UCLA-CSD-TR#020043

Towards Plenoptic Dynamic Textures

Gianfranco Doretto Stefano Soatto

Department of Computer Science, UCLA, Los Angeles - CA 90095

November 6, 2002

Keywords: stereo and structure from motion, image-based modeling, multiview stereo, 3-D reconstruction

Abstract

We present a technique to infer a model of the spatio-temporal statistics of a collection of images of dynamic scenes seen from a moving camera. We propose three inference algorithms that can be used to stabilize and register images of dynamic scenes, as well as to generate synthetic sequences where the motion of the vantage point can be controlled interactively.

1 Introduction

Images of objects with complex shape, motion and material properties are commonplace in our visual world: think of a silk gown, a burning flame, a waterfall. The complexity of these physical phenomena is far superior to that of the images that they generate and, therefore, the inverse problem of visual perception is intrinsically ill-posed. For instance, the rivulets on the surface of a lake could be the result of homogeneous material (water) being tossed around by the wind and the currents; however, the same perception could be elicited by a giant flat screen where a time-varying signal is projected to match the appearance of the rivulets, adapted to the viewer's moving vantage point. Therefore, unless additional prior assumptions are available[1], one has to give up the goal of inferring *"the"* model of the physical world, and settle for a much poorer representation, one that can explain the measured data. What representation to choose depends on the task at hand.

In this paper, we are interested in inferring models of the spatio-temporal statistical properties of visual scenes that can be used to generate synthetic sequences of images, where both the temporal statistics and the motion of the vantage point can be controlled.

[1] As usually done in "shape from X" algorithms.

1.1 Related work and contributions of this paper

Our goal is simple to state: we are given images of a dynamic scene, taken from a moving camera, and we want to extract a model so that we can re-play the sequence from an arbitrary vantage point. As simple as the goal sounds, the discussion above suggests that it is unattainable. Since we know at the outset that we cannot retrieve a physically correct model of the shape, dynamics and material properties of the scene, we will let our task guide our assumptions on the representation to lead to a well-posed inference problem.

We will model the images (or filtered versions of the images) as the output of a dynamical model. The model, together with a stochastic input, represents the dynamic variability of the image sequence. However, unlike prior work, we explicitly model the vantage point, so that during the synthesis, we can change it arbitrarily and render sequences from a camera undergoing a virtual motion.

Several algorithms have been proposed for interpolating and extrapolating views of a scene without explicit reconstruction (see for instance [2, 7, 18] and references therein), as well as techniques to model changes in the viewpoint that use little or no scene structure information (e.g. [13, 11, 19]). All of these techniques, however, require the scene to be static (or a large number of static calibrated cameras), whereas we are interested specifically in modeling dynamic scenes.

The literature on modeling of dynamic scenes is also sizeable. For instance, [14] consider the problem of physically-correct novel view interpolation between two views of a scene where objects are in different positions. Another example is mosaicing for dynamic scenes (e.g [12]), just to cite an example. Most of these techniques, however, consider scenes made of a number of rigid objects, whereas we are interesting in scenes where the temporal dynamics of the entire scene can be modeled, and no rigidity constraints are available.

Image-based rendering techniques are available for specific classes of non-rigid objects, e.g. for facial expression animations [6, 9, 24, 4], and thoroughly exploit the structure of the object of interest. Here, we are interested in scenes for which no prior information is available, and allow complex shape, motion, and material properties. We will, however, make assumptions on the "temporal regularity" of the scene, in ways that we will explain in later sections.

This latter class of scenes has recently received some attention, and there are techniques to model changes in the dynamics of such scenes (e.g. temporal textures [15, 21, 23], time-varying textures [3], video textures [17, 10, 5], dynamic textures [20], textured motions [22]). Therefore, one can find techniques to model either changes in the viewpoint, as mentioned above, or changes in the dynamics of the scene, but not both. To the best of our knowledge, this paper is the first to attempt to model the temporal dynamics of the scene in ways that allow direct control and rendering from an arbitrary (moving) vantage point. Therefore, our approach can be though of as an extension of light field rendering to dynamic scenes, or as an extension of video/dynamic textures to moving cameras.

In the next section we introduce the formalism to be used throughout the paper, and in Section 3 we propose three approaches to model moving images of dynamic scenes for the purpose of view synthesis and rendering. The performance of two of these approaches are explored in the experimental Section 4.

2 Formalization of the problem

Let $\mathbf{X} \in \mathbb{R}^3$, and $I(\mathbf{X}, t) \in \mathbb{R}_+$ the "energy" of a particle in position $\mathbf{X}$ in space at time t. An image $y(\mathbf{x}, t)$ at pixel $\mathbf{x} \in \Omega \subset \mathbb{R}^2$ at time t is in general obtained by integrating the energy along the projection ray $\mathbf{x}\lambda \in \mathbb{R}^3$, where $\lambda \in \mathbb{R}_+$ and $\mathbf{x}$ is expressed in homogeneous (projective) coordinates: $y(\mathbf{x}, t) =$

$\int_{\mathbb{R}_+} I(\mathbf{x}\lambda, t)d\lambda$. If we assume that particles are opaque, or that they are concentrated on a surface, then the integral reduces to the value of I at the particular $\mathbf{X}$ that corresponds to $\mathbf{x} = \pi(\mathbf{X})$, where π is the canonical projection: if $\mathbf{X} = \mathbf{x}\lambda$, then $\pi(\mathbf{X}) = \mathbf{x}$. More in general, let $P(\mathbf{x}) \in \mathbb{R}^3$ be the surface in space that contributes to the image irradiance y at $\mathbf{x}$, and let the viewpoint move according to a motion described by a group $g(t)$ (Euclidean, affine or projective). Therefore, under these assumptions, we have $y(\mathbf{x}, t) = I(g(t)P(\mathbf{x}), t) + w(\mathbf{x}, t)$ where w is a noise term that we assume to be white and zero-mean. Now, if we further allow the surface P to change over time, we have the image formation model in the most general form that we will address in this paper:

$$y(\mathbf{x}, t) = I\left(g(t)P(\mathbf{x}, t), t\right) + w(\mathbf{x}, t). \tag{1}$$

The goal is, given measurements of $y(\mathbf{x}, t)$ for $\mathbf{x} \in \Omega \subset \mathbb{R}^2$ and $t = 1, \ldots, \tau$, to recover a model of the form above, consisting of the unknowns $I(\cdot, \cdot), g(\cdot)$ and $P(\cdot, \cdot)$, such that novel sequences can be generated by altering the model or controlling its states.

2.1 Reduction of the model

Unfortunately, the model (1) is "too rich," in that given any measured sequence $\{y(\mathbf{x}, t),\ \mathbf{x} \in \Omega, 0 \leq t \leq \tau\}$ there exist infinitely many models I, g, P that generate them. Therefore, learning would be subject to overfitting and the resulting model would have little predictive power. In fact, if we write the model (1) in more compact form as $I_t \circ g \circ P_t$, then it is immediate to see that, for any choice $\tilde{g} \in SE(3)$ and arbitrary $\tilde{P}(\mathbf{x}, t)$, we can always choose $\tilde{I}(\mathbf{X}, t)$ that satisfies the equation $I_t \circ g \circ P_t = \tilde{I}_t \circ \tilde{g} \circ \tilde{P}_t$. Therefore, we need to restrict the class of allowable energy fields I. In the following, we will assume that it is subject to a temporal dynamics that is second-order stationary. That is, $I(\mathbf{X}, t)$ is *allowable* if there exist $C(\mathbf{X})$, A and $\xi(t)$ such that

$$\begin{aligned} \xi(t+1) &= A\xi(t) + v_\xi(t) \quad \xi_0 \sim \mathcal{N}(0, \Sigma_\xi) & (2)\\ y(\mathbf{x}, t) &= C(\mathbf{X})\xi(t) + w(t) & (3) \end{aligned}$$

for some v_ξ, w white, zero-mean Gaussian random processes, and $I(\mathbf{X}, t) = C(\mathbf{X})\xi(t)$. In other words, $I(\cdot, t)$ is a 3-D linear dynamic texture [20]. Notice that $I(\mathbf{X}, t)$ cannot be measured for all $\mathbf{X} \in \mathbb{R}^3$, but it is instead sampled on a set of measure zero determined by $\mathbf{X}(t) = g(t)P(\mathbf{x}, t)$.

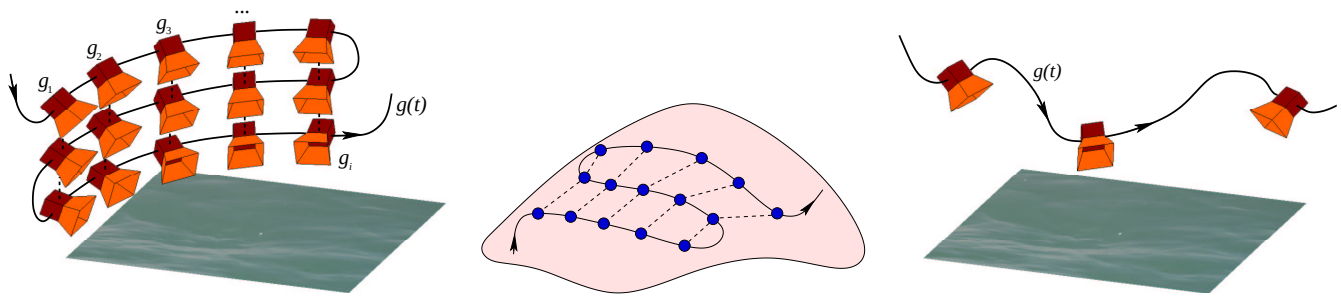
As restrictive as this model may appear, it is not enough to guarantee a one-to-one correspondence between parameters and output realizations. In fact, given C, g, P and ξ, one can always find $\tilde{C}, \tilde{g}, \tilde{P}, \tilde{\xi}$ that satisfy

$$\tilde{C}(\tilde{g}(t)\tilde{P}(\mathbf{x}, t))\tilde{\xi}(t) = C(g(t)P(\mathbf{x}, t))\xi(t).$$

Indeed, one can choose a function $\tilde{C}(\cdot)$ and $\tilde{g}(t)$ arbitrarily, and always find $P(\mathbf{x}, t)$ that satisfy the equation above. Therefore, we need to restrict P. One possibility is to assume that P is a static surface, and only the viewpoint $g(t)$ and the radiance $I(\cdot, t)$ are allowed to change over time. In that case, we have

$$I(\mathbf{X}, t) = C(g(t)P(\mathbf{x}))\xi(t) \tag{4}$$

and $y(\mathbf{x}, t) = I(\mathbf{X}, t) + w(\mathbf{x}, t)$. To render matters even simpler, we assume that the scene is not just static, but can be approximated locally by a plane, so that $g(t)P(\mathbf{x})$ can be represented as a homography



(a) (b) (c)

Figure 1: *Interpolation of the camera trajectory in $SE(3)$ (or $\mathbb{R}^3$), (a), requires knowledge of the extrinsic calibration, which can be obtained with a laborious procedure. Interpolation in the space of the model parameters (b), can be easily attained by building the local connectivity graph by computing mutual distances among each model. Under the homographic approximation, one can estimate model parameters along a one-dimensional subset of camera trajectories (c).*

of the image plane[2] $H(t) \in \mathbb{R}^{3\times 3}/\mathbb{R}$, and therefore we have $I(\mathbf{X}, t) = C(H(t)\mathbf{x})\xi(t)$, where $\mathbf{x}$ is expressed in homogeneous coordinates. Finally, we need a dynamical model for the evolution of $H(t)$: in lack of a better model[3], we will assume it to be a first-order random walk. Summarizing this simplified model, we have

$$\begin{aligned} \xi(t+1) &= A\xi(t) + v_\xi(t) \quad \xi(0) = \xi_0 \\ H(t+1) &= H(t) + v_H(t) \quad H(0) = I \\ y(\mathbf{x}, t) &= C(H(t)\mathbf{x})\xi(t) + w(\mathbf{x}, t). \end{aligned} \tag{5}$$

We call the noiseless output $C(H(t)\mathbf{x})\xi(t)$ a *linear plenoptic dynamic texture*. Now, the problem of inferring a model consists of, given $\{y(\mathbf{x}, t),\ \mathbf{x} \in \Omega, t = 1, \ldots, \tau\}$, estimating $A, C(\cdot), H(t), \xi(t)$ as well as the covariance of the noise terms v_ξ, v_H and w. This problem is difficult because C is sampled only on a set of measure zero.

3 Three approaches

In the next three subsection we propose three approaches to learn and synthesize plenoptic dynamic textures. The first is conceptually straightforward but difficult to implement in practice, because it requires a large number of calibrated and synchronized cameras. The second corresponds to doing model

[2]If the plane has normal vector $\nu \in \mathbb{R}^3$ relative to the camera frame, and moves with a Euclidean motion $g(t) \in SE(3)$, represented by a rotation matrix $R(t) \in SO(3)$ and a translation vector $T(t) \in \mathbb{R}^3$, then $H(t) = R(t) + T(t)\nu^T$ up to scale.

[3]Of course, if a better model is available, for instance from knowledge of the dynamics of the camera platform, it can be used. The random walk model for H is defined in the embedding space $\mathbb{R}^{3\times 3}$, as opposed to the quotient $\mathbb{R}^{3\times 3}/\mathbb{R}$. Definition on the quotient, although more principled, carries little practical consequences, at the expenses of a much more complicated notation, and is therefore not reported here.

interpolation in the space of models, and does not require calibration. The last approach has yielded the most rewarding results in the experiments, and consists of an alternating minimization directly on the model discussed in the previous section.

3.1 The Dynamic Lumigraph

The first and conceptually simplest approach to modeling and learning space-time dynamic textures is to start directly from the model

$$\begin{aligned} \xi(t+1) &= A\xi(t) + v_\xi(t) \quad \xi_0 \sim \mathcal{N}(0, \Sigma_\xi) && (6) \\ I(\mathbf{X}, g, t) &= C(\mathbf{X}, g)\xi(t) + w(t) && (7) \end{aligned}$$

and collect data for a large collection of "voxels" $\mathbf{X}$ and viewpoints g. At that point, synthesis is trivially performed by choosing $g(t)$ and P via

$$y_{synth}(\mathbf{x}, t; g(t), P(\mathbf{x}, t)) = C(g(t)P(\mathbf{x}, t))\xi(t). \quad (8)$$

This approach corresponds to a dynamic version of the so-called Lumigraph [11], and aims at modeling the plenoptic function [1] directly. Although conceptually viable, this approach is impractical because it would require a large number of synchronized cameras, one per desired location g. In the Lumigraph, the camera is moved, so that time is used to sample space, a trick that we cannot apply here since we need to sample both in space and time as our scene is not static. Since we do not have an experimental facility that would allow us to collect synchronized images, this avenue is not pursued in the experimental Section 4.

3.2 Model interpolation

Consider a camera trajectory, represented by $g(t) \in SE(3)$. This is a one-dimensional subset of the six-dimensional space of all possible camera poses. If we consider spherical projection, then we can consider simply the trajectory of the optical center, which moves in the three-dimensional space $\mathbb{R}^3$. In either case, our data spans a one-dimensional space only, and we are interested in interpolating and extrapolating the viewpoint so as to be able to move the camera arbitrarily in space.

If enough viewpoints $g(t)$ are sampled uniformly in $SE(3)$ (or $\mathbb{R}^3$ in case of spherical projection), then one could think of generating synthetic sequences by interpolating models of the form (7) from neighboring g's. Naturally, this *requires knowing the pose of the camera* that captured the original data, i.e. that the (extrinsic) calibration of the camera be known. Most often, however, the camera pose is not known precisely, which is another shortcoming of the straightforward extension of the Lumigraph to dynamic objects.

Consider now the set of models corresponding to a trajectory of viewpoint samples $g_1, g_2, \ldots, g_i$ (see Figure 1-(a)). These are points in the space of models (see Figure 1-(b)), which can be endowed with a metric structure, which allows determining the complete distance graph between each sample model. Therefore, one can envision generating synthetic sequences by generating local interpolations in the space of models. The shortcoming of this method is that one cannot manipulate the viewpoint directly, and therefore the editing power is reduced. In Section 4 we show a few experiments using this technique, which is illustrated in Figure 1.

Interpolation can be performed linearly once the local connectivity graph is available. In order to compute it, one needs to define a norm in the space of models. This can be done in a variety of ways.

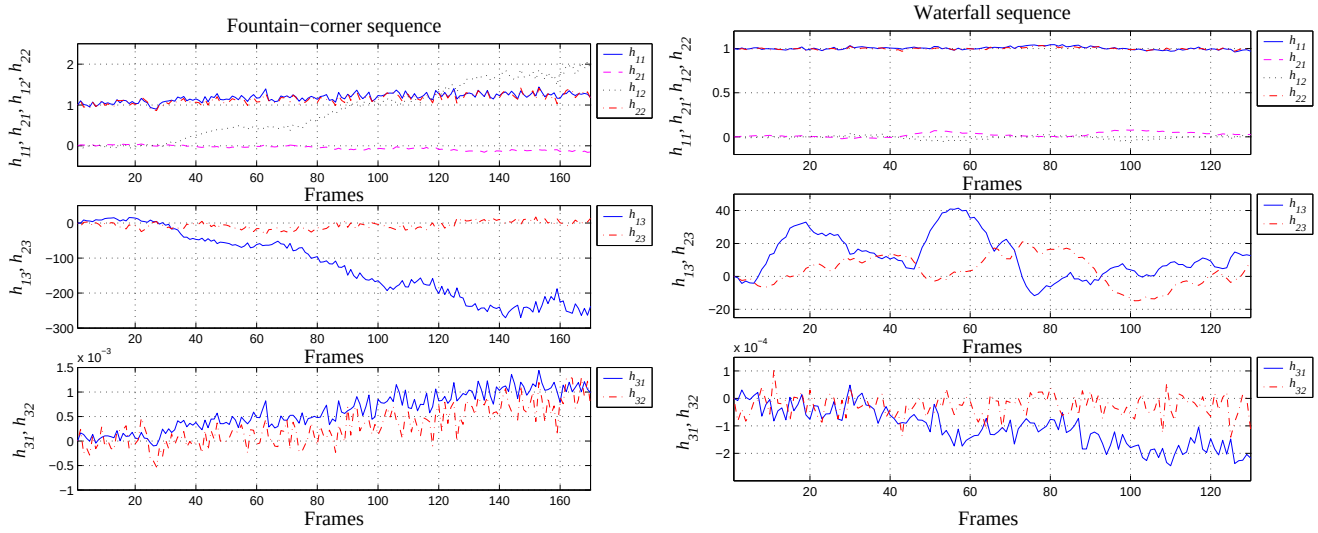


Figure 2: *Homography estimated values $\hat{H}(t)$, for the fountain-corner sequence (left), and the waterfall sequence (right). Using matlab notation, the homography matrix is defined as $H = [h_{11}h_{12}h_{13}; h_{21}h_{22}h_{23}; h_{31}h_{32}\ 1]$.*

We choose subspace angles among extended observability subspaces, as recently proposed by DeCock and DeMoor; the interested reader can refer to [8] for details.

3.3 Homographic approximation

In this section we explore interpolation based directly on the model (5), where the matrix C is inferred only along the one-dimensional subset of camera trajectories, and therefore synthesis will be limited to a (homographic) neighborhood of this trajectory (naturally, if the scene is planar, this neighborhood spans the entire space, and therefore one can generate views from an arbitrary vantage point). This setup is illustrated in Figure 1-(c).

The pedestrian way to inferring the state of the model (5) consists of first estimating the homography off-line, for instance by registering a set of point features that are known to be static, and then applying any of the modeling algorithms as if the viewpoint was fixed, for instance [21, 20, 22]. This corresponds to defining $\tilde{y}(\mathbf{x}, t) \doteq y(H^{-1}\mathbf{x}, t) = y(\tilde{\mathbf{x}}(t), t) = C(\tilde{\mathbf{x}}(t), t)\xi(t) + w(\tilde{\mathbf{x}}(t), t)$ where, by assumption, $C(\tilde{\mathbf{x}}(t), t) = C(\tilde{\mathbf{x}}(0), 0)$ is constant.

A more convenient and practical algorithm, which we explore in Section 4, consists of setting up an alternating minimization problem where we start with

$$\hat{H}_0 = I;\ \hat{\xi}_0 = 0 \tag{9}$$

and at a generic iteration i alternate a step of the subspace identification algorithm N4SID (see [16] for details)

$$\hat{C}_{i+1}, \hat{A}_{i+1} = \text{N4SID}(\{y(\hat{H}_i^{-1}\mathbf{x}, t)\}_{t=1,\dots,\tau}) \tag{10}$$

with a step along the gradient of the norm $\phi(i) \doteq \sum_t \|y(\mathbf{x}, t) - \hat{C}_{i+1}\hat{\xi}(t)\|$ subject to $\hat{\xi}(t+1) = \hat{A}_{i+1}\hat{\xi}(t)$, which is given by

$$\hat{H}_{i+1}(t)\mathbf{x} = \hat{H}_i(t)\mathbf{x} + \alpha_i\phi(i)\nabla ID(\hat{H}_i(t), \mathbf{x}) \tag{11}$$

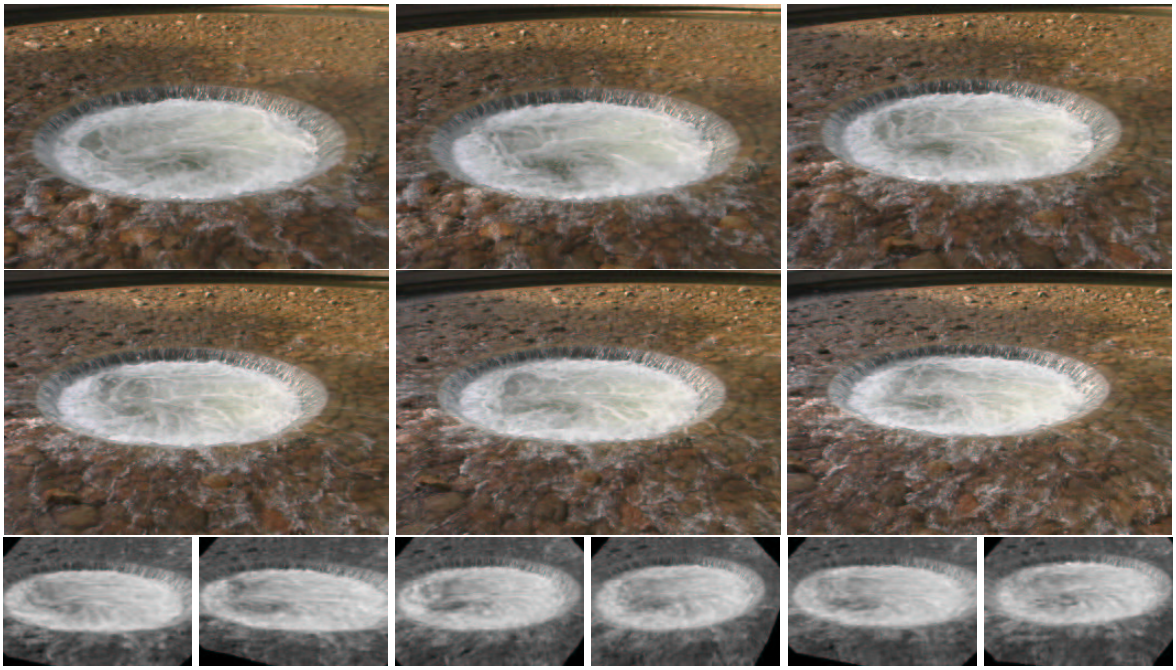


Figure 3: **Inverted-fountain.** *Top two rows: samples of the 6 original sequences of the data set ordered from the first to the last one. Third row: samples of a synthesized sequence, according to the interpolation algorithm described in section 3.2. (See uploaded movie.)*

where, if we call $\mathbf{x}(t) = H(t)\mathbf{x}$, $D(\mathbf{x}, t)$ is given by

$$\frac{1}{x_3(t)} \begin{bmatrix} \mathbf{x}^T & 0 & -\frac{x_1(t)}{x_3(t)}[x_1,\ x_2] \\ 0 & \mathbf{x}^T & -\frac{x_1(t)}{x_3(t)}[x_1,\ x_2] \end{bmatrix} \in \mathbb{R}^{3\times 8}. \tag{12}$$

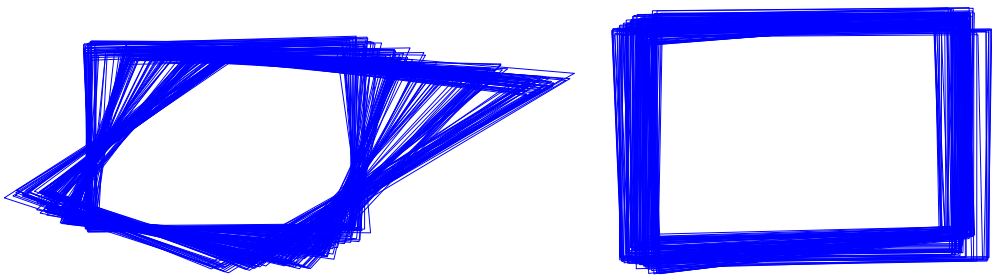
Equation (11) entails a slight abuse of notation, since it applies to H represented as a $9-$dimensional vector (rather than a 3×3 matrix), with the component $H_{3,3}$ set to one.

Once the model is identified, and $\hat{C}$, $\hat{A}$ are given, one can easily generate novel sequences by choosing an arbitrary camera path $\{(R(t), T(t))\}_{t=1,2,...}$, sampling an input noise $v_\xi \sim \mathcal{N}(0, \sigma_\xi)$, and generating $H(t) = R(t) + T(t)\nu^T$. The sequence of images $y_{synth}(t)$ is then produced by iterating the model (5) forward in time starting from an arbitrary initial condition. Indeed, by allowing more general changes in $H(t)$ one can simulate changes of the internal parameters of the camera and simulate changes in focal length (zoom), aperture (field of view) etc. Since the synthesis phase is computationally trivial, the viewpoint can be manipulated interactively (in real time), for instance from an input device with six degrees of freedom, such as a "space-ball" or a joystick.

In the next section we show the results of applying this algorithm to a real image sequences.

4 Experiments

To test the approach described in Section 3.2, we collected a data set of 6 sequences of 120 colored frames of 350×240 pixels, of what we call the inverted-fountain. From the first to the last sequence



(a) (b)

Figure 4: *Superposition of the frame boundaries after the homography registration for the fountain-corner sequence (a), and the waterfall sequence (b). Visualization is limited to the area inside every frame boundary.*

the camera is approximately sampling a circular trajectory that pans around the inverted-fountain, see Figure 3 for a sample of the data set. Before applying the algorithm, we transformed color data to 256 gray levels and down-sample the frames for computational reasons, and we set to 50 the dimension of the state of model (7). In the synthesized movie, the camera motion corresponds to a trajectory in model space that connects the models, one after the other, from the first to the last, and goes back and forth a couple of times. The interpolation among two models is linear. The resulting movie appears to be made by a camera that is panning smoothly around the inverted-fountain on a circular trajectory, as we would have expected to see. The movie is 200 frames long, and the frame dimension is 175×120 pixels. (See uploaded movie.)

We tested the procedure described in Section 3.3 with two sequences that we call fountain-corner and waterfall. The former has 170, and the latter 130 colored frames of 350×240 pixels. The state dimension of the model (5) was set to 50. The estimated homography, which ideally registers the moving images onto a common (projective) reference, is visualized in Figure 2, as well as Figure 4, that shows the superposition of the frame boundaries registered with respect to the first frame using $\hat{H}(t)$. Figures 5 and 6 show samples of the original sequences fountain-corner and waterfall in the top row, the same samples after the rectification with respect to the estimated homographies in the middle raw, and some synthesized frames in the last row. The synthesized movies are 200 frames long, and the frame dimension is 175×120 pixels. Only the pixels in the common superposition are displayed. (See uploaded movie.)

As it appears from the movie uploaded with the submission, the latter approach provides full control over the camera motion synthesis. In the fountain-corner synthesized movie the camera is first zooming in, then translating to the left, turning to the left, right, and finally zooming out. In the waterfall synthesized movie, the camera is first zooming in, than translating to the left, down, right, up, and finally rotating to the left. Besides the camera motion editing power of the latter approach with respect to the former, it is also noticeable how this approach provides better synthesized dynamics of the scene, while the interpolation in model space tends to somehow blur the images so that dynamics becomes harder to perceive. Notice that, while the waterfall sequence can be well approximated by a planar scene, the fountain-corner sequence contains objects on two planes (the water plane and the edge of the fountain). Therefore, the synthesis based on the homographic approximation only works well in a neighborhood of the original trajectory.

Notice that the technique described here can also be used to stabilize scenes with complex dynamics. For instance, the original movie of the waterfall is very jittery (due to the hand-held camera), but in the

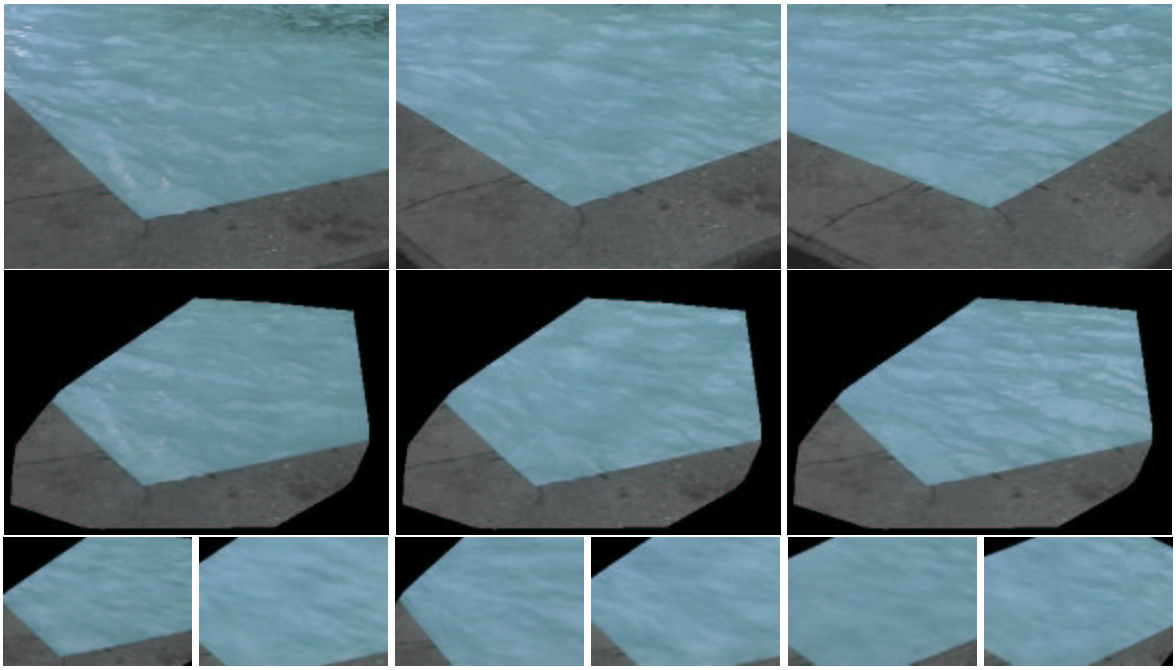


Figure 5: **Fountain-corner.** *From top to bottom: samples of the original sequence, corresponding samples after the homography registration, samples of a synthesized sequence. The training sequence is composed by 170 frames, and the synthesized movie is 200 frames long. The technique described in section 3.3 can be used to register and stabilize scenes with complex dynamics, and therefore generate smooth camera motions from jittery input data. (See uploaded movie.)*

synthesis phase one can generate arbitrary smooth motions while preserving the dynamic appearance of the waterfall.

5 Conclusions

We have presented an algorithm for identifying a model of the spatio-temporal dynamics of a non-rigid scene seen from a moving camera. Although the model does not capture the physics of the scene, it is sufficient to extrapolate the appearance of the images in space and time, and to allow changing the viewpoint arbitrarily during the synthesis.

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, MA, 1991.

[2] S. Avidan and A. Shashua. Novel view synthesis in sensor space. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 1034–1040, 1997.

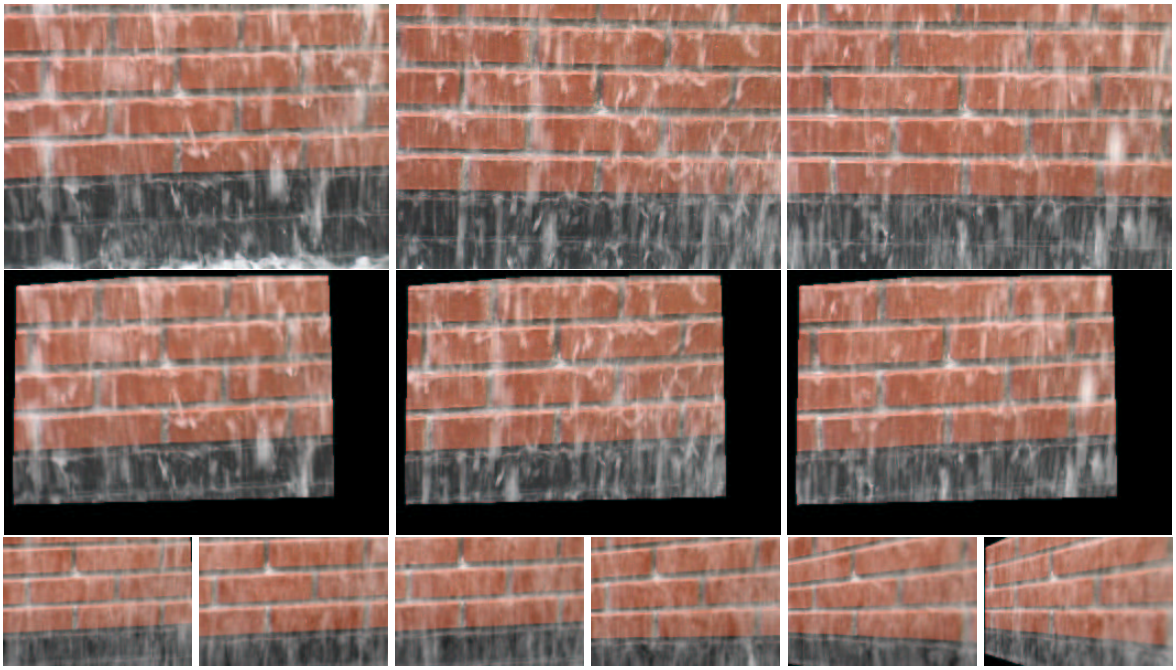


Figure 6: **Waterfall.** *From top to bottom: samples of the original sequence, corresponding samples after the homography registration, samples of a synthesized sequence. The training sequence is composed by 130 frames, and the synthesized movie is 200 frames long. Since this scene is well approximated by a plane, it can be rendered from an arbitrarily moving vantage point. (See uploaded movie.)*

[3] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, 2001.

[4] M. Brand. Voice puppetry. In *Proc. SIGGRAPH '99*, pages 21–28, 1999.

[5] M. Brand. Subspace mappings for image sequences. In *Proc. Workshop Statistical Methods in Video Processing*, June 2002.

[6] C. Bregler, M. Covell, and M. Slaney. Video rewrite: driving visual speech with audio. In *Proc. SIGGRAPH '97*, pages 353–360, 1997.

[7] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH '93*, pages 279–288, 1993.

[8] K. De Cock and B. De Moor. Subspace angles between linear stochastic models. In *Proc. 39th Int. Conf. on Decision and Control*, volume 2, pages 1561–1566, Dec 2000.

[9] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in image sequences. In *Proc. 6th Int. Conf. on Computer Vision*, pages 317–322, 1998.

[10] A. W. Fitzgibbon. Stochastic rigidity: image registration for nowhere-static scenes. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 662–669, July 2001.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Choen. The lumigraph. In *Proc. SIGGRAPH '96*, pages 43–54, 1996.

[12] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *Proc. 5th Int. Conf. on Computer Vision*, pages 605–611, 1995.

[13] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. SIGGRAPH '96*, pages 161–170, 1996.

[14] R. A. Manning and C. R. Dyer. Interpolating view and scene motion by dynamic view morphing. In *Proc. Computer Vision and Pattern Recognition Conf.*, volume 1, pages 388–384, 1999.

[15] R. C. Nelson and R. Polana. Qualitative recognition of motion using temporal texture. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 56(1):78–89, July 1992.

[16] P. Van Overschee and B. De Moor. N4sid: subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30:75–93, Jan 1994.

[17] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proc. SIGGRAPH '00*, pages 489–498, July 2000.

[18] S. M. Seitz and C. R. Dyer. View morphing. In *Proc. SIGGRAPH '96*, pages 21–30, 1996.

[19] H.-Y. Shum, A. Kalai, and S. M. Seitz. Omnivergent stereo. In *Proc. 7th Int. Conf. on Computer Vision*, pages 22–29, 1999.

[20] S. Soatto, G. Doretto, and Y.-N. Wu. Dynamic textures. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 439–446, July 2001.

[21] M. Szummer and R. W. Picard. Temporal texture modeling. In *Proc. Int. Conf. on Image Processing*, volume 3, pages 823–826, 1996.

[22] Y. Z. Wang and S. C. Zhu. A generative method for textured motion analysis and synthesis. In *Proc. European Conf. on Computer Vision*, volume 1, pages 583–597, June 2002.

[23] L. Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. SIGGRAPH '00*, pages 479–488, 2000.

[24] Y. Yacoob and L. Davis. Computing spatio-temporal representations of human faces. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 70–75, 1994.