Towards Moment Imagery: Automatic Cinemagraphs

James Tompkin, Fabrizio Pece, Kartic Subr, Jan Kautz

University College London

{j.tompkin, f.pece, k.subr, j.kautz} @ cs.ucl.ac.uk

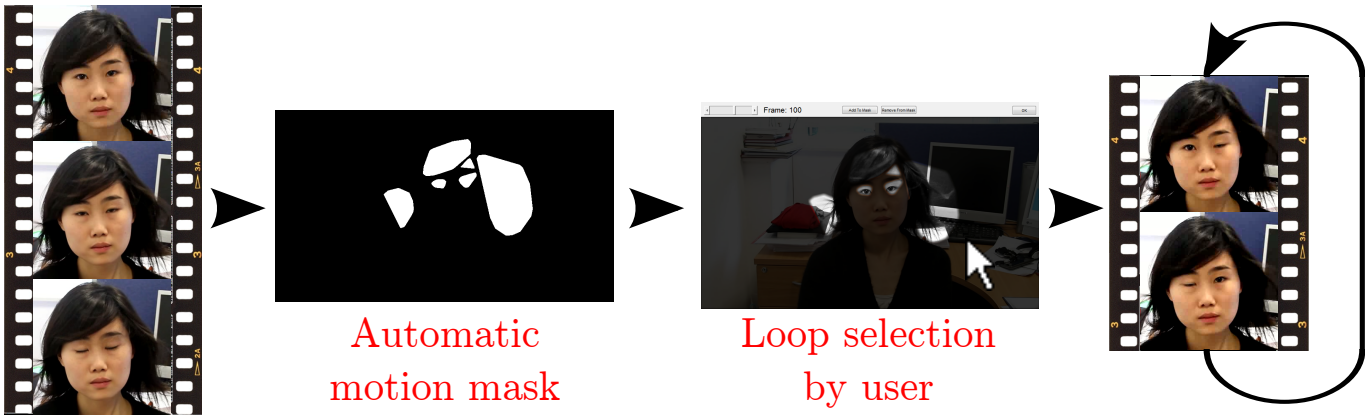

Figure 1: Our cinemagraph creation process. A cinemagraph is a short video that is akin to a still photograph but highlights specific dynamic element by seamlessly looping through selected motions. In this example, input is video of a blinking person with wind-swept hair. We register the frames if needed, automatically find the mask for regions of motion, and ask the user which regions to loop. The user freezes the hair and loops the eyes individually at different times to create a winking effect.

Abstract

The imagination of the online photographic community has recently been sparked by so-called cinemagraphs*: short, seamlessly looping animated GIF images created from video in which only parts of the image move. These cinemagraphs capture the dynamics of one particular region in an image for dramatic effect, and provide the creator with control over what part of a moment to capture. We create a cinemagraphs authoring tool combining video motion stabilisation, segmentation, interactive motion selection, motion loop detection and selection, and cinemagraph rendering. Our work pushes toward the easy and versatile creation of moments that cannot be represented with still imagery.*

Keywords: cinemagraphs, videos, video textures

1 Introduction

Capturing the dynamics of a moment is an intriguing problem [4]. Imagine the moment when two people shake hands. A photograph can draw attention to the specific moment, but the dynamics of the hand-shake are lost. A video would capture the dynamics, but drawing attention to the hand-shake is difficult as it only lasts a fleeting moment. Recently, online photographic communities have been abuzz with the creation of *cinemagraphs*: short, seamlessly looping videos (usually stored as GIF images), where only parts of the scene are animated to emphasize select dynamics in the scene.

The construction of cinemagraphs from video sequences includes several challenges. First, for video captured with a translating or rotating camera, the motion in the image does not correspond to the motion of objects. Next, objects undergoing motion must be isolated from static parts of the scene. Finally, creating a seamless loop from the video sequence is non-trivial as even objects with repeating motions often deform or move within the frame. Currently, the process of creating cinemagraphs requires careful scene and actor setup to ensure that loops are possible. The footage then needs to be manually masked per-frame. The masks and hand-selected frames are then composited using image-editing software and exported as an animated GIF image.

In this paper, we present an automated tool which simplifies the authoring of cinemagraphs and expands the range of footage that is suitable for creating cinemagraphs (shown simplified in Figure 1). Our system aims to be efficient in terms of computational cost and memory, as the envisioned platform for this method is smart phones. The system functions in several stages. First, we register the input video sequence to remove shake. This first step separates object motion from camera motion, and all background pixels are made static in a common image frame. Next, we automatically isolate regions of

object motion in the registered video sequence, and allow easy definition of still and under-motion regions in the output. Then, we find independent loops per region and ensure that loops meet seamlessly by interpolating new frames. Lastly, motion regions are composited to the desired length, including the addition of dramatic pauses, to create a cinemagraph.

2 Related work

Our algorithm can be broadly divided into three main steps: software video stabilization, motion detection and segmentation, and video loop generation.

2.1 Video stabilization

Two-dimensional video stabilization methods are now popular in consumer video editing packages [16, 13]. These algorithms estimate camera motion in the 2D image plane motion and zoom or crop to compensate. This motion can be evaluated in a variety of ways, including optical flow, stable feature points, and block-based cross-correlation. These methods estimate a global 2D transformation (such as affine or projective), and so fail to correctly register when the camera undergoes translation. 3D video stabilization takes this a step further by finding stable 3D feature points by structure-from-motion and applying image-based or warping techniques to cope with some parallax [9].

Liu *et al.* simplified 3D stabilization methods to achieve state-of-the-art performance. They attempt to smooth camera motions to make hand-held camera footage more pleasing to watch. Their most recent work [10] decomposes feature point tracks in sequences to find eigentracks. Smoothing the eigentracks, reconstructing the feature points, and feeding these into a temporally-aware local warp produces automatic stabilization.

2.2 Motion detection and segmentation

Motion detection and segmentation is a fundamental technique for analysing image sequences. While motion estimation detects regions of interest where motion happens or is likely to happen, motion segmentation "compresses" the sequence into sets of pixels moving coherently across the sequence with associated motions. Estimating and segmenting motion in captured scenes is a non-trivial task and, even though this research field has been very active for the past decades, the problem of detecting and segmenting motion is still open. Generally, this task is subdivided into two steps: identifying areas of motion, and segmenting and tracking moving objects.

For estimating areas of motion, typical methods can be categorized into pixel-based methods ("direct") and feature-based methods ("indirect",). These include, but are not limited to, pixel-intensity variance (background subtraction), optical flow, image frequency analysis and block-matching algorithms. There is a large body of literature in this area, and we can only cite a few exemplary techniques. Methods based on background subtraction [18] are commonly used due to their simplicity. For instance, Reinhard *et al.* [20] propose a method based on pixel-intensity variance for High Dynamic Range (HDR) imaging of dynamic scenes. Similarly, Pece and Kautz [17] introduce a method based on Median Threshold Bitmaps (MTB, [28]) to detect clusters of moving pixels within a bracketed exposure sequence using simple binary operations. Manzanera and Richefeu [12] improve upon standard variance-based motion estimation by accounting for temporal and local variations (e.g., lighting and camera shake) in the context of video surveillance. In contrast to these methods, we do not require pixel-accurate motion estimation as we copy large regions from our registered video volume.

A different approach to detect motion in a sequence is by analysing its optical flow. Optical flow describes the pattern of apparent motion of objects, surfaces, and edges in a scene [1]. Once the optical flow is computed, motion areas are found by analysing the flow field (e.g., by thresholding). This approach is also often used for estimating a motion-free background plate. Wixson [29] detects motion by integrating frame-to-frame optical flow over time. The author defines the salient motion as the motion that tends to move in a consistent direction over time. The saliency measure used is directly related to the distance over which a point has traveled with a consistent direction. Gutchess *et al.* [6] employed motion as metric to initialize their background model. The authors divide the pixels into temporal subintervals with similar values and then the "best" subinterval belonging to the background is found as the subinterval with the minimum average motion estimate. Mittal and Paragios [15] also employ optical flow for their motion-based background subtraction model. The authors use optical flow to describe the dynamic characteristics of the scene as features in a higher dimensional space. An estimate of the probability that the observed data belongs to the background defines a segmentation.

With motion estimated, segmentation (and tracking if needed) is now possible. One approach, based on multi-resolution image pyramids, is introduced by Irani *et al.* [7]. However, this solution does not involve any sort of shape tracking or modeling, so information about scene events is not available. Meyer and Bouthemy [14] based their segmentation step on the work of Bouthemy and François [2], and use object outlines (i.e., a polygonal representation) to track them. Smith and Brady [24] introduce an optical-flow-based system to track and segment objects. The system starts by tracking 2D image features over time as they move across the image plane. This flow field is segmented into clusters which have both internally consistent and unique flow variation, and a different flow than the background. Tweed [26] describes novel techniques that consider occlusion and multiple moving regions. Basing his work on analysis within local windows and using iterative refinement, the method can relate motion types and occlusion effects. This allows the author to easily classify motion configuration, and thus segment the objects through an object graph where nodes correspond to coherently moving sub-regions and components to global moving objects. However, in our work, we do not currently track objects in motion. Instead, we find areas occupied by dynamic objects. This results in motion areas where multiple objects may simultaneously appear when motion tracks overlap in image space.

2.3 Video loops generation

Producing a cinemagraph requires the creation of a finite duration video that can be played continuously without any visible discontinuities. However, this task is not trivial, and requires the original video to be split into independently moving regions, each analyzed and rendered independently. It may also require loop closure with new synthesized frames.

Arguably, the most influential work in video loop generation is by Schödl *et al.* [23]. Their paper on *video textures* describes the generation of infinitely looping video from stocastic or periodic video. The method finds frames in the input where a "jump" to a different frame in the sequence can be made without noticeable artifacts. Using these jumps, the authors are able to generate infinite loops from a finite length video. New frame orderings are generated either randomly (a Monte-Carlo stochastic technique) or by obtaining a subset of the original frames that will guarantee a loop after a specific number of frames. Rendering video textures generally includes blending between jump frames to prevent temporal artifacts. Our pipeline differs from video textures as we are not interested in creating random videos (i.e., texture-like videos like waterfalls), but instead try to find individual coherent movements that can be repeated.

Bregler *et al.* [3] introduced similar work called *video rewrite*. Here, the authors extract a video sequence of a person's mouth from a training sequence of the person speaking, then reordered to match a new phoneme sequence from audio. Pollard *et al.* [19] introduced the term *video sprite* for applying 3D view interpolation to an alpha-matted region of a video rather than to the whole image. Similarly to this, Finkelstein *et al.* [5] alpha-matte video elements in their multi-resolution video work, naming them *video clip-art*.

3 Approach

Our approach is summarized in Figure 2. Given a video sequence, we first stabilize the video if necessary to remove any motion in background regions (Sec. 3.1). Our system then identifies regions of motion as candidates for looping (Sec. 3.2). The user selects a subset of these regions to be animated, and with this selection identifies an important frame to be included in the motion (Sec. 3.3). We then automatically find loops around these user-identified important frames (Sec. 3.4), including synthesizing new frames to ensure smooth looping. The output of our system is a cinemagraph with only the chosen subset of regions in motion.

We assume that the input footage is taken either hand-held or with a tripod, and that any camera motions are either rotations or small translations. We also assume that at least some of the background scene in the footage is not moving, and that any lighting variations across the sequence are small. The remainder of this section explains our algorithm.

3.1 Stabilization

First, we ensure that the video sequence is stable across frames. We make the assumption in later stages (Sec. 3.2) that background pixels do not move, so here we try to remove as much ego-motion as possible. Tripod shots are not stabilized unless they intentionally rotate, and any minor shake of the camera while on the tripod (creating small motions at image edges) is effectively filtered out in later stages.

We tested existing algorithms in off-the-shelf stabilization tools to see whether they were suitable for our task. We set both Deshaker (a 2D stabilizer plugin for VirtualDub) and the Warp Stabilizer in Adobe After Effects (an implementation of the recent work by Liu *et al.* [10]) to remove all motion to register our videos. Figure 3 shows a comparison on mean images. Unfortunately, both systems failed to remove all motion in the background. The warp stabilizer was confused by small object motions and introduced distortion into the background. Instead, we find KLT feature tracks through our video volume. We reject tracks with RANSAC that suffer large reprojection errors ($d > 0.8$) after fitting a standard camera model. Then, we locally warp the volume on a frame-by-frame basis using as-affine-as-possible moving least squares [22]. As the KLT tracks are temporally coherent and do not follow moving objects, this successfully registers background pixels in our examples. This is similar in spirit to the recent method of Ryu *et al.* [21], though we employ a more advanced warping technique.

3.2 Segmentation

As discussed in Sec. 2.2, motion detection and segmentation is an unsolved problem with many intricacies. We wish to create a *fast* and *robust* tool, but not to necessarily separate overlapping motion regions. Many methods that attempt complex motion segmentation are computationally expensive, and sometimes fail to produce sufficiently accurate occlusion boundaries. This would lead to strong artifacts in the output. We feel our approach of not separating overlapping motions is a suitable compromise which does not lead to these kinds of artifacts.

There are many ways to identify objects in motion in registered video. We tested Chebyshev distance (maximum absolute differences between each frame and the mean image across all frames), accumulated variance [23], accumulated optical flow [29], blur detection on the mean image (where smooth regions signify motion from averaging different pixels) [8], structural similarity (SSIM) [27], visual difference prediction (VDP) [11], and bitmap motion detection (BMD) [17].

Since producing quantitative differences for motion masks is somewhat ill-defined, as motion difference is a perceptual property, we provide a qualitative comparison (Fig. 4). Anecdotally, the Chebyshev distance provided a good quality mask in the fastest time (33ms, all timings per 960x540 frame in MATLAB averaged over a 434 frame video). SSIM provided a conservative mask, but computation times were 3x longer than Chebyshev (106ms). Accumulated variance failed to detect motions which covered the background only briefly over long

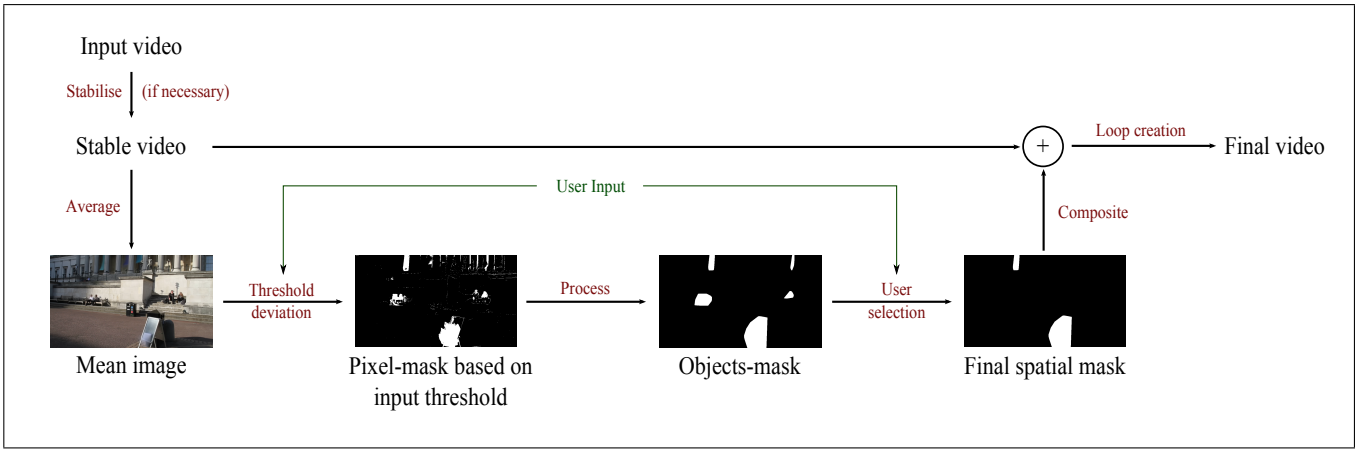

Figure 2: Algorithm summary. Video is stabilized only if necessary, but all other operations always occur. Loop creation includes interpolating new frames to seamlessly close loops.

sequences (67ms), and blur detection failed similarly while being computationally expensive (3516ms, though it produced better results than other approaches in lightly textured regions such as clothing). Accumulated optical flow produced good results at edges but is more expensive (212ms), BMD was too sensitive to noise (45ms), and VDP, which as given is not intended for this use, was too sensitive to any changes at all (7275ms).

For time-critical applications, such as on a mobile device, the Chebychev distance provides a good mask quickly. For less constrained applications, we would use SSIM. For both, we compute an initial threshold value one standard deviation above the mean distance, and obtain a binary mask that identifies pixels with underlying motion.

The pixel-level mask produced by the thresholding step is processed to represent locally moving objects by removing small elements ($area < 0.1\%$ of the image size), filling holes, and computing the convex hull of connected components. We then present these conservative regions to the user for selection.

3.3 User input

The user is shown moving regions for selection as desaturated regions against a coloured faded background (Fig. 5). Scrolling the mouse wheel changes the threshold used for the binary mask, which quickly adds or subtracts motion regions. This is not usually necessary, and is intended for small motion regions or for artistic effects. If desired, the user can go further and paint on the mask to make fine adjustments. The user is free to move through frames to see the effect of the mask on the video volume. When left clicking on a highlighted region, the user indicates that this particular frame is a key frame to be included within a moving region in the cinemagraph. In this way, they can be sure to capture a specific part of the sequence in an output loop. When right clicking, the user requests that region to freeze on a particular frame for the duration of the cinemagraph. Although we provide options for control and fine adjustment, unless desired, the user need only click once per region to select moving regions for the creation of a cinemagraph.

3.4 Loop generation

Once motion regions have been selected and the mask computed, the next step is to generate a seamless, looping sequence from the stabilized video. We compute the sum of squared difference (SSD) matrix [23] for each masked motion region. The SSD signifies similarity between frames in the motion region, i.e., a loop without temporal artifacts requires a small SSD between the first and the last frame. The user has specified a key frame that they wish to be in the looping motion region, so we use this as a starting point for a loop search within the matrix. We perform an iterative search over loop length and SSD thresholds. We start with a small SSD threshold (1% of max difference of region area), and search increasingly large loops around the key frame. If a loop is found with SSD less than the threshold then we keep it. We then increase the threshold and repeat the loop search. This generates a list of possible loops containing the key frame, sorted first by order of SSD and then by length.

Given this list, we ask the user to pick an output length (from the minimum to the maximum loop length discovered). We find the lowest-scoring best-matching-length loop for each motion region. As many cinemagraphs play on the idea that the viewer is looking at a photograph, then reveal motion later, we also ask the user whether they would like to include a period of still frames at the beginning of the cinemagraph.

Not all input footage lends itself to loop well and, even if it is captured so, objects rarely match exactly. To correct this, we interpolate frames at the end of the looping cinemagraph to smoothly return to the starting frame. We compute SIFT flow [8] bi-directionally between the start and end frames, generate interpolated frames by scaling the vectors between them, then blend each frame (source to target blended with target to source) to generate an output frame. Often, minor background motions will appear to pop if this blending is not computed. We also

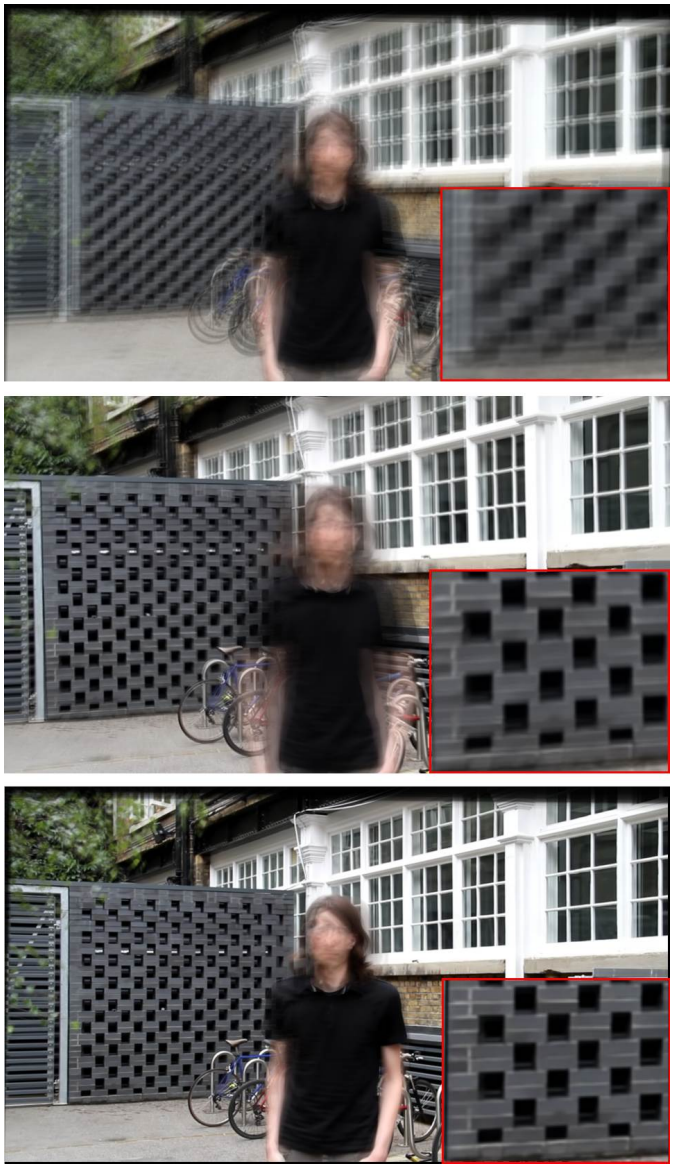

Figure 3: Video registration results. **Top:** *Deshaker (2D) mean image.* **Middle:** *Warp stabilizer (subspace) mean image.* **Bottom:** *Our method mean image, which is less blurry. Zoom-in cutouts are outlined in red (2.5x magnification).*

tested optical flow as a method for interpolation [25], but found SIFT flow gave superior results.

The final step is to composite the moving objects onto the mean image, per frame, using the spatial mask. We feather the edges of the mask to reduce aliasing and better cope with any minor lighting variations in the scene.

3.5 Results

Input, output and supplementary videos are available here: `http://www.cs.ucl.ac.uk/research/vr/Projects/AutoCinemagraphs/`.

Please refer to our supplementary video to see input footage and output cinemagraphs. In one of the eye examples, hair flicks in front of the eye and causes what looks like a blending artefact: this is not a blending artefact; rather, it is a problem of finding a good loop that considers this fine detail hair. For a 960x540 video frame, our CPU stabilization takes under 5000ms per frame (750ms for KLT tracking, and 4096ms for moving least squares warping). Mask creation timings are given in section 3.2, and are either 33ms or 106ms. Mask cleaning takes 20ms. For a 130x140x434 motion region, SSD matrix computation takes 136s. Loop finding takes 5s, frame interpolation takes 70s for the bi-directional blend, and final composition and file writing takes 33ms per frame. In total, our system takes just over 5 seconds per video frame, plus one-off costs totaling 75s plus approximately 33ms per region per frame selected (varying by size).

Admittedly, this is currently too long: a professional artist with the right footage and tools may be able to work quicker than our system. However, for novice users without the skills or professional tools we feel this is still a reasonable time to wait, especially as our approach is somewhat robust to 'bad' footage containing shake and mismatching loops (something that is more difficult to fix or takes time even with professional skills and tools). Of course, our naïve method could be faster, and many of the methods we use have corresponding GPU implementations. A production system exploiting them would see timings significantly reduced. Finally, while we think this would be an ideal application for a smart phone, even with GPU acceleration mobile users may find the waiting time too much. Use of a tripod (or a very steady hand) would reduce this time significantly as stabilization could be skipped.

4 Conclusion

We have presented a system to assist in the authoring of cinemagraphs. Our system can run automatically, only requiring the user to select which regions of motion to keep in the output. If desired, the user can edit the motion mask in two ways: by varying a threshold on the motion detection and by correcting mask regions manually. In addition, the user may select non-moving motions to be freezed at any frame. We find loops within the video sequence around user-specified key frames, and then ensure loop closure with frame interpolation. Users can add dramatic pauses to the cinemagraph to enhance the

Figure 4: Motion segmentation comparison. **Top, left to right:** *Mean image (person and leaves are moving); Chebychev; SSIM; accumulater variance.* **Bottom, left to right:** *Accumulated optical flow; blur analysis; BMD; VDP.*

Figure 5: **Top:** *A frame from a highway timelapse video.* **Middle:** *Automatically computed mask.* **Bottom:** *Interface for region selection. The background is faded and unselected regions are desaturated. As the user clicks to add motion regions to the cinemagraph, the regions saturate with colour.*

motion effect. The output of our system is a seamlessly looping cinemagraph.

References

[1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[2] Patrick Bouthemy and Edouard François. Motion segmentation and qualitative dynamic scene analysis from an image sequence. *Int. J. Comput. Vision*, 10:157–182, April 1993.

[3] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 353–360, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[4] Michael F. Cohen and Richard Szeliski. The moment camera. *Computer*, 39(8):40 –45, August 2006.

[5] Adam Finkelstein, Charles E. Jacobs, and David H. Salesin. Multiresolution video. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 281–290, New York, NY, USA, 1996. ACM.

[6] D. Gutchess, M. Trajkovics, E. Cohen-Solal, D. Lyons, and A.K. Jain. A background model initialization algorithm for video surveillance. In *IEEE International Conference on Computer Vision*, volume 1, pages 733–740, 2001.

[7] Michal Irani, Benny Rousso, and Shmuel Peleg. Detecting and tracking multiple moving objects using temporal integration. In *Proceedings of the Second European Conference on Computer Vision*, ECCV '92, pages 282–287, London, UK, 1992. Springer-Verlag.

[8] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. Sift flow: Dense correspondence across different scenes. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 28–42, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28:44:1–44:9, July 2009.

[10] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30:4:1–4:10, February 2011.

[11] Rafal Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions.

[12] Antoine Manzanera and Julien C. Richefeu. A new motion detection algorithm based on [sigma]-[delta] background estimation. *Pattern Recognition Letters*, 28(3):320–328, 2007. Advances in Visual information Processing, Special Issue of Pattern Recognition Letters on Advances in Visual Information Processing. (ICVGIP 2004).

[13] Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:1150–1163, July 2006.

[14] François G. Meyer and Patrick Bouthemy. Region-based tracking using affine motion models in long image sequences. *CVGIP: Image Underst.*, 60:119–140, September 1994.

[15] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 302–309, 2004.

[16] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 5, pages 2789 –2792 vol.5, May 1998.

[17] Fabrizio Pece and Jan Kautz. HDR for dynamic scenes. In *Conference for Visual Media Production 2010 (CVMP 2010)*, London, UK, United Kingdom, 2010.

[18] M. Piccardi. Background subtraction techniques: a review. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.

[19] Stephen Pollard, Maurizio Pilu, Sean Hayes, and Adele Lorusso. View synthesis by trinocular edge matching and transfer. In *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, WACV '98, pages 168–, Washington, DC, USA, 1998. IEEE Computer Society.

[20] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann Inc., San Francisco, CA, 2005. Chapter 4, Pages 147-152.

[21] Yeon Geol Ryu, Hyun Chul Roh, and Myung Jin Chung. Long-time video stabilization using point-feature trajectory smoothing. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 189–190, 2011.

[22] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25:533–540, July 2006.

[23] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[24] S.M. Smith and J. M. Brady. ASSET-2: Real-time motion segmentation and shape tracking. *Computer Vision, IEEE International Conference on*, 0:237, 1995.

[25] Deqing Sun, S. Roth, and M.J. Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432 –2439, june 2010.

[26] David S. Tweed. *Motion Segmentation Across Image Sequences*. PhD thesis, Department of Computer Science, University of Bristol, April 2001.

[27] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600 –612, april 2004.

[28] Greg Ward. Fast, robust image registration for compositing high-dynamic range photographs from handheld exposures. *Journal of Graphics Tools*, 8(2):17–30, 2003.

[29] L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):774–780, aug 2000.