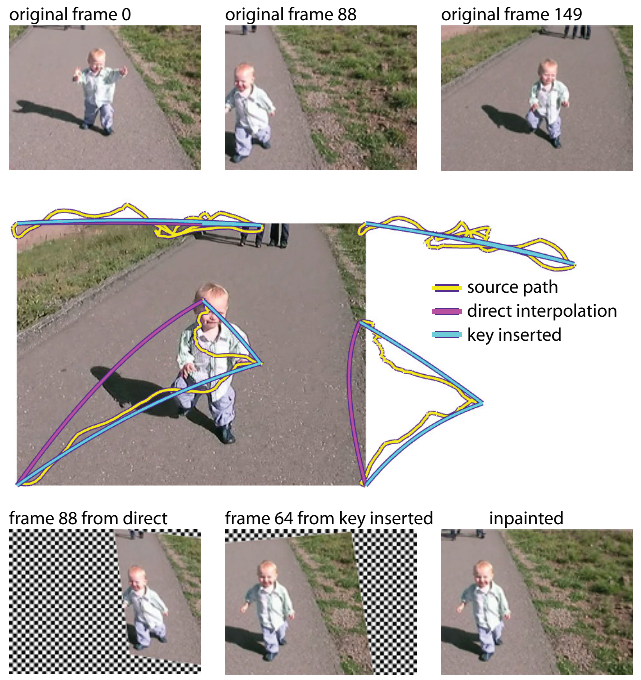


Figure 6: 5 second segment walking backwards tracking a toddler. Both subject and camera are weaving significantly. Direct interpolation provides a smooth path, but creates an extremely uncovered and distored frame. Inserting a key at this frame causes it to appear as in the source. The smaller problems that remain can be addressed with inpainting.

frames with bad enough losses to be considered a problem. The algorithm is constrained to keep a minimum spacing between keys ($p_{minKeySpace} = 30$ frames). The simple algorithm's parameters are the minimum key spacing ($p_{minKeySpace} = 30$ frames), and the thresholds below which penalties are not considered a "problem." There are separate penalty thresholds for salience offscreen ($p_{so} = 0$ i.e. it is unacceptable for any important region to be moved offscreen), distortion ($p_d = 20$ pixels of corner movement), and uncovered area ($p_{uc} = 15\%$ of the screen area). Rather than determining scaling factors between the different penalty terms of §3.5, we choose a strict ordering: we choose salience offscreen problems first, distortion penalties second, and uncoverage penalties last. Note that since the penalties are created by large transformations, frames with high values for one penalty often have high values for the others.

A more complex algorithm that explicitly measured the cost benefit ratio of different key placements offered little benefit over the simple greedy algorithm described.

While choosing the keys such that $\mathbf{C}(j) = \mathbf{S}_b^{-1}(j)$ minimizes the error on frame j, it is not necessarily the best choice to avoid penalties on in-between frames. In particular, uncoverage penalties on non-key frames can be reduced by scaling up the keyframe images, just as with static shots. We use the same method to choose a scaling that prevents losing important information. We use a single scale factor for all keys in the segment to avoid wobbles in the zoom. However, the use of a single scale is often insufficient as different parts of the movement may exhibit different problems, and the single scale must be chosen for the worst case over the whole duration.

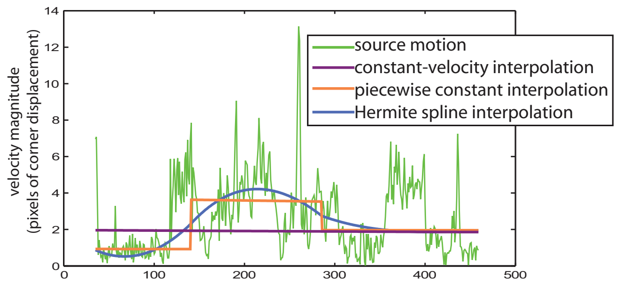


Figure 7: Graphs of the magnitude of the camera velocities for the paths in Fig 8. The green line measures the velocities of the source motion, purple is a constant-velocity interpolation between the endpoints. The orange and blue paths add 2 keyframes and interpolate with piecewise-constant-velocity and Hermite splines respectively.

4.5 Interpolation Types

Inserting a key into a constant-velocity interpolation creates a velocity discontinuity. If the direction does not change, the discontinuity is hard to notice even if there is a large discontinuity in magnitude. We explain this by observing that most of the velocity discontinuities are about the same magnitude as the velocity changes that occur between each pair of frames in the source motion (see Fig 7). Directional changes in velocity are noticable, but often seem intentional. This is because a change in virtual camera motion direction is almost always caused by a corresponding change in the source camera's motion, which was probably motivated by something. Meandering source paths become crisply directed virtual camera paths, with similar targets.

Velocity discontinuities seem to be noticeable and objectionable when the camera starts or stops. Therefore, we apply an ease curve (§3.4) when a moving segment connects to a static one.

An alternative to constant-velocity interpolation, velocity-continuous interpolation is implemented using Hermite cubic splines to interpolate the matrix logarithms. While these splines remove velocity discontinuities, they also remove the velocity constancy. The tradeoff is subjective, and the effects are surprisingly subtle. Our implementation uses constant-velocity interpolation by default. An example of camera paths is illustrated in Fig 8. The corresponding velocity profiles are shown in Figure 7.

5. EXPERIMENTS

Our re-cinematography prototype is implemented on PCs running Windows XP using compressed AVI files. Motion estimation, homography evaluation, and motion salience are computed as a pre-process. Our implementation of these standard components is inefficient. State of the art systems provide near real-time performance [4, 28].

The main steps of re-cinematography are performed in our interactive system and are all much faster than real time. The system caches the video in texture memory for efficient playback and precomputes all $O(n^2)$ stabilizations, meaning that memory becomes an issue after (approximately) 2700 frames on a machine with 2G of memory. After the video is read from disk (which happens faster than real time) none of the steps (including the $O(n^2)$ step) take more than a few

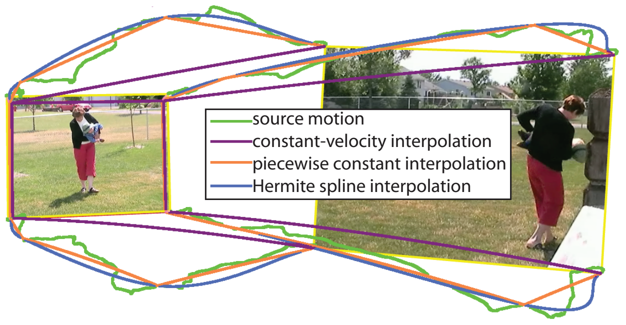

source motion
constant-velocity interpolation
piecewise constant interpolation
Hermite spline interpolation

Figure 8: Camera paths illustrated for a moving segment in the coordinate system of its base frame. The green path shows the movements of the corners of the image from the source motion. The purple paths show the constant-velocity interpolation between the endpoints. The orange and blue paths add 2 keyframes and interpolate with piecewise constant-velocity and Hermite splines respectively.

seconds (for an entire 2 minute video), when the data fits into memory.

Most of our test data (and all of the non-synthetic examples in this paper) were recorded using a Sanyo Xacti C6 camera. It records Quicktime MPEG4 at 640x480 at 30 frames per second. All examples use automatic white balance and exposure. Most examples were originally shot using the camera's digital image stabilization, which does degrade input quality. We have also done tests using video from a Motorola V3xx cell phone that records video in the 3GP format (176x144, 15fps, highly compressed).

Because most of our examples were shot with in-camera image stabilization, the re-cinematography results are effectively a comparison with current technology. For a more fair comparison, we have also implemented the low-pass filter method from [27], however we apply it within our system using the same motion estimation and in-painting implementation as used for re-cinematography. We also compare our results with *DeShaker* [11], a post-process image stabilization program.

5.1 Contrived Tests

In addition to real examples, we experimented with video created specifically for testing. These examples illustrate the differences between re-cinematography and filtering. Some examples were footage shot in our lab, while others were created synthetically using a compositing tool. The latter have the advantage that we can compute the homographies directly, without relying on motion estimation.

Figure 9 shows a synthetic example simulating a videographer filming a UFO weaving over a city. The "videographer" tries to keep the saucer centered (i.e. following the bobbing), but has an unsteady hand (simulated by adding white noise). Image stabilization methods can remove the noise, but leave the smooth (approx 1hz) zig-zagging of the camera, which can be quite sea-sickness inducing - even with a larger filter kernel than suggested in [27]. Also, the saucer is no longer tracked perfectly, and has an unsteady wobble. The re-cinematography result has a camera that moves in straight pans, leaving a smoothly weaving saucer. The resulting camera path began as a direct interpolation between the beginning and end position, with a key inserted where the salient object was maximally out of the frame. Because

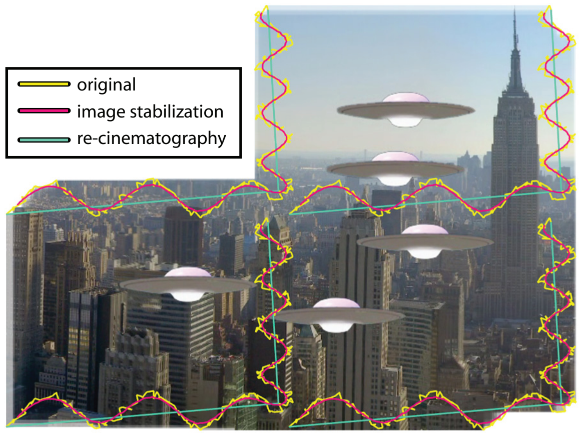


Figure 9: Synthetic video example of a flying saucer zig-zagging over New York. The video simulates a videographer tracking the saucer. Lines in this mosaic image show the paths of the corner of the video images. Yellow is the original (noisy) path. Magenta shows the result of low-pass filtering from [27]. Cyan shows the re-cinematography result.

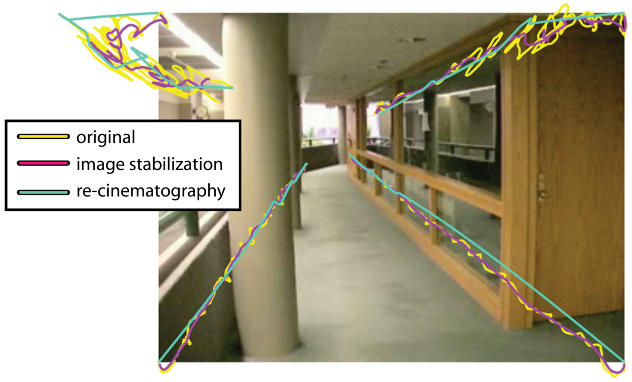


Figure 10: Video taken walking down hallway. Image corner paths shown using the first frame as base. Yellow is the original motion, Magenta is filtered using the method of [27], and Cyan is the re-cinematography solution.

of the noise, the key is inserted such that the path of the camera is not precisely horizontal.

The smooth oscillations appear in real examples as well. Figure 10 shows a video taken while walking down a corridor. With standard image stabilization, the oscillations due to the videographer's stride are very apparent. Our approach removes these oscillations. Also note that the forward motion is not well represented by the homographic model, yet our system can still create a good result by inserting keyframes periodically.

5.2 Evaluation

We have experimented with our implementation on a collection of home videos. The results are difficult to convey in still images. They exhibit very stable static segments, and motions appear directed and intentional. Pictures from these examples appear throughout the paper, including Figures 6 and 13.

The moving segments produced by the system achieve their intended effect. The desirability of these effects is a subjective question. Our initial informal evaluation suggests

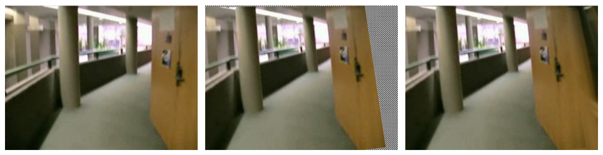


Figure 11: A frame from near the end of the example of Fig 10 that exhibits the worst penalties in the result. Left: original. Center: result without inpainting. Right: simple inpainting is ineffective as the homographies poorly model the actual motion.

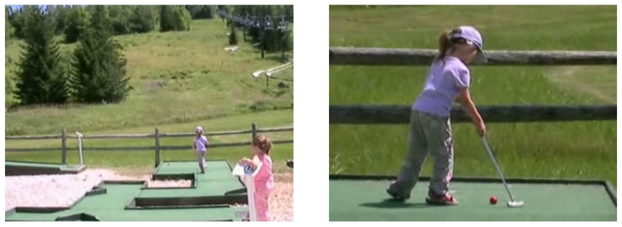


Figure 12: Beginning and end frames of a zoom segment shown in Fig 13.

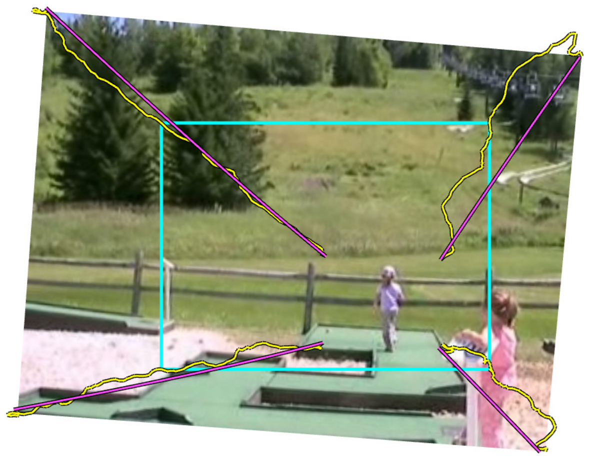


Figure 13: Visualization of re-cinemtography of zoom segment (Fig 12). The first frame is shown transformed to the base frame (whose boundary is shown in Cyan). Yellow paths trace the corners of the original video, magenta paths (darker) trace the corners of the resulting camera motion.

that viewers appreciate the "improved" motion, especially when the original source video was shaky. However, this may just be confirming what camera manufacturers already know: that image stabilization is a good thing.

We have compared our results with *DeShaker* [11], a post-process image stabilization program. That software uses a filtering-based approach and is able to remove the high-frequency movements on almost all of our examples. Re-cinematography offers a clear advantage in making the static segments more stable. For videos with static segments connected by short motions (which are very common among our sample videos), the stable static shots with directed connections in the re-cinamatography results are particularly compelling.

For moving segments, the differences between the systems are more subjective. DeShaker's smooth motions are quite different than the point-to-point motions produced by our approach. In both systems, the worst artifacts are the uncovered screen areas as neither system offers an advanced in-painting scheme (c.f. Fig 5.1). DeShaker's implementation of scale to cover is problematic as it varies the scale too rapidly causing rapid oscillations in size (this problem is noted in the manual). At the opposite extreme, our scaling solution (§4.2) picks a single scale factor for each segment (to avoid oscillations in size) which is often insufficient, especially since the scale is chosen conservatively. An advanced inpainting method such as [27] or [35] is a better cure for uncovered regions.

5.3 Failures and Limitations

Re-cinematography creates a new video by transforming an existing one. This places a number of fundamental limitations on the technique. Because re-cinematography is limited to transforming the frames of the source video, if it is not in the source video, re-cinematography cannot put it into the result. Furthermore, the amount of change that can be applied to a frame is limited, which in turn limits the amount of change that can be made to the motion.

A richer class of transformations (as in [8]) would allow larger changes without distortion. However, poor quality source video makes estimating rich motion models difficult, and interpolation of more complex transformations may be problematic. Our simple inpainting implementation also limits the amount of change that can be made to a frame. Recent inpainting methods [27, 35] would provide improved performance.

Another limitation in re-cinematography is the reliance on motion and importance estimation. While state of the art methods may provide better performance than our prototype, no method can be completely reliable as some video is inherently ambiguous.

Given the limitations, we recognize that re-cinematography cannot help all videos. We consider a result from our system a failure if it produces a result that is worse than the source. The system should recognize places where it cannot help.

One category of failure is when the system incorrectly assesses a bad homography as reliable. This leads to transformations that give visual artifacts, including distortions and small jiggling. These errors come from poor motion estimation and scenes not well modeled by the projective assumptions. In future work, we believe that it is possible to make a reliable assessment of both of these kinds of errors and to use this to restrict the amount of transformation used.

Another class of failure is when the system makes a bad tradeoff, for example, choosing to show a large uncovered area rather than a less smooth motion. Part of the problem is that these tradeoffs often depend on the image, the application, and the viewer's preference. A related class of failure is when the system misses an opportunity to make an improvement because it is being too conservative.

Our approach, like any stabilization approach, trades off in image quality to achieve better motion quality. There is an implicit assumption that some loss of image quality is worthwhile to gain an improvement in motion. Re-cinematography discards image detail, particularly when it zooms in. Given that much casual video is often presented in a small format (e.g. on a video sharing site, a portable media player, etc.), these details might be lost anyway.

6. DISCUSSION

Our re-cinematography approach can improve the camera dynamics in a wide range of casual videos. Wandering hand-held motions are turned into smooth directed movements that exhibit ease-in and out where appropriate, and the camera stays rigidly fixed when this is similar enough to the source movement. Better evaluation is important to understand the performance and utility of our system.

Our current system is almost completely automated. A more appropriate use of the technology may be as a more interactive authoring tool where a user has more control over the process. A user can make decisions about quality tradeoff parameters based on their preferences and knowledge of the video's content and application. The user can also provide high level insight based on the semantics of the video and the desired storytelling in the result. By definition, this latter information cannot be obtained without the user.

Ultimately, our approach is limited by the quality of the source footage. No post-process is likely to be a replacement for good planning, skillful camera work, proper camera supports, and artistic talent. Re-cinematography will not turn casually captured video clips into masterpieces of film. However, the techniques presented in this paper can effectively improve the apparent camera dynamics in casual video.

Acknowledgements. We thank Rachel Heck for her assistance in preparing demonstrations. The city background image in Figure 9 is by S. Schultz and used under a Creative Commons license. This research was sponsored in part by NSF grant IIS-0416284.

7. REFERENCES

[1] M. Alexa. Linear combinations of transformations. In *SIGGRAPH '02*, pages 380–387, 2002.

[2] D. Arijon. *Grammar of the Film Language*. Silman-James Press, 1991.

[3] E. Bennett and L. McMillan. Proscenium: a framework for spatio-temporal video editing. *Proceedings ACM international conference on Multimedia*, pages 177–184, 2003.

[4] A. Bevilacqua and P. Azzari. High-quality real time motion detection using ptz cameras. In *AVSS '06: Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*, page 23, 2006.

[5] B. A. Block. *The Visual Story: Seeing the Structure of Film, Tv, and New Media*. Focal Press, 2001.

[6] D. Bordwell and K. Thompson. *Film Art: An Introduction*. The McGraw-Hill Companies, Inc., 1997.

[7] B. Brown. *Cinematography: Theory and Practice: Imagemaking for Cinematographers, Directors & Videographers*. Butterworth-Heinemann, 2002.

[8] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *IEEE CVPR*, volume 2, pages 609–614, 2001.

[9] L.-Q. Chen, X. Xie, X. Fan, W.-Y. Ma, H.-J. Zhang, and H.-Q. Zhou. A visual attention model for adapting images on small displays. *ACM Multimedia Systems Journal*, pages 353–364, 2003.

[10] M. Christie, R. Machap, J.-M. Normand, P. Olivier, and J. Pickering. Virtual camera planning: A survey. In *Proceedings Smart Graphics*, pages 40–52, 2005.

[11] Deshaker. www.guthspot.se/video/deshaker.htm, 2006.

[12] R. Dony, J. Mateer, and J. Robinson. Techniques for automated reverse storyboarding. *IEE Journal of Vision, Image and Signal Processing*, 152(4):425–436, 2005.

[13] A. Girgensohn, J. Boreczky, P. Chiu, J. Doherty, J. Foote, G. Golovchinsky, S. Uchihashi, and L. Wilcox. A semi-automatic approach to home video editing. In *UIST '00*, pages 81–89, New York, NY, USA, 2000. ACM Press.

[14] M. Gleicher. Projective registration with difference decomposition. In *IEEE CVPR*, pages 331–337, 1997.

[15] V. M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *IEEE CVPR*, 2004.

[16] R. Heck, M. Wallick, and M. Gleicher. Virtual videography. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2007.

[17] X.-S. Hua, L. Lu, and H.-J. Zhang. Optimization-based automated home video editing system. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):572–583, May 2004.

[18] M. Irani and P. Anandan. Video indexing based on mosaic representations. *Proc. IEEE*, 86(5):905–921, 1998.

[19] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. *International Conference on Computer Vision*, pages 605–611, 1995.

[20] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *IEEE CVPR*, pages 454–460, 1994.

[21] O. Johnston and F. Thomas. *The Illusion of Life: Disney Animation*. Abbeville Press, 1981.

[22] S. D. Katz. *Film directing shot by shot: visualizing from concept to screen*. Michael Wiese Productions, 1991.

[23] L. Kavan, S. Collins, C. O'Sullivan, and J. Zara. Dual quaternions for rigid transformation blending. Technical Report TCD-CS-2006-46, Trinity College Dublin, 2006.

[24] A. Litvin, J. Konrad, and W. Karl. Probabilistic video stabilization using kalman filtering and mosaicking. In *Proc.IS&T/SPIE Symp. Electronic Imaging, Image, and Video Comm.*, pages 663–674, 2003.

[25] F. Liu and M. Gleicher. Video retargeting: Automating pan and scan. In *ACM Multimedia 2006*, pages 241–250, 2006.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[27] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE PAMI*, 28(7):1150–1163, 2006.

[28] D. Nistér. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005.

[29] R. Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157–3163, 1999.

[30] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic thumbnail cropping and its effectiveness. In *UIST '03*, pages 95–104. ACM Press, 2003.

[31] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, 1996.

[32] R. Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, 2006.

[33] L. Teodosio and W. Bender. Salient stills. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(1):16–36, 2005.

[34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE CVPR*, pages 511–518, 2001.

[35] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE PAMI*, 29(3):463–476, 2007.

[36] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin. Multiperspective panoramas for cel animation. In *SIGGRAPH '97*, pages 243–250, 1997.