Interactive Viewpoint Video Textures

Philippe Levieux, James Tompkin, Jan Kautz
University College London

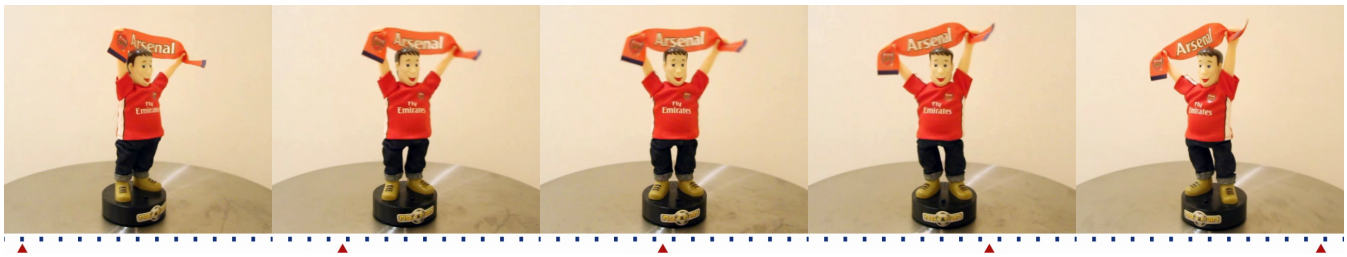


Figure 1: We enable the viewer to spatially explore a temporally coherent dynamic scene. In this example, captured discontinuously with a single camera, the viewer changes the rotation of the figurine while it coherently waves the scarf above its head. The waving motion continues even when the viewer stops rotating the figurine at novel viewpoints.

ABSTRACT

We propose an approach to interactively explore video textures from different viewpoints. Scenes can be played back continuously and in a temporally coherent fashion from any camera location along a path. Our algorithm takes as input short videos from a set of discrete camera locations, and does *not* require contemporaneous capture – data is acquired by moving a single camera. We analyze this data to find optimal transitions within each video (equivalent to video textures) and to find good transition points *between* spatially distinct videos. We propose a spatio-temporal view synthesis approach that dynamically creates intermediate frames to maintain temporal coherence. We demonstrate our approach on a variety of scenes with stochastic or repetitive motions, and we analyze the limits of our approach and failure-case artifacts.

1. INTRODUCTION

Videos have been used for many decades to provide the viewer with a sense of place and to immerse them into a scene or an event. Video is especially well suited to capture the dynamics of a scene and enable the viewer to follow the path taken by the camera. However, interaction with the medium is extremely limited. For instance, the viewer cannot stop a moving camera at a location and observe the dynamics of the scene as the only option is to stop playing the video. Over the past decade, image-based rendering (IBR) techniques [13] have enabled some interaction with photos and videos. For instance, the video textures technique [17, 1] allows the viewer to continuously explore the (stochastic or repetitive) dynamics of a scene, albeit only for a fixed spatial position.

CVMP '12, December 5 – 6, 2012, London, United Kingdom

Free viewpoint video approaches [18] allow the user to move a virtual camera in the scene, but require expensive specialized multi-camera setups, and cannot generate endless animation.

Our goal is to allow the viewer to observe the stochastic or repetitive dynamics of a scene at any location along a path sampled by video captures (see Figure 1). We want to avoid carefully calibrated multi-camera setups, as these are costly and cannot be used for forward-motion camera paths (along the line of sight of the camera) without loss of resolution. This raises a challenge: the scene can only be captured by a single camera at a number of discrete locations along a camera path, and so critically we cannot time-synchronously capture the dynamics of the object. Nonetheless, from this footage, we want to be able to generate a continuously looping video at any camera location along the path, even between discrete capture locations, while also being able to move in a spatially and, novelly, a temporally coherent fashion.

We extend the concept of video textures to narrow-baseline multi-view data to achieve this goal. This medium provides an infinitely varying stream of images that may never exactly repeat, presenting a more natural experience to the user than hard-looping video. To create smooth transitions between the spatial sampling points along the camera path, we generate an appropriate number of virtual viewpoints in between each camera location. We make extensive use of optical flow to combine frames of neighboring camera locations into new virtual viewpoints while maintaining the appearance of correct dynamics. Our flow-based method requires camera baselines to be smaller than are typical in geometry-based free-viewpoint video; however, our baselines are similar to many existing data-driven image-based rendering techniques. As such, our technique provides 'endless' photo-realistic viewing of temporally consistent dynamic objects with real-time narrow baseline view changes and, unlike other photo-realistic IBR techniques of this kind, requires no complicated equipment, setup, or calibration.

2. RELATED WORK

Early dynamic textures work focused on model-based representations for stochastic temporal textures [20, 3, 19], which could synthesize dynamic sequences of phenomena such as smoke and water. Video textures [17] was the first approach to create dy-

namic textures of arbitrary objects under repetitive and stochastic motions. Frames from a training video are reordered and repeated indefinitely such that a new generated video is never exactly the same as the input. Video textures builds a matrix of frame-to-frame L_2-distances and assures correct temporal progression and periodic dynamics by matrix filtering and dead-end analysis.

This work includes an example where video textures are combined with 3D depth estimation to produce 3D video textures, where several cameras capture an event contemporaneously. However, only one camera is used to generate the video texture, while the other cameras are used to generate a depth map. Our work improves upon that work by allowing smooth transitions between different motions in each video view. Schödl et al.'s view-interpolation method works for left-to-right camera paths, but fails for paths with forward sections as the cameras would see each other. Our decoupling of time and viewpoint solves this problem.

Schödl et al. [16] also provide an extension to generate video textures of moving objects viewed from static cameras. [9] create better transitions within video textures by finding minimum cost seams through a window around similar frames, and so enable video textures for more difficult surfaces such as water. [1] extend video textures to panoramic imagery by rotating a single camera from a single position and solving for y,t video volume slices.

Free-viewpoint video [22, 5, 12, 2] techniques solve a similar problem to ours. A scene is usually captured with multiple video cameras. The videos are analyzed and 3D geometry is reconstructed. Novel viewpoints are rendered from the videos and the (per-frame) 3D geometry. Some methods only rely on dense per-pixel image correspondences [11], which is more akin to our method. However, in contrast to all free-viewpoint video methods, which require a multi-camera setup, our method only requires a single capture and sequential capture.

Sand et al. [15] describe a method for bringing two videos, captured from similar spatial but different temporal locations, into spatiotemporal alignment. The method relies on regularized sparse correspondence formed from static image regions, and it explicitly rejects correspondence on dynamic objects. We specifically require correspondence on dynamic objects. Applying this technique to our problem will not find correspondence for motion in dynamic scenes, and will cause ghosting. Knorr et al. [8] turn monocular video into super-resolved stereo and multi-view video. However, as this approach is dependent on structure-from-motion, again it is not suitable for finding correspondence on dynamic objects.

Uyttendaele et al. [21] present an interactive system to navigate omnidirectional video of real-world environments. They perform object replacement to enhance the static world viewed when the camera position is stopped, e.g., replacing a static fireplace with a video texture. These video texture replacements are not view-dependent; in this use case, our method could provide view-dependence.

Our method is indirectly related to light fields [10, 7]. These allow the viewer to re-render a scene from new viewpoints as the complete incident light field is recorded. While the original approach was geared towards static scenes, it is possible to capture and re-render dynamic scenes from novel viewpoints. However, the light fields acquisition setup is complex. Camera paths towards the scene are difficult as the sampling rates are generally too low.

3. CONSTRUCTION AND RENDERING

First, we acquire short video segments at discrete, spatial locations that need not be uniformly spaced. These are analyzed to find good transition points within and between videos. Our representation for possible transition points is derived from video textures

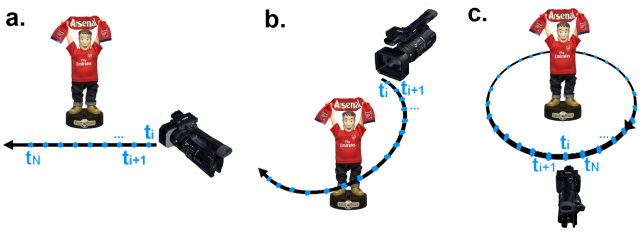


Figure 2: It is possible to capture any type of camera path, be it straight (a), curved (b,c), and so forth. It is also possible – and often easier – to keep the camera fixed and to move the object.

[17], namely transition tables, but is extended to handle spatial as well as temporal transitions. At run-time, we create continuously looping videos of the scene using these transition tables. They can be viewed from any camera location along the camera path as we synthesize new frames in real-time between real camera locations.

3.1 Capture

Our capture phase requires no complicated camera setup and could be completed by a novice or home user. This is in contrast to many existing techniques and was an important requirement when devising our method. We record short segments of video containing a repetitive or stochastically moving object at a number of discrete non-uniform spatial locations $\mathbf{x}_l$ (usually along a forward or sideways path, see Figure 2). Each segment must capture the periodicity of the object, or for stochastic motions be long enough to capture the range of motion (typically 5–20 seconds long). In our examples we capture approximately 10–30 locations at typically 5°–15° angular baselines. A single capture session takes between 10 and 30 minutes depending on the effect desired. In principle, the length of the path is not limited as the real-time renderer loads the required data on the fly, but practically the preprocessing time restricts the number of camera locations.

3.2 Preprocessing the Dataset

We generate a within-video texture transition matrix independently for each camera location. Once completed, we generate the across-video textures transition matrices that link each pair of spatially neighboring videos (Figure 3).

3.2.1 Within-video Textures

We follow the original video textures technique [17] to construct a transition table for each video segment l at $\mathbf{x}_l$: We compute an L_2-distance table D^l_{ij} between all pairs of frames i and j for each video. As in Schödl et al., we preserve dynamics by diagonally filtering the matrix, and we avoid dead ends by including future frame transition costs (with low future costs suggesting a seamless frame sequence, and high future costs suggesting a dead end with no similar frames to continue the video texture). D^l_{ij} is thresholded and local non-maxima suppressed, and converted into the transition probability table P^l_{ij}. We store one table P^l_{ij} for each camera location $\mathbf{x}_l$.

3.2.2 Across-video Textures

We extend the transition table idea to find good transition points between videos. For each pair of neighboring camera locations $\mathbf{x}_l$ and $\mathbf{x}_m$, we compare all frames i from camera l to frames j from camera m, and compute their distance with a metric to yield a cross-video distance matrix D^{lm}_{ij}. Since the viewer is allowed to travel bi-directionally, we also compute a distance matrix for the reverse direction D^{ml}_{ji}.

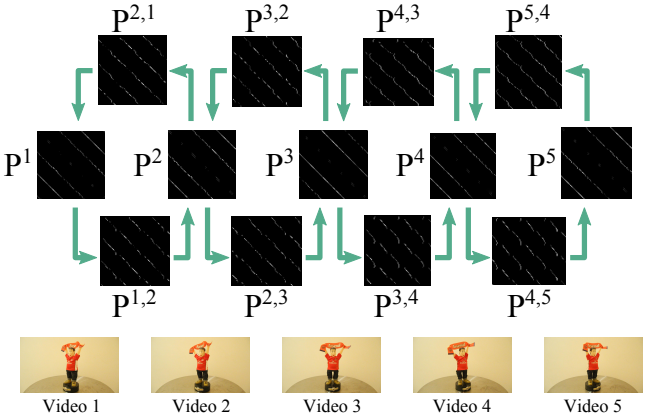


Figure 3: Within-video transition tables P^1 to P^5 are computed for all video clips 1 to 5, and across-video transition tables $P^{1,2}$ to $P^{4,5}$ and $P^{5,4}$ to $P^{2,1}$ are computed for all spatially neighbouring video clips pairs. These tables are walked as graphs to find appropriate transitions and to generate endless video at any point along the virtual camera path.

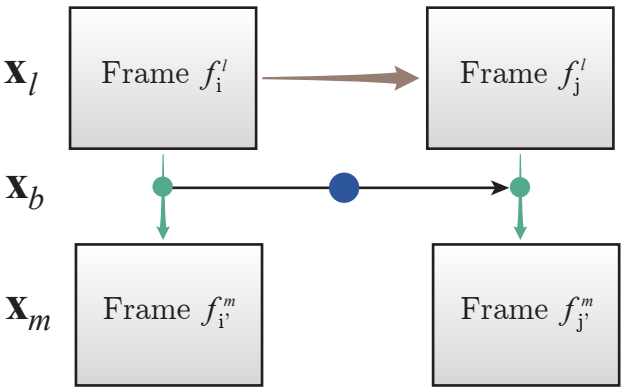


Figure 4: Based on the current frame f_i^l we choose the next frame f_j^l (brown arrow) as in standard video textures. For both frames we find the best matching frames $f_{i'}^m$ and $f_{j'}^m$ according to the cross-video transition table. The frame for location $\mathbf{x}_b$ is rendered by interpolating between frames f_i^l and $f_{i'}^m$ as well as $f_{j'}^l$ and $f_{j'}^m$ (green circles). These two intermediary frames are then interpolated to yield the final frame (blue circle) as this gives smoother results.

For the entries of D_{ij}^{lm}, we must find a meaningful distance metric between spatially neighboring frames that is invariant to slight viewpoint changes. We have investigated a number of different metrics and found that, surprisingly, for our examples the L_2-distance between images often performs well, at least if the viewpoint changes are slight and the objects are large. 'Slight' and 'large' here are both scene dependent, but are typically $5-15^\circ$ and approximately 20-30% of scene pixels for our L2 metric. These measures depend on motion parallax in the background and any camera noise.

One other metric that worked consistently for us is based on optical flow [4]. We compute the displacement field between each pair of images, and take the average of all the displacement magnitudes. We found this metric to detect similarities even in cases with large viewpoint variations. However, the L_2-distance may still be preferred as it is 3-12x cheaper computationally. One typical optical flow quality metric, computing the appearance difference (RMS intensity difference) between f_i^l and the warped version of f_j^m, is not appropriate as often the frame to frame differences are small enough that little error exists to be a reliable measure.

To preserve scene dynamics, we filter the across-video distance matrices D_{ij}^{lm} and D_{ji}^{ml} with a diagonal kernel of normalized binomial weights (see [17] Sec. 3.1). When transitioning to a neighboring video m, it is possible that the frame to which we transition leads to a dead end. As such, it is important to avoid such transitions. However, in contrast to the within-video case, it is not necessary to include future costs in the across-video distance matrices as, by the inclusion of future costs in the within-video matrices, it is very unlikely that dead ends will be picked for display (and, hence, transition) at all. Instead, we simply ensure that the chosen transition frame in the neighboring video has a high probability in P_{ij}^m. Finally, we convert the distance matrices to transition probability tables P_{ij}^{lm} and P_{ji}^{ml} as in [17]. We prune the probability tables to favor local maxima and threshold to avoid low probability jumps.

3.2.3 Estimating Camera Distances

We wish to create a smooth and seamless interactive experience from discrete camera locations, but we also do not wish to require accurate or regular camera placement. This is a challenge: interpolating between irregularly placed cameras will result in a jerky and unsmooth viewing experience. However, if we can estimate the relative distances between neighboring cameras, we can synthesize a proportional number of virtual viewpoints in between capture locations. This would ensure that the viewer can move at a constant speed along the camera path.

To accomplish this, we could opt for structure-from-motion techniques [14] to estimate camera locations, but our scenes consist of dynamic objects which tend to break these methods. As we only require approximate distances as a heuristic, we propose to use optical flow. Between each pair of adjacent video segments l and m, we choose a random subset of frame pairs (i, j), where i is from l and j is from m. We compute optical flow for each pair, and then compute the average displacement magnitude $\bar{\delta}^{l,m}$ over all flow fields. This tends to average the flow induced by the dynamic object, leaving the remaining camera motion. We assume the magnitude $\bar{\delta}^{l,m}$ is proportional to the actual distance, and so the number of viewpoints inserted between l and m is proportional to $\bar{\delta}^{l,m}$. While this relative distance heuristic estimate works well in many cases, it can fail: when there are insufficient features on static objects for optical flow to pick up or when the dynamic motion fills the camera.

3.3 Rendering

With the help of the within- and cross-video transition tables, we now need to develop a real-time method to render temporally coherent videos across location changes.

If the viewer picks a camera location $\mathbf{x}$ that coincides with one of the capture locations $\mathbf{x}_l$, we simply play the standard within-video texture associated with l. When the viewer moves to a camera location $\mathbf{x}_b$ that is *between* capture locations $\mathbf{x}_l$ and $\mathbf{x}_m$, e.g., $l < b < m$ along the path, we propose the algorithm illustrated in Figure 4.

We use the video texture from the closest capture location, say $\mathbf{x}_l$, to predict the dynamics of the scene. That is, based on the current frame f_i^l we choose the next frame f_j^l by sampling the within-video texture probability table P_{ij}^l. For these two frames, we then find matching frames from the neighboring video m: we find $f_{i'}^m$ and $f_{j'}^m$ by sampling the cross-video probability transition tables at P_i^{lm} and P_j^{lm}. However, an additional condition is added: the chosen frame $f_{j'}^m$ must have a non-zero probability in the row of $P_{i'}^m$, i.e., there needs to be some probability that frame $f_{j'}^m$ would follow on

from $f^m_{i'}$. This ensures that dynamics will be preserved even when $b = m$ and that the switch to video texture at m is not perceptible.

Given the four frames (see Figure 4), we now interpolate them using fast optical flow [6]. The obvious choice would be to simply interpolate between f^l_i and $f^m_{j'}$ according to b. However, we achieve smoother results by also including the current frames f^l_i and $f^m_{i'}$. We interpolate between spatially neighboring frames f^l_i and $f^m_{i'}$ to yield $f^b_{i''}$ (left green circle in Figure 4), and between f^l_j and $f^m_{j'}$ to yield $f^b_{j''}$ (right green circle in Figure 4). Finally, we interpolate halfway between $f^b_{i''}$ and $f^b_{j''}$ to yield the final interpolated frame $f^b_{i''j''}$ (blue circle in Figure 4). Frame interpolations are performed as in [6]. These are bi-directional blended morphs – each input frame is rendered as a quad mesh whose vertices are advected by the flow, which transforms the frame to its corresponding pair. Once advected, both input frames are linearly blended.

4. RESULTS

Figure 5 presents a variety of results which are best viewed in our supplemental video. For computational efficiency, the L2 metric is used in all of our presented sequences.

Figurine. This example features a moving figurine on a turntable while the camera is kept still. It consists of 17 video segments spanning a baseline of approximately 125° (7° between cameras) – this is in line with other image-based view interpolation methods. Geometry-based techniques, which do allow larger baselines between cameras, would not work well on dynamic scenes such as ours. Long camera paths or large cumulative baselines are also not a principled problem with our technique.

Interactive Example. The user loads a dataset into memory and moves forward and backward along the path in real-time. We use a small buffer to improve the response-time and interaction. This scene features a plant blowing in the wind in which the camera is moving across 11 viewpoints.

Drinking Bird. A scene acquired from 17 camera locations. Here, the user moves away from a drinking bird while the dynamics of the bird are preserved. The yellow and blue sliders illustrate the frames picked from the original sequences.

Candle. This scene shows the stochastic motions of a candle blowing in the wind. The camera slides horizontally to the left along 25 viewpoints. Here, we demonstrate how it is possible to travel along a path with a temporally coherent dynamic element.

Plants (Different Speeds & Frames). First, we show how it is possible to change the number of frames played per virtual viewpoint, with the number of frames inversely proportional to the speed of travel. Second, we vary the (proportional) number of virtual viewpoints between two original camera locations. On the left, with the proportion set to 0, no virtual viewpoints are generated. In the center, with the proportion set to 0.1, roughly 10 virtual viewpoints are generated. Finally, on the right, approximately 20 virtual viewpoints yield a very smooth motion along the path.

Two Plants. We present a scene with two plants where each is independently rendered while keeping the same proportion of virtual viewpoints and frames played per viewpoint. This scene split is interactively defined.

Gas Stove. Here, we additionally modify the state of the scene as well as moving the camera. We capture a gas stove burner with a low flame and increase its intensity within each video segment captured. This results in an interactive viewpoint video texture in which the user can precisely vary the intensity of the flames (by selecting the range of frames used for the within-video transition tables) as well as change the viewpoint. Unlike existing work, this shows how scene content, within limits, can be photo-realistically altered, demonstrating how to interact with scene content using image-based rendering. We also use this example to demonstrate the smoothness gained from the camera distance measure (3:00 in video). As the cameras do not regularly sample a path, the experience without accounting for camera distance is jerky (camera distances: min: 0.47, max: 2.89, mean: 1.20, st.dev.: 0.61).

4.1 Timings and Data Costs

The pre-processing time depends on the number of frames per segment, the number of segments, and the frame size. We usually downsample the flow computation to speed up the preprocess. For our examples, the preprocess using the L_2-distance took between 2–9 hours. With the optical flow metric, it increased to 5-24 hours.

While it might seem that there are more efficient approaches for matching similar frames within and across videos such that some frame comparisons could be skipped, there is a trade-off involved. All-pair matching is an intrinsic part of the video textures approach. For high-quality results, any frame-matching algorithm must guarantee that the video is loopable. At the same time, to provide variation in pseudo-random scenes, we want to find all good loopable subsequences. Efficient frame matching becomes even more of a problem for stochastic motions (such as plants or flames) as the motion can change substantially in very few frames.

Rendering is inexpensive; however, we must still compute flow vectors for each interpolation. For videos of 640×360 pixels, each new synthesized frame takes approximately 17ms on an NVIDIA GTX 580M; for videos of 1280×720 pixels, this takes approximately 30ms on the same hardware.

Our work has comparable data costs to existing free-viewpoint video approaches. Many interactive video techniques have expensive pre-processes and we are no different. As expected, for periodic motions, we actually require the minimum of data for each view (just enough to capture the periodicity of the motion). Along with the input videos (or their minimal subset of frames), we require one square within-video matrix per video with sides equal in length to the number of frames in the video, and one square/rectangular across-video matrix per neighbouring pair with width and height equal in length to the number of frames in the first and second videos in the pair respectively.

4.2 Discussion

Our technique produces photo-realistic results on the range of objects and scene setups that we have demonstrated, and this has immediate applications in advertising. However, since our rendering method is based on optical flow, artifacts tend to appear where optical flow has difficulties, e.g., for transparent or translucent objects such as glass or water, or for large parallax. Ghosting or warping can occur in novel view synthesized frames if the dynamic motion is temporally undersampled. This is especially true of stochastic motions which do not regularly repeat (see Figure 6). In general, if the video segments do not fully capture the motion then it is difficult to create both a convincing video texture and a convincing spatiotemporal transition as the number of possible seamless dynamic motion transition points between neighbouring videos is low. Likewise, if the number of capture locations is too few (and

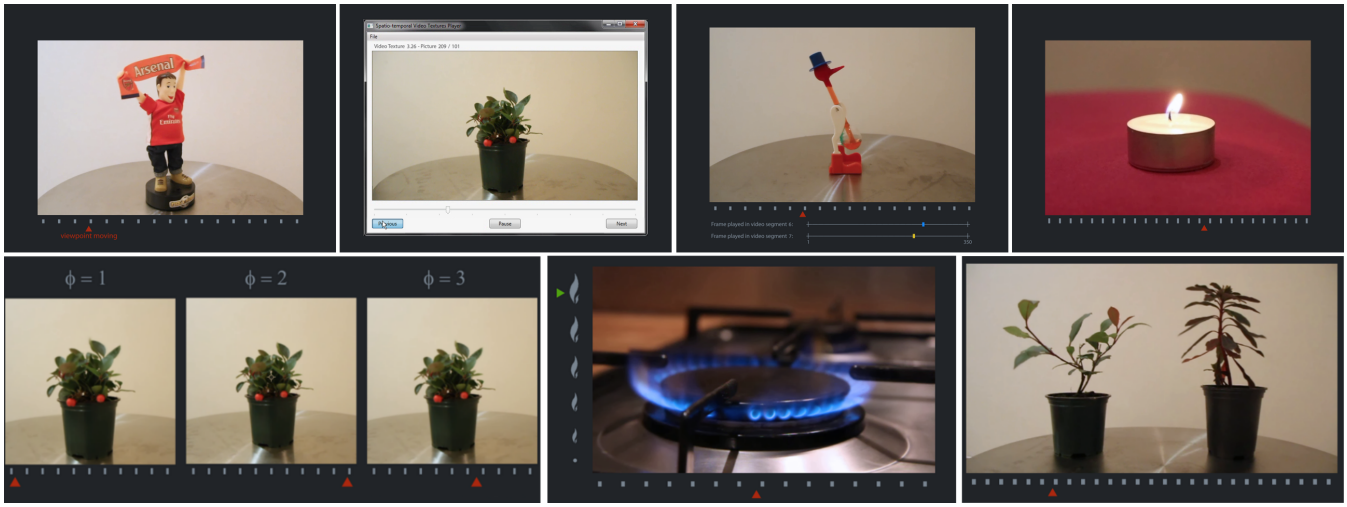


Figure 5: An overview of our results. Please view the supplemental video for a better impression.

Figure 6: Common artifacts in novel views: *Left:* The transparent glass edges have incorrect flow and so ghost. The liquid in the neck of the drinking bird also ghosts because the across-video L2 distance metric cannot accurately account for all scene motion. Switching to the optical flow metric alleviates this problem. *Center:* Leaves in this tree ghost and warp because the stochastic motion is temporally undersampled and so there are no good across-video transitions. *Right:* The drinking bird is temporally understampled and so ghosts.

the distances between them too far, see Table 1), then rendering quality suffers as correct correspondences are often not found during optical flow. More free-form capture setups, such as hand-held capture, may be possible with a rigid software stabilization preprocess for each input clip, but complicated capture environments with many dynamic objects or variable lighting conditions (as is common outdoors) would cause problems for our technique. More sophisticated correspondence estimation methods, such as those in [11], allow more substantial camera motions, but these approaches take many hours per frame pair to compute correspondence.

Table 1 focuses on the figurine example to assess and relate both our camera distance measure and the artifacts which are introduced when increasing the angular baseline between capture locations (4:20 in video). For this assessment we use maximally 25 views, each approximately 5° apart and covering 125° total. Gradually, we remove cameras such that the baseline between cameras increases but the overall baseline between the path start and end cameras is the same. Visible artifacts at each of these different spatial samplings are identified, and we show how camera distance relates to baseline. As the camera distance is an average, the values (in pixels) should not be taken literally, e.g., a 25° baseline undergoes more than a 3 pixel movement. In our experience, the limiting factor for disparity is not the ability of the metric to find good transitions, but is rather the optical flow interpolation for generating intermediate frames. As we use a real-time flow technique, ghosting artifacts may start to appear as the disparity increases, but this is scene dependent. Better morph results could be achieved with more expensive dense correspondence matching but then our approach would no longer be interactive.

How much data can you throw away? For this example, a baseline of 14° (9 views)/camera distance of 1.59 still gives pleasing results. This would allow a 360° rotation with 24 spatial locations. If the object had a short cycle (< 30s), a complete capture session would take approximately 30 minutes with no setup or calibration.

For forward-motion camera paths along the line of sight of the camera, or dolly-like shots, baseline/camera distance has a different artifact relationship than for curved camera paths. A dolly-like shot generally moves towards (or away) from one point in the scene – the focus of expansion (or contraction; henceforth FOE) – at which there is zero optical flow. Even for very large camera distances, scene content at and around this point will be artifact free. Conversely, scene content far from the FOE will suffer artifacts even at small camera distances. Table 2 demonstrates this with the drinking bird example. In this example, with the object spanning much of the image height, a baseline of 5cm/camera distance of 2.32 gives pleasing results. Image regions close to the FOE suffer no artifacts even at the most distant baseline tested. While image regions that are middle distance from the FOE do not suffer ghosting until large camera distances of 5+, far distances suffer major ghosting and warping much earlier.

5. CONCLUSION

We have presented a user-friendly method to capture and render interactive viewpoint video textures. Our work produces real-time high-quality results free from disagreeable artifacts, using a simple and easily-reproduced automatic approach. This approach reduced the equipment requirements for a specific subset of free-viewpoint video applications. Every technique has trade-offs: in our case, in providing camera capture flexibility and reducing equipment costs, we restrict the scope of possible scenes.

Our approach is straightforward and this is positive given the photo-realistic quality we achieve. We relate two previously uncon-

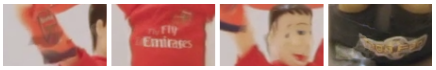
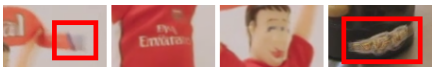
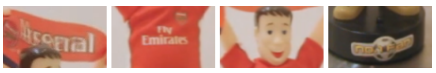
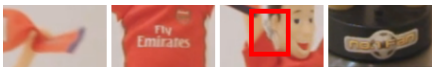
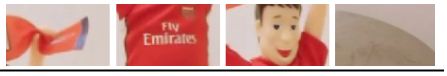
Views	Baseline	Cam. Dist.	Artifacts	Zoom
5	25°	2.69	Major warping and discontinuities across many figurine parts; reduced sense of temporal consistency.	
6	20°	2.31	Temporal consistency, but major warping/ghosting on many parts.	
7	18°	2.00	Facial disfigurement and warping on base logo.	
9	14°	1.59	Face intact; some scarf ghosting and base logo warping on some transitions.	
13	10°	1.16	Minor flow artifact on ear.	
25	5°	0.69	Minor ghosting on turntable surface; otherwise no noticeable artifacts.	

Table 1: How much data can you throw away? Figurine curved path dataset with gradually increasing number of camera locations for a fixed 125° arc, their camera distance, and the artifacts caused. All numbers of views suffer some turntable ghosting (see video). Camera distance is in pixels.

nected research fields (of video textures and free-viewpoint video) and bring important accessibility and flexibility gains. Needing only one camera, anyone at home could create these effects easily with only a tripod: camera placement need not be uniform as our camera distance estimation automatically compensates to produce smooth, photo-realistic results with no calibration. Our rendering algorithm is computationally inexpensive, and enables the user to interact with video textures by moving the viewpoint and, in our gas stove example, by changing scene content. As such, our work is immediately applicable to advertising. We demonstrate the effectiveness of our approach on a variety of examples and show at which points it is likely to break with a baseline/camera distance analysis.

5.1 Future Work

We have demonstrated linear and curved paths. Non-linear paths and grid arrangements are possible, though our viewing interface would have to be adapted for grid arrangements. There is more to explore with along-line-of-sight paths also, as our examples could arguably be achieved with zooming – dolly-zooms and forward-sweeping paths should be explored. Currently our work is limited to repetitive or stochastic motions. Ideally, we would extend our method to deal with structured motions while keeping the capture just as simple, though this is a challenging extension. Less homogenous backgrounds are possible so long as the camera path does not induce a larger flow vector change through parallax motion in the background than the flow vector change in cyclic movement in the foregorund object, and these situations need to be tested. Backgrounds with moving content would also be a challenging addition. Finally, we would like to explore a greater range of interactive scene content changes as exemplified by our gas stove example. This use of our approach provides fertile ground for future research into interactive techniques.

6. REFERENCES

[1] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. *ACM Trans. Graph.*, 24(3):821–827, 2005.

[2] L. Ballan, G. J. Brostow, J. Puwein, and M. Pollefeys. Unstructured video-based rendering: interactive exploration of casually captured videos. *ACM Trans. Graph.*, 29(4):87:1–87:11, 2010.

[3] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. volume 7, pages 120–135. IEEE Computer Society, 2001.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV*, volume 3024, pages 25–36, 2004.

[5] P. Einarsson, C.-F. Chabert, A. Jones, W.-C. Ma, B. Lamond, T. Hawkins, M. Bolas, S. Sylwan, and P. Debevec. Relighting Human Locomotion with Flowed Reflectance Fields. pages 183–194, Nicosia, Cyprus, 2006. Eurographics Association.

[6] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Computer Graphics Forum*, 27(2):409–418, 2008.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 43–54, 1996.

[8] S. Knorr, M. Kunter, and T. Sikora. Stereoscopic 3d from 2d video with super-resolution capability. *Image Commun.*, 23(9):665–676, Oct. 2008.

[9] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 277–286, 2003.

[10] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 31–42, 1996.

[11] C. Lipski, C. Linz, K. Berger, A. Sellent, and M. Magnor. Virtual video camera: Image-based viewpoint navigation

Baseline	Cam. Dist.	Artifacts	Zoom (close, middle, far from FOE)
2cm	1.31	No noticeable artifacts.	
3cm	1.68	Minor warping and ghosting at image edges far from FOE.	
4cm	2.01	Minor warping and ghosting at image edges far from FOE.	
5cm	2.32	Warping and ghosting at image edges far from FOE.	
6cm	2.87	Warping and ghosting at image edges far from FOE.	
7cm	3.29	Warping and ghosting at image edges far from FOE, minor ghosting at middle distances from FOE.	
13cm	5.43	Major warping and ghosting at image edges far from FOE, major ghosting at middle distances from FOE.	

Table 2: How much data can you throw away? Drinking bird dolly shot dataset with gradually reducing spatial camera locations, their camera distance, and the artifacts caused. Camera distance is in pixels. FOE stands for focus of expansion.

through space and time. *Computer Graphics Forum*, 29(8):2555–2568, Dec. 2010.

[12] M. Magnor. *Video-based Rendering*. A K Peters, 2005.

[13] L. McMillan and S. Gortler. Image-based rendering: A new interface between computer vision and computer graphics. *SIGGRAPH Comput. Graph.*, 33:61–64, 1999.

[14] D. P. Robertson and R. Cipolla. Structure from motion. In M. Varga, editor, *Practical Image Processing And Computer Vision*. John Wiley, 2008.

[15] P. Sand and S. Teller. Video matching. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 592–599, New York, NY, USA, 2004. ACM.

[16] A. Schödl and I. A. Essa. Controlled animation of video sprites. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 121–127, 2002.

[17] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 489–498, 2000.

[18] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand. 3d video and free viewpoint video - technologies, applications and mpeg standards. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 2161 –2164, july 2006.

[19] S. Soatto, G. Doretto, and Y. Wu. Dynamic textures. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 439–446, 2001.

[20] M. Szummer and R. W. Picard. Temporal texture modeling. In *IEEE Intl. Conf. Image Processing*, volume 3, pages 823–826, 1996.

[21] M. Uyttendaele, A. Criminisi, S. Kang, S. Winder, R. Szeliski, and R. Hartley. Image-based interactive exploration of real-world environments. *Computer Graphics and Applications, IEEE*, 24(3):52 – 63, may-jun 2004.

[22] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.