Tracking and Rendering using Dynamic Textures on Geometric Structure from Motion

Dana Cobzas and Martin Jagersand

Department of Computing Science, University of Alberta, Canada
http://www.cs.ualberta.ca/˜ {dana,jag}

Abstract. Estimating geometric structure from uncalibrated images accurately enough for high quality rendering is difficult. We present a method where only coarse geometric structure is tracked and estimated from a moving camera. Instead a precise model of the intensity image variation is obtained by overlaying a dynamic, time varying texture on the structure. This captures small scale variations (e.g. non-planarity of the rendered surfaces, small camera geometry distortions and tracking errors). The dynamic texture is estimated and coded much like in movie compression, but parameterized in 6D pose instead of time, hence allowing the interpolation and extrapolation of new poses in the rendering and animation phase. We show experiments tracking and re-animating natural scenes as well as evaluating the geometric and image intensity accuracy on constructed special test scenes.

A problem of significant interest is how to capture and represent the image information from several sample views of a scene for the purpose of generating views from novel directions of that same scene. Such methods have applications in the confluence of vision and graphics where real scenes, too complex or tedious to model with conventional techniques, can instead be captured from photos or video and included in graphics renderings and animations.

Image-based rendering techniques focus on the ray set and generate images by mapping pixels from an original set of sample views without having a precise geometric model of the scene. There are two main approaches to this problem. One is sampling the plenoptic function under some viewing constraints - limited camera motion inside a bounding box [6, 11] or to only rotations [12]. Another approach is presented in [10], where a photoconsistent volumetric model is reconstructed from a set of input images by organizing the rays. These methods will theoretically generate correct renderings but are practically hard to apply for real scene and require calibrated cameras.

Another approach is to reconstruct a projective, affine or metric model from the input views using traditional structure from motion techniques. New renderings are generated by mapping the texture from the source images on the model. In most of the cases the metric structure is reconstructed based on planar surfaces [18] or lines [4] and the process is tedious and require a lot of human intervention. In order to have a valid texture reprojection, the model has to be decomposed in small planar patches. An advantage of metric reconstruction is that the model can be correctly reprojected under perspective projection. Non-euclidean models are more easy to construct from a set of input images [5, 21], but without additional metric information it is difficult to specify physically correct novel views.

Two significant practical challenges are: (1) Corresponding points are needed when acquiring models from images. In principle, accurate geometric structure can be obtained if the correspondences of all points in a scene between successive images can be determined. In practice, usually only a few points can be tracked reliably and accurately and object structure estimated at best coarsely. (2) In conventional texture-based rendering, the placement of triangles so that real edges on the object are aligned with edges in the model is critical. However, with a sparse model obtained from images of an otherwise unknown object this is difficult to ensure.

In this paper we present a two stage method which computes a coarse structure from motion, and then compensates for inaccuracies in the structure by modeling residual image variation as a dynamic, time varying texture on the coarse structure (see Figure 1). Both the structure and dynamic texture variation is parameterized in terms of object-camera pose, hence allowing reprojection and rendering in new positions.

In [1] a mixture model is introduced where intensity variation is modeled as a linear combination of iconic, form (motion), specular and illumination changes. We extend this by considering variation not on the image plane but on an object surface approximation. We also derive a connection between geometric plane homography warps and spatial image derivatives and show how image motion resulting from the warps can be expressed in the same way as iconic variation [13], i.e. using a set of specially tuned "eigen-filters" here optimized for capturing image motion instead of object appearance. Note that, as we show in [9], these eigen-filters parameterize motion in directly in image intensity space, unlike other approaches [2, 1], where the optic flow field has to be computed.

We present experimental results that compare the dynamic texturing with a regular static texture rendering and measuring errors in image intensities of the rendered images and geometric pixel errors of corresponding points.

1 Theory

The presented method generates new views by warping a *dynamic texture* on the projection of an estimated *structure* of the scene (see Figure 1). The input to the algorithm is a sequence of images $I(t)$ and tracked fiducial points $\mathbf{p}(t) = (\mathbf{u}(t), \mathbf{v}(t))$ grouped in quadrilateral regions. More precisely, each image I is composed from Q quadrilaterals I_q:

$$I = \sum_{q=1}^{Q} I_q \tag{1}$$

where each quadrilateral patch is obtained by warping its corresponding dynamic texture I_{wq} (Equation 3) from a standard shape (rectangle) to the desired position specified by the projection $\mathbf{u}_q, \mathbf{v}_q$ of its corresponding fiducial points P in the affine structure (Equation 4) through a homography (Equation 2, Section 1.2).

$$I_q = I_{wq}(H(\mathbf{u}_q, \mathbf{v}_q)) \tag{2}$$

$$I_{wq} = B_q \mathbf{y}_q + \bar{I}_q \tag{3}$$

$$\begin{bmatrix} \mathbf{u}_q \\ \mathbf{v}_q \end{bmatrix} = RP + \begin{bmatrix} a \\ b \end{bmatrix} \tag{4}$$

Equation 3 represents the dynamic texture decomposition into its components on an estimated basis that capture the image variability caused by nonplanarities and illumination changes, algorithm described in Subsection 1.3. Equation 4 describes the reprojection of the estimated affine structure P in the image position specified by a scaled rotation R and the image components of the translation a, b. Subsection 1.1 describes the proposed factorization algorithm that is estimating the affine structure of the

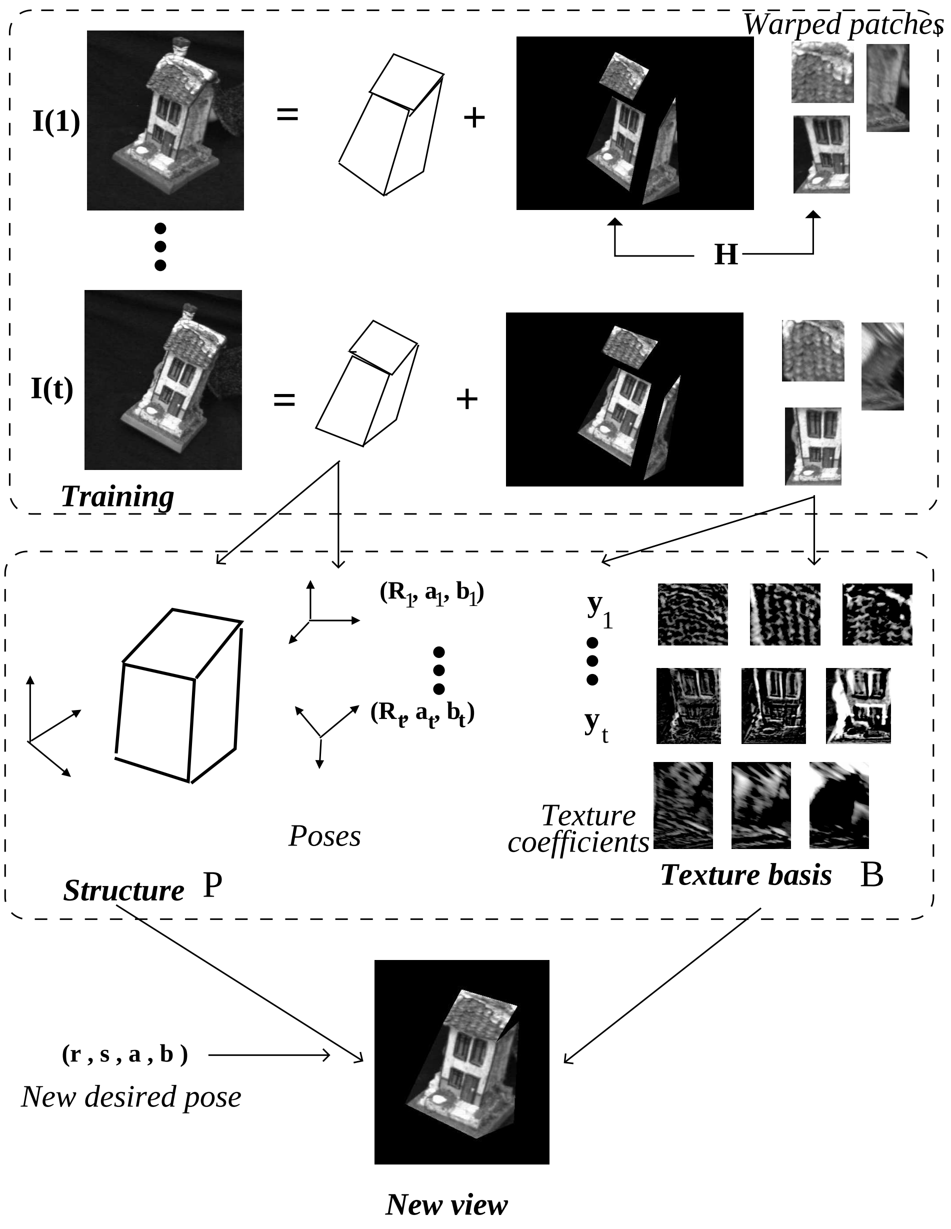

Fig. 1. A sequence of training images $I(1) \cdots I(t)$ is decomposed into geometric shape information and dynamic texture for a set of quadrilateral patches. The scene structure P and motion (r, s, a, b) is determined from the projection of the structure using a factorization algorithm. The dynamic texture for each quadrilateral is decomposed into its projection $\mathbf{y}$ on an estimated basis B. For a given desired position, a novel image is generated by warping new texture synthesized from the basis B on the projected structure.

scene (fiducial points) and the motion parameters from the image sequence under weak perspective assumption.

1.1 Acquiring geometric structure from motion

There are several techniques to build model structure from multiple uncalibrated images. In general they rely on obtaining corresponding points between images and solve for structure and motion using viewing geometry constraints. For a few images, under perspective projection assumption, this can be done using epipolar geometry (two images) or trilinear tensor (three images) [5, 14].

In the case of video, *i.e.* long motion sequences, methods which utilize all image data in an uniform way are preferred. Such methods recover affine [21, 22] or projective [19] structure using factorization approaches.

In our case of uncalibrated vision we have to pick a camera model and associated geometric framework. The methods above are formulated in either projective or affine geometry, corresponding to a perspective, weak perspective or orthographic camera assumption. In [3] we compare image based rendering using projective or affine geometry. Methods (e.g. [19, 16]) using projective camera depend on the fundamental matrix, F, between pairs of images. With only a few tracked points we found it difficult to accurately estimate F.

On the other hand, a weak perspective model allow a direct linear formulation of the viewing geometry constraints between multiple images, and using factorization, an efficient method for decomposing the image data into object structure and pose. Using multiple images allow for stable solutions despite relatively few tracked points and typical tracking errors.

Here we develop an extension of the Tomasi-Kanade factorization algorithm[21] for weak perspective camera projection model inspired by [22]. First, the algorithm recovers affine structure from a sequence of uncalibrated images. Then, a relation between the affine structure and camera coordinates is established. This is used to transform the estimated scene structure to an orthogonal coordinate frame. Finally, using similarity transforms expressed in metric rotations and translations, the structure can be reprojected into new, physically correct poses. Since we use only image information our metric unit of measure is pixel coordinates.

Weak perspective projection - a factorization approach Under weak perspective projection, a point $\mathbf{P}_i = (\mathbf{X}_i, \mathbf{Y}_i, \mathbf{Z}_i)_T$ is related to the corresponding point $p_{ti} = (u_{ti}, v_{ti})_T$ in image frame $I(t)$ by the following affine transformation:

$$\begin{aligned} u_{ti} &= s_t \mathbf{i}_t^T \mathbf{P}_i + a_t \\ v_{ti} &= s_t \mathbf{j}_t^T \mathbf{P}_i + b_t \end{aligned} \qquad (5)$$

where $\mathbf{i}_t$ and $\mathbf{j}_t$ are the components along the camera rows and columns of the rotation R_t , s_t is a scale factor and (a_t, b_t) are the first components $\mathbf{t1}_t$ of the translation $\mathbf{t}_t$ (R_t and $\mathbf{t}_t$ aligns the camera coordinate system with the world reference system).

Having N points tracked in M frames we can write

$$\begin{bmatrix} u_{11} & \cdots & u_{1N} \\ \vdots & & \vdots \\ u_{M1} & \cdots & u_{MN} \\ v_{11} & \cdots & v_{1N} \\ \vdots & & \vdots \\ v_{M1} & \cdots & v_{MN} \end{bmatrix} = \begin{bmatrix} s_1 \mathbf{i}_1^T \\ \vdots \\ s_M \mathbf{i}_M^T \\ s_1 \mathbf{j}_1^T \\ \vdots \\ s_M \mathbf{j}_M^T \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_N \end{bmatrix}^T + \begin{bmatrix} a_1 \\ \vdots \\ a_M \\ b_1 \\ \vdots \\ b_M \end{bmatrix}$$

or

$$W = RP + \mathbf{t1} \tag{6}$$

where W contains image measurements, R represents both scaling and rotation, P is the shape and $\mathbf{t1}$ is the translation in the image plane [22].

If the image points are registered with respect to their centroid in the image plane and the center of the world coordinate frame is the centroid of the shape points, the projection equation becomes:

$$\hat{W} = RP \quad \text{where} \quad \hat{W} = W - \mathbf{t1} \tag{7}$$

Rank theorem Following [21], in the absence of noise $rank(\hat{W}) = 3$. Under most viewing conditions with a real camera the effective rank is 3. Assuming $2M > N$, $\hat{W}$ can be decomposed $\hat{W} = O_1 \Sigma O_2$, where O_1 is an orthonormal $2M \times N$ matrix, Σ is an $N \times N$ diagonal matrix and O_2 is an $N \times N$ orthonormal matrix (SVD).

Defining

$$\begin{aligned} \hat{R} &= O_1' \\ \hat{P} &= \Sigma' O_2' \end{aligned} \tag{8}$$

we can write

$$\hat{W} = \hat{R}\hat{P} \tag{9}$$

O_1' is formed from the first three columns of O_1, Σ' is the first 3×3 matrix of Σ and O_2' contains the first three rows of O_2 (assuming the singular values are ordered in decreasing order).

Metric constraints The matrices $\hat{R}$ and $\hat{P}$ are a linear transformation of the metric scaled rotation matrix R and the metric shape matrix P. More specifically there exist a 3×3 matrix Q such that:

$$\begin{aligned} R &= \hat{R}Q \\ P &= Q^{-1}\hat{P} \end{aligned} \tag{10}$$

Normally, to align $\hat{P}$ with an exocentric metric frame the world coordinates of at least four scene points are needed. In our case we assume no scene information, and we instead align $\hat{P}$ with the pixel coordinate system of the camera row and column. This relates Q to the the components of the scaled rotation R:

$$\begin{aligned} &\hat{\mathbf{i}}_t^T Q Q^T \hat{\mathbf{i}}_t = \hat{\mathbf{j}}_t^T Q Q^T \hat{\mathbf{j}}_t \qquad (= s_t^2) \\ &\hat{\mathbf{i}}_t^T Q Q^T \hat{\mathbf{j}}_t = 0 \end{aligned} \tag{11}$$

where $\hat{R} = [\hat{\mathbf{i}}_1 \cdots \hat{\mathbf{i}}_M \hat{\mathbf{j}}_1 \cdots \hat{\mathbf{j}}_M]^T$ The first constraint assures that the corresponding rows $s_t \mathbf{i}_t^T$, $s_t \mathbf{j}_t^T$ of the scaled rotation R in Eq. 6 are unit vectors scaled by the factor s_t and the second equation constrain them to orthogonal vectors. This generalizes [21] from an orthographic to a weak perspective case. The resulting transformation is up to a scale and a rotation of the world coordinate system. To eliminate the ambiguity we align the axis of the reference coordinate system with the first frame and estimate only eight parameters in Q (fixing a scale). We solve this data fitting problem using Levenberg-Marquardt non-linear minimization algorithm [15].

To extract the motion parameters from each camera position, we first estimate the scale factor s_t and rotation components $\mathbf{i}_t$ and $\mathbf{j}_t$ by computing the norm of the rows in R that will represent the scale factors and then normalizing them. Considering that $\mathbf{i}_t$ and $\mathbf{j}_t$ can be interpreted as the orientation of the vertical and horizontal camera image axes in the object space, we compute the direction of the camera projection axis $\mathbf{k}_t = \mathbf{i}_t \times \mathbf{j}_t$. We now have a complete representation for the metric rotation that we parametrize with Euler angles $r_t = [\psi_t, \theta_t, \varphi_t]$

$$[\mathbf{i}_t, \mathbf{j}_t, \mathbf{k}_t]^T = \mathcal{R}(r_t) =$$
$$\begin{bmatrix} \cos\psi_t \cos\varphi_t - \cos\theta_t \sin\psi_t \sin\varphi_t & \cos\psi_t \sin\varphi_t + \cos\theta_t \sin\psi_t \cos\varphi_t & \sin\theta_t \sin\psi_t \\ -\sin\psi_t \cos\varphi_t + \cos\theta_t \cos\psi_t \cos\varphi_t & -\sin\psi_t \sin\varphi_t + \cos\theta_t \cos\psi_t \cos\varphi_t & \sin\theta_t \cos\psi_t \\ \sin\theta_t \sin\varphi_t & -\sin\theta_t \cos\varphi_t & \cos\theta_t \end{bmatrix}$$

We recover the complete pose information $X_t = [r_t, s_t, a_t, b_t]$ for each image $I(t)$ in the initial sequence and the metric structure of the scene P (remember that the image components of the translation can be represented by the coordinates of the centroid a_t,b_t).

Reprojection property Given a set of position parameters at time t, $X_t = [r_t, s_t, a_t, b_t]$, we can reproject the estimated object shape P using

$$\mathbf{p}_t = \begin{bmatrix} u_{t1} & \cdots & u_{tN} \\ v_{t1} & \cdots & v_{tN} \end{bmatrix} = s_t \mathcal{R}(r_t) P + \begin{bmatrix} a_t \\ b_t \end{bmatrix} \tag{12}$$

1.2 Warping through a homography

To apply the dynamic texture techniques described in Subsection 1.3, patches in the original images have to be warped to a standard shape. Considering that each patch is determined by four tracked points and represents a rigid part an object, it can be approximated with a planar surface and mapped to a rectangular shape through a homography. It is well known [5, 20] that, under perspective transformation, points in two planar scene views $I(t_1)$ and $I(t_2)$ are related by a 2D projective transformation (homography).

$$\begin{bmatrix} u_{1t_1} & \cdots & u_{Nt_1} \\ v_{1t_1} & \cdots & v_{Nt_1} \\ \xi_{1t_1} & \cdots & \xi_{Nt_1} \end{bmatrix} = H \begin{bmatrix} u_{1t_2} & \cdots & u_{Nt_2} \\ v_{1t_2} & \cdots & v_{Nt_2} \\ 1 & \cdots & 1 \end{bmatrix} \tag{13}$$

This transformation is up to a scale, so in general H has eight independent parameters $h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8$ and it can be computed from four corresponding points in the two views. In our case we map the quadrilateral patches to a standard rectangular shape by computing the homography defined by the four corners and mapping the interior points through this homography.

1.3 Texture variability modeling

Let $I_w(t)$ be a rectified texture patch extracted from the image stream and warped to a standard rectangle as described in the previous section. Commonly, the texture $I_w(t)$

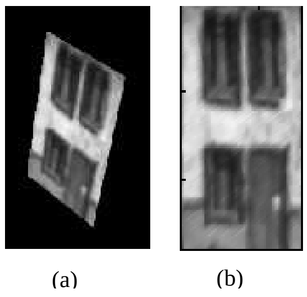

Fig. 2. (a) Original patch (b) Warped patch

is assumed constant. This is motivated by the fact that planar patches are mapped in a physically correct way to the camera plane. However, in a real case there will be errors in the estimated plane parameters, and the real world surface that we model may not be quite planar. In addition, we include illumination in our variability model. Hence, here we purposefully focus on the intensity variation. In the following we first motivate analytically that under small motions the image intensity variations in the texture can be modeled as a set of spatial basis filters modulated by the motion, then we show how to statistically estimate this basis.

Consider the distribution of texture differences $I_z(t) = I_w(t) - I_{tref}$ around a texture image I_{tref}. This distribution can be described in both statistical and structural terms. Assuming small image variation we apply an optic flow constraint, $I_z = \frac{\partial I_w}{\partial u}\Delta u + \frac{\partial I_w}{\partial v}\Delta v$, where u and v are the camera coordinates.

Structural image variability The above motion equation relates image intensity variation to a motion field. Assuming the texture patch is sourced from a rigid surface, the structure of the motion field is constrained to a low dimensional parameter space. Recall that we obtained the texture patch from a homography based on tracked points. Variation in the warping homography parameters $h_1 \dots h_8$ introduces image variability of the form:

$$\Delta \mathbf{I}_w = \sum_{i=1}^{8} \frac{\partial \mathbf{I}}{\partial h_i}\Delta h_i = \left[\frac{\partial \mathbf{I}}{\partial u}, \frac{\partial \mathbf{I}}{\partial v}\right] \begin{bmatrix} \frac{\partial u}{\partial h_1} \cdots \frac{\partial u}{\partial h_8} \\ \frac{\partial v}{\partial h_1} \cdots \frac{\partial v}{\partial h_8} \end{bmatrix} \Delta h_i = [\mathbf{B}_1 \dots \mathbf{B}_8][y_1, \dots, y_8]^T, \tag{14}$$

Where $\mathbf{I}$ is the image patch I_w flattened into a column vector. Note the form of the above equation: A weighted linear combination of eight filters only dependent on the spatial image derivatives times a constant matrix. This now reduces the space of possible image flow fields to an 8-dimensional variation.

In the above we assumed a planar patch. Real world patches will often not be planar, and this planarity will introduce a parallax error. The intensity variation due to the depth parallax can be written as a linear basis:

$$\Delta \mathbf{I}_d = [\mathbf{B}_{d1}, \mathbf{B}_{d2}][y_{d1}, y_{d2}]^T \tag{15}$$

Illumination variation It has been shown that for a convex Lambertian object, the image variability due to different illumination can be expressed as a three dimensional linear basis[17, 7]. For a general object, the illumination component can be approximated with a low dimensional basis. (In practice often three to five basis vectors suffice)

$$\Delta \mathbf{I}_i = [\mathbf{B}_{i1} \dots \mathbf{B}_{i3} \dots][y_{i1} \dots y_{i3} \dots]^T, \tag{16}$$

Composite variation model Adding up the above contributions we can differentially model the composite image intensity variation as:

$$\Delta \mathbf{I} = \Delta \mathbf{I}_w + \Delta \mathbf{I}_d + \Delta \mathbf{I}_i + \Delta \mathbf{I}_e = B\mathbf{y} + \Delta \mathbf{I}_e, \tag{17}$$

where $\Delta \mathbf{I}_e$ is a noise term. The structural model in particular is only valid for parameter changes Δh_i which are small with respect to the spatial image derivatives. By extending the basis to a scale space hierarchy[8] (where the spatial derivatives are smoothed in the coarser scales) the applicability can be widened by adding more elements to the basis $B = [\mathbf{B}_1 \dots \mathbf{B}_k]$. However, as a result, the basis grows large and indeed is likely to represent more than the actually occurring texture variation.

Statistical variability modeling In principle a variability basis B could be computed if very accurate a-priori models of the objects, lighting, cameras and their relative geometry are available.[1] In practice this is seldom the case. Instead we propose to estimate a sufficient basis $\hat{B}$. Consistent with the estimation of coarse geometry in the previous section we do this from observing an image sequence in an uncalibrated camera, $\hat{\mathbf{I}}_w(t)$, and obtaining a sample for $t = 1 \dots M$. Note that:

1. The estimated basis only represents the intensity variability actually present in the image sequence, which can be less than the possible variation as described above.
2. The estimated basis $\hat{B}$ can be any linear transform of the analytically derived basis B above. It can also contain basis vectors that in addition to what is captured in B represents other types of variability.

A standard technique to estimate a linear basis $\hat{B}$ which best captures (in a least squares sense) the part of B actually present in our observed intensity sequence is principle component analysis. The variation in texture is considered a stationary process with temporal mean $\bar{\mathbf{I}} = \sum_{t=0}^{M} \frac{1}{M} \mathbf{I}_w(t)$. Let the zero mean texture be $\hat{\mathbf{I}}_z(t) = \mathbf{I}_w(t) - \bar{\mathbf{I}}$, and a measurement matrix $A = [\mathbf{I}_z(1), \dots, \mathbf{I}_z(M)]$. The principle components are the eigen vectors of the covariance matrix $C = AA^T$. A dimensionality reduction is achieved by keeping only the first k of the eigenvectors.

For practical reasons, usually $k \ll M \ll l$, where l is the number of pixels in the texture patch, and the covariance matrix C will be rank deficient. We can then save computational effort by instead computing $L = A^T A$ and eigen vector factorization $L = VDV^T$, where V is an ortho-normal and D a diagonal matrix. From the k first eigenvectors $\hat{V} = [\mathbf{v}_1 \dots \mathbf{v}_k]$ of L we form a k-dimensional eigenspace $\hat{B}$ of C by $\hat{B} = A\hat{V}$. Using the estimated $\hat{B}$ we can now write a least squares optimal estimate of any intensity variation in the patch as

$$\Delta \mathbf{I} = \hat{B}\hat{\mathbf{y}}, \tag{18}$$

[1] One of the components, ΔI_w, can be computed only from spatial image derivatives. In tracking this commonly used for a low dimensional (in practice 2-4) variability model of planar warps[7]. However, the 8 derivatives for the homography parameters are difficult to compute accurately from only one image.

the same format as Eq. 17, but without using any a-priori information to model B. While $\hat{\mathbf{y}}$ captures the same variation as $\mathbf{y}$, it is not parameterized in the same coordinates, so in addition we estimate a second transform J between our pose description and $\hat{\mathbf{y}}$. In our application we represent one object using several texture patches, and estimate J between texture mixing coefficients $\hat{\mathbf{y}}$ and tracked global camera-object pose $\hat{\mathbf{x}}$ ($= [r, s, a, b]^T$).

For every training image $\mathbf{I}_t$ we have from the orthogonality of $\hat{V}$ that the corresponding texture mixing are the columns of $[\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_M] = \hat{V}^T$. From the factorization of geometric structure we also have the corresponding $\hat{\mathbf{x}}_t$. Estimating a linear model $\varDelta\hat{\mathbf{y}} = J\varDelta\hat{\mathbf{x}}$, where $\varDelta\hat{\mathbf{y}}_t = \hat{\mathbf{y}}_t - \hat{\mathbf{y}}_{ref}$ we need 6 or more motions $\varDelta\hat{\mathbf{y}}, \varDelta\hat{\mathbf{x}}$ (From one center reference and 6 nearby training images) to solve for J in

$$\begin{bmatrix} \varDelta\hat{\mathbf{y}}_1 \\ \vdots \\ \varDelta\hat{\mathbf{y}}_m \end{bmatrix} = \begin{bmatrix} \varDelta\hat{\mathbf{x}}_1 \\ \vdots \\ \varDelta\hat{\mathbf{x}}_m \end{bmatrix} J^T \tag{19}$$

The linear model $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{ref} + J\varDelta\hat{\mathbf{x}}$ is only locally valid and has to be recomputed around each desired pose $\hat{\mathbf{x}}_{ref}$. A globally valid spline approximation $\hat{f}$, s.t. $\hat{\mathbf{y}} = \hat{f}(\hat{\mathbf{x}})$ can be computed instead if, in the training samples, $\hat{\mathbf{x}}_t$ is monotonic and plaid. For a linear spline, this corresponds to organizing the computed matrices J into an indexable array.

1.4 Interpretation of the variability basis

In our application, the geometric model captures gross image variation caused by large movements. The remaining variation in the rectified patches is mainly due to:

1. Tracking errors as well as errors due to the weak perspective approximation cause the texture to be sourced from slightly inconsistent locations in the training images. These errors can be modeled as a small deviation $\varDelta[h_1, \ldots, h_8]^T$ in the homography parameters from the true homography, and causes image differences according to Eq. 14. The weak perspective approximations, as well as many tracking errors are persistent, and a function of object pose. Hence they will be captured by $\hat{B}$ and indexed in pose $\hat{\mathbf{x}}$ by f.
2. Depth variation is captured by Eq. 15. Note that Z and $\varDelta Z$ is depth along the camera optic axis, and hence also varies as a function of object pose.
3. Assuming fixed light sources and a moving camera or object, the light variation is a function of relative camera-object pose as well.

From the form of Equations 14 and 15 we expect that pose variations in the image sequence will result in an texture variability described by combinations of spatial image derivatives. In Fig.3 we compare numerically calculated spatial image derivatives to the estimated variability basis B.

In synthesizing texture to render a sequence of novel images the function f modulates the filter bank B so that the new texture dynamically changes with pose $\hat{\mathbf{x}}$ according to $\mathbf{I}_w = Bf(\hat{\mathbf{x}}) + \bar{\mathbf{I}}$.

2 Composition of geometric and image-based models

Based on the theory described in Section 1 we have developed an algorithm that generates new views using an estimated geometric model and a dynamic texture mapping.

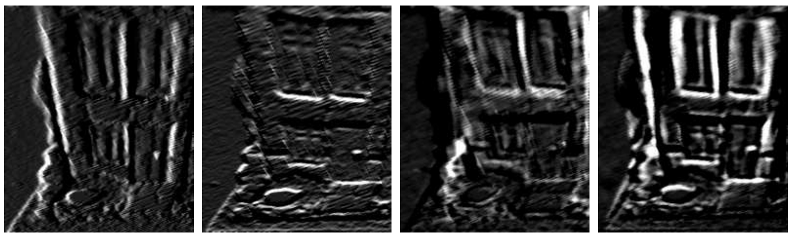

Fig. 3. Comparison between spatial derivatives $\frac{\partial I_w}{\partial x}$ and $\frac{\partial I_w}{\partial y}$ (left two texture patches) and two vectors of the estimated variability basis $[\mathbf{B}_1, \mathbf{B}_2]$ (right) for the house in Fig. 1.

Training data We took sequences of M images $I(t)$ and tracked N fiducial points $p_t = [\mathbf{u}_t, \mathbf{v}_t]^T$ using SSD trackers from XVision system [7]. The tracked points are grouped into disjunctive quadrilaterals that will cover the scene. For a simple squared object, it is possible to find a decomposition into quadrilateral regions that correspond to physical planes. In practice, considering also the limitation of the tracking algorithm, we are decomposing the object into nonplanar regions. The dynamic texture algorithm will correct the reprojection errors due to nonplanarities.

Structure from motion We use the factorization algorithm from Section 1.1 to acquire the structure of the scene P and calculate the pose $\mathbf{x}_t$.

1. Form the normalized measurement matrix $\hat{W}$ by registering the image coordinates with respect to their centroid.
2. Compute SVD of $\hat{W}$ and the approximate rotation $\hat{R}$ and shape $\hat{P}$ Equation 8.
3. Impose metric constraints from Equation 11 to calculate the true scaled rotations R and shape P.
4. Estimate the camera pose $[r_t, s_t, a_t, b_t]$ for each view. s_t is the norm of the rows corresponding to $\mathbf{i}_t$ and $\mathbf{j}_t$ in R. $r_t = [\psi_t, \theta_t, \varphi_t]$ are the Euler angles corresponding to the rotation matrix formed from $\mathbf{i}_t$, $\mathbf{j}_t$ and $\mathbf{k}_t = \mathbf{i}_t \times \mathbf{j}_t$.

Dynamic texture For each quadrilateral I_q we compute a texture basis B_q and a set of texture coefficients $\mathbf{y}_q$:

1. Warp the patch I_{qt} to a standard shape Iw_{qt} using the homography (see Section 1.2) determined by the four corners. We choose the biggest rectangle that includes the patch through the image sequence as the standard shape.
2. Form a zero mean sample and compute the PCA as described in Section 1.3. Keep k, typically about a dozen basis vectors B_q and the corresponding coefficients in each frame $\mathbf{y}_{qt}$.

New view rendering Given a new pose $\mathbf{x} = [r, s, a, b]^T$ we render the dynamic texture on the reprojected structure.

1. Compute reprojection $p = [\mathbf{u}, \mathbf{v}]^T$ of shape P in the desired pose using Equation 12.
2. For each quadrilateral q

(a) interpolate a texture coefficient $\mathbf{y}_q$ corresponding to the new pose $\mathbf{x}$ using the nearest neighbors in the training data,
(b) compute the texture in the standard shape using Equation 18,
(c) rewarp the texture in the desired image position determined by the reprojected corners.

3 Experimental results

To test the proposed algorithm we recorded motions of three types of objects: a calibration pattern (`pattern`), a toy house (`house`) and a flower (`flower`). We tested each for several sequence lengths between 15 and 256 images. The first two objects can be relatively accurately decomposed into a few planar patches. However, this is not the case for the flower. The motion was mostly composed by rotations with some depth variation. We tracked between 8 and 15 points using XVision [7] and grouped them into 3 or 4 non-overlapping quadrilaterals.

We compute the metric structure and pose for the tracked points using the algorithm from Section 1.1 and the texture basis and coefficients for each quadrilateral. As mentioned in Section 1.3, a minimum of 13 basis vectors will capture the errors caused by depth and illumination changes under small motions. In a real case with significant motion variation, and other errors like inaccuracies in tracking, more vectors are required to capture the image variability. In our case we kept about 25 texture basis vectors for the long sequences. For the shorter sequences, where the actual variation in motion was limited, we found that 3 to 5 vectors were enough. From the estimated model, we generated three types of new image sequences by:

1. Interpolating the original motion. New pose points were generated between the original poses and a longer temporally up-sampled movie of both the original and new poses was rendered from the model.
2. Smoothing the original motion. In this case we computed the original pose trajectory, smoothed it by polynomial fitting, and rendered a new video sequence with smooth motion.
3. Perturbing the original rotations by user supplied values (we used up to $5°$-$15°$) and rendering a motion edited animation.

Figure 4 shows some examples of rendered picture from the `house` sequence. The motion is limited by the fact that the factorization algorithm requires that the fiducial points are visible in all the frames. To generate larger motions we composed renderings from two sequences taken from different view angles. Another solution would be to incrementally update the model when introducing a new view based on the common fiducial points.

For testing the performance of our algorithm, we compared the *dynamic texture* algorithm with a *static texture* algorithm. For generating the static texture sequences we warped texture from original images onto the reprojected metric structure. Considering that the dynamic texture algorithm requires a mean image and k basis texture images, we source the static texture from $k + 1$ original images. The source images are equally spaced through the sequence and the texture for the current generated image will be textured from the closest source image. There are two types of errors that we are investigating for the generated image sequence - *static errors* in individual rendered image frames and *dynamic errors* through the whole sequence.

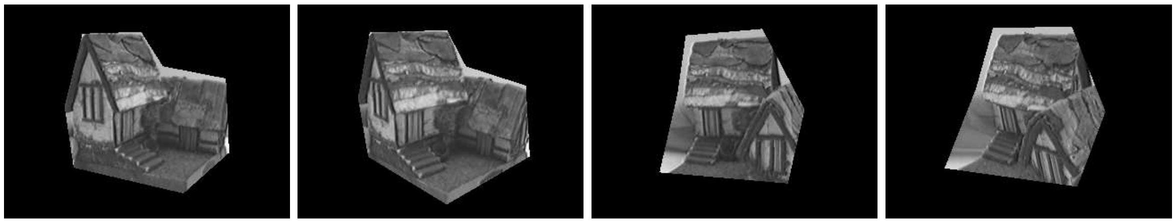

Fig. 4. Generated pictures from `house` sequence

3.1 Static errors

Most of the static errors are caused by nonplanarities of the texture patches that will result in a misalignment in the generated images. Figure 5 shows a rendered picture from the `pattern` sequence using static texture (left) and dynamic texture (right). Notice the effect of bent horizontal line in the left image, that was corrected using the dynamic texture in the right image. The scene was decomposed into three quadrilaterals two planar on the sides and one nonplanar in the middle, where the pattern has a corner. The nonplanar region in the middle is causing this distortion. A similar effect can be seen in Figure 6, which shows pictures generated from the `house` sequence. The small house (middle of the zoomed picture) appear broken in the case when using static texture.

While in the case of the pattern or house it can be argued that these types of scenes can be better decomposed into planar surfaces, this would not be possible for many scenes with natural instead of man-made objects. As an example we show the `flower` sequence. Figure 7 shows the quadrilateral decomposition (left), a rendered image using static texture with visible geometric errors (middle) and a rendered image from the same pose using dynamic texture (right) where the geometric errors are compensated.

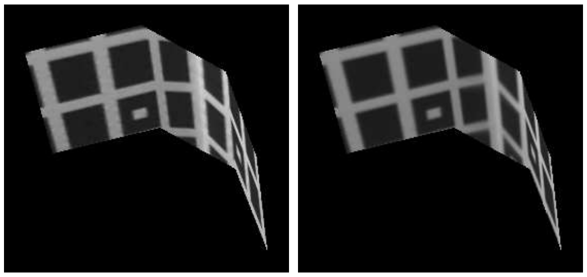

Fig. 5. Geometric errors (Left) Static texture: broken line caused by nonplanar patch; (Right) Dynamic texture: error was compensated

To quantify the geometric errors we regenerate the original positions from the `pattern` sequence and compare them with the original images by measuring the differences in pixel intensities (Figure 8). Notice that the error was almost constant in the case of dynamic texture and very uneven in the case of static texture. For the static texture case we used frame 0,5,9,13 for sourcing the texture (consistent with using three texture basis vectors in the dynamic case) so is expected that the error drops to zero when reproducing these frames. The mean relative intensity error was 1.17% in the case of static texture and 0.56% in the case of dynamic texture.

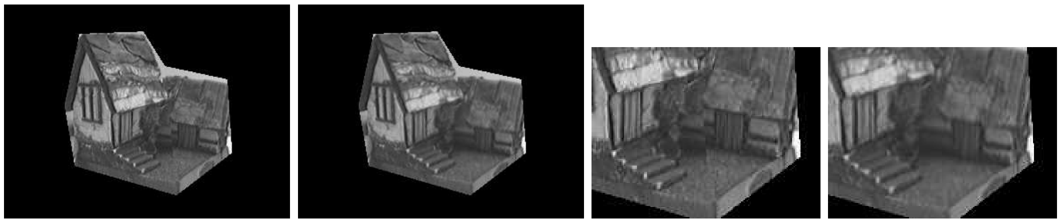

Fig. 6. Geometric errors in `house` sequence. (Left) Rendered images using static and dynamic texture respectively; (Right) Detail showing the geometric error

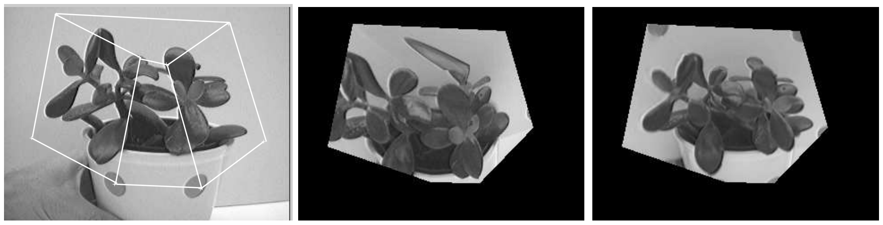

Fig. 7. Geometric errors in `flower` sequence. (Left) One of the original images and the outline of the quadrilateral patches. (Middle) Image generated using static texture. (Right) Image generated in the same pose using dynamic texture.

3.2 Dynamic errors

For an animation there are global errors through the whole movie that are not visible in one frame but in motion impression from the succession of the frames. We call these dynamic errors. We identified two types of dynamic errors connected to *depth impression* and *motion smoothness*. We again compared the static and dynamic texturing.

If the surfaces that are rendered with static texture are not physical planes, it is very hard to capture the depth impression when re-animating the object. Our dynamic texture algorithm capture the depth variation and give an impression of a "real" 3D object. This is only noticeable in a movie (refer to the attached movie `flower_animation.mpg`). Notice how the rendering with dynamic texture animates the flower to create a realistic impression of the leaves being spaced in 3D, while in the static texture it looks like the flower is a picture pasted on a few planes.

Another important quality is smoothness of motion. When using static texture we source the texture from a subset of the original images ($k + 1$ if k is the number of texture basis) so there is significant jumping when changing the texture source image. We tracked a point through a generated sequence from the `pattern` in the two cases and measure the smoothness of motion. Table 1 shows the average pixel jitter. As an appli-

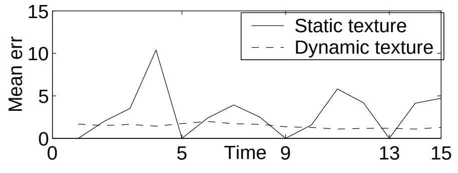

Fig. 8. Intensity pixel error in the rendered images (compared to original image)

cation to motion smoothness, we used the proposed algorithm to correct the unevenness in the original motion. The enclosed movie `flower_animation.mpg` shows the regenerated smooth motion of the original `flower` sequence.

	Vertical pixel jitter	Horizontal pixel jitter
Static texture	1.15	0.98
Dynamic texture	0.52	0.71

Table 1. Average pixel jitter

4 Discussion

The presented method allows acquisition of scene structure and appearance using video from an uncalibrated camera. The scenes can then be rendered from novel poses. Our main objectives were to balance requirements for calibration, user interaction, and computation, while rendering realistic images and motion animations. By using only coarse approximate geometric structure, this structure can be extracted from images in a computationally efficient and reliable way. The structure corresponds to the major salient object features. The user can tune the representation by focusing on the most salient structure in the scene, yet the interaction is limited to pointing out only a few (about a dozen) scene points in the first view of the scene to initialize the real-time visual tracking.

To be able to accurately capture the scene details we developed a two stage method. First, the coarse geometry is used to warp scene patches into a canonical form. Then, the residual intensity variation is captured using a locally valid linear model based on a multidimensional optic flow-like constraints. The resulting representation is a *dynamic texture* which, when modulated by overall object pose, can add fine scale variation representing fine scale scene geometry.

To validate the model we recorded and re-animated various scenes and compared our dynamic texture rendering to conventional model-based rendering using a static texture image. We found that our method reduced both intensity error and motion jitter to about half of those values obtained by conventional rendering.

Applications of our method include: (1) Animation editing, where a movie segment of a scene with motion is recorded, but the motion trajectory needs to be somewhat adjusted afterwards. We tested this for adjustments up to 15 degrees of angle. (Translation range is unlimited). (2) Motion smoothing. This can stabilize video where a scene has been recorded, for instance, from a vehicle, and the original video has vibrations and bumps. (3) Visual modeling in general when done for representing intensity variation rather than obtaining detailed geometry.

Currently, in one model, we only represent the scene regions which are simultaneously visible in the whole video sequence. To achieve larger motions we have to piece together several models. In future work we plan to extend our work to a panoramic camera, using cylindrical texture images to capture interior of a whole indoor scene in one model.

References

1. M. Black, D. Fleet, and Y. Yacoob. Robustly estimating changes in image appearance. *Computer Vision and Image Understanding*, 78(1):8–31, 2000.
2. M. Black, Y. Yacoob, A. Jepson, and D. Fleet. Learning parameterized models of image motion. In *Computer Vision and Pattern Recognition*, pages 561–567, 1997.
3. D. Cobzas and M. Jagersand. A comparison of non-euclidean image-based rendering. In *Proceedings of Graphics Interface*, 2001.
4. P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from phtographs. In *Computer Graphics (SIGGRAPH'96)*, 1996.
5. O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Boston, 1993.
6. S. J. Gortler, R. Grzeszczuk, and R. Szeliski. The lumigraph. In *Computer Graphics (SIGGRAPH'96)*, pages 43–54, 1996.
7. G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.
8. M. Jagersand. Saliency maps and attention selection in scale and spatial coordinates: An information theoretic approach. In *Fifth International Conference on Computer Vision*, 1995.
9. M. Jagersand. Image based view synthesis of articulated agents. In *Computer Vision and Pattern Recognition*, 1997.
10. K. Kutulakos and S. Seitz. A theory of shape by shape carving. *International Journal of Computer Vision*, 38:197–216, 2000.
11. M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics (SIGGRAPH'96)*, pages 31–42, 1996.
12. L. McMillan and G. Bishop. Plenoptic modeling: Am image-based rendering system. In *Computer Graphics (SIGGRAPH'95)*, pages 39–46, 1995.
13. H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.
14. M. Pollyfeys. *Tutorial on 3D Modeling from Images*. Lecture Nores, Dublin, Ireland (in conjunction with ECCV 2000), 2000.
15. W. H. Press, B. Flannery, S. A. Teukolsky, and W. T. V. tterling. *Numerical Recipies in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1992.
16. S. M. Seitz and C. R. Dyer. View morphing. In *Computer Graphics (SIGGRAPH'96)*, pages 21–30, 1996.
17. A. Shashua. *Geometry and Photometry in 3D Visual Recognition*. PhD thesis, MIT, 1993.
18. I. Stamos and P. K. Allen. Integration of range and image sensing for photorealistic 3d modeling. In *ICRA*, 2000.
19. P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *ECCV (2)*, pages 709–720, 1996.
20. R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, pages 22–30, March 1996.
21. C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.
22. D. Weinshall and C. Tomasi. Linear and incremental aquisition of invariant shape models from image sequences. In *Proc. of 4th Int. Conf. on Compute Vision*, pages 675–682, 1993.