View Dependent Texturing using a Linear Basis

Martin Jagersand, Neil Birkbeck and Dana Cobzas

Abstract We present a texturing approach for image-based modeling and rendering, where instead of using one (or a blend of a few) sample images, new view dependent textures are synthesized by modulating a differential texture basis. The texture basis models the first order intensity variation due to image projection errors, parallax and illumination variation. We derive an analytic form for this basis and show how to obtain it from images. Experimentally we compare rendered views to ground truth real images and quantify how the texture basis can generate a more accurate rendering compared to conventional view dependent textures. In a hardware accelerated implementation we achieve frame rate of on regular PC's and consumer graphics cards.

1 Introduction

Texture normally means fine scale visual or tactile properties of a surface. The word is related to textile, and indeed it was used to describe the particular surface created by the the interwoven threads in a fabric. In computer graphics texturing is the process of endowing a surface with fine scale properties. Often this is used to make the visualization richer and more natural than if only the 3D geometry had been rendered. There are a wide range of computer graphics texturing approaches. Early texturing involved replicating a small texture element over the surface of an object to enrich its appearance. Commonly this is done by warping a small 2D texture element onto the structure but other variations exist including 3D texturing. Texture can also be used to model light and reflections by texturing a model with a specular highlight texture.

The focus of this chapter is on image texturing. We will study how to compose a texture image suitable for photo-realistic image-based rendering. In this case the texturing element is a comparably large image, and unlike the above mentioned

Authors are at University of Alberta, Canada http://www.cs.ualberta.ca/˜ vis

techniques, not repeated over the surface. Hence, we are to some extent transcending out of the original domain of texturing by now not only modeling a repetitive fine scale structure, but also potentially medium and large scale changes in texture, including light and geometry aspects not captured by the model. Here we will focus on aspects that are specific to image based modeling and rendering, and not treat standard issues such as implementation of 2-D warps, filtering and multi-resolution texturing. The background on basic texturing is covered in the literature[16] and recent text books[20].

One purpose of the texture representation is to capture intensity variation on the pixel level. This includes the view dependency of potentially complex light surface interactions and subpixel surface structure. Another purpose, for models captured from images, is to compensate for discrepancies between approximate captured geometry and true object surfaces. The first purpose is similar to that of Bi-directional Texture Function (BTF) representations [6] and the second similar to view-dependent textures (VDTM) [7].

In a parallel line of research Freeman, Adelson and Heeger noted that small image motions could be modulated using a fixed spatial image basis[11]. This was extended to image synthesis of whole motion sequences using a PCA basis[17], and later used to animate stochastic motion [8]. Another variation decomposes the basis into a multi-linear form, where two or more variations (e.g. light and viewpoint) are represented separately[26]. The above works all represent intensity variation on the 2D image plane, but others realized that it is more efficient to represent the intensity variation on the surface facets of a 3D triangulated model [12, 5]. Both in spirit and actual implementation all these representations are quite similar in their use of a set of basis textures/images to modulate a new texture.

The work presented in this chapter falls inbetween *relief textures* and *lightfield* approaches in its approach to photorealistic rendering. Relief textures provides an explicit geometric solution to adding 3D structure to a planar texture[21]. However, relief textures require a detailed a-priori depth map of the texture element. This is normally not available in image-based modeling if only uncalibrated camera video is used. Thus, relief textures have been mostly used with a-priori graphics models. The floating texture approach similarly to relief textures performs a geometric pertubation of the pixels at render time, but instead of a depth map uses a 2D motion vector field to drive the pertubation[9].

While initially lightfield (e.g. [18, 13]) and geometry+texture approaches to image-based rendering were disparate fields, recent work attempts to close this gap. Our work is in the intersection of the two, using a relatively dense image sampling from real-time video, and representing appearance on an explicit geometric model. Most closely related are work in the lightfield area using geometric proxies. Here the lightfield can be represented on a geometry closely enveloping an object[4]. However, in rendering textures are blended directly from input images, unlike our approach which uses an intermediate basis with well defined both geometric and photometric interpolation capabilities. In surface light fields, instead of a geometric proxy, an accurate object geometry is used. Wood et. al.[28] studies how to efficiently parameterize these light fields to capture complex reflectance, but doesn't address the issues arising from geometric misalignments, but instead relies on range scanned precise models being accurately hand-registered with calibrated imagery. On the other hand, our approach explicitly addresses the misalignment issues and builds in the necessary correction in the texturing.

The main contributions of the chapter are as follows:

1. Theoretically, we derive the analytical form of texture variation under a full perspective camera, where previous formulations have been either image-plane based or using simplified linear camera models. This variation is derived to capture misalignments between the geometric model and the texture images, parallax arising when planar model facets approximate non-planar scenes, and light variation occurring naturally in camera-based texture capture.
2. Practically, we show how the actually occurring variability in a particular texture-image sequence can be estimated, and provide an identification of the above mentioned analytically derived forms in the real data. To show that the method is practical we present an implementation allowing real-time rendering on consumer grade PC's and graphics cards.
3. Experimentally, we compare rendering results from our model to static textures, traditional view-dependent texturing, and lumigraph ray-based rendering. To compute models we use image sequences from four objects of increasing difficulty.

Our texturing method combined with standard geometry acquisition using Structure-From-Motion (SFM) or Shape from Silhouette (SFS) is particularly suited for the consumer market. Anyone with any video camera, from a $100 web cam to a good quality digital camera can capture image sequences of scenes and objects, build his or her own image-based scene models and then generate reasonable quality renderings. To stimulate use by others we provide a downloadable capture and modeling system and a renderer[2].

2 Background: image geometry

Given images of a scene or object, the 3D geometry can be recovered in a variety of ways. Classic photogrammetry recovers 3D from 2D point correspondences using calibrated cameras. In the past decade much work was devoted to 3D recovery from uncalibrated images[15]. Despite this, no system can recover accurate dense geometry robustly and reliably from general scenes. One of the few publicly accessible systems is KU Leuven's 3Dwebservice[27], for which one can upload image sets of scenes and get back 3D reconstructions. It sequences Structure-From-Motion (SFM), auto-calibration, and dense stereo. The procedure is computationally demanding and runs in parallel on a computer network. Reconstructions often take hours to complete. Practically care must be taken in selecting both scenes and viewpoints for the system to work well. Nonetheless, it is a representative of the state of the art in SFM based systems.

Shape-from-silhouettes (SFS)[24], on the other hand is a very robust method to obtain a visual hull geometry. It only requires the object silhouette and the calibration of the cameras. Besides it is quite robust to silhouette or calibration errors. In our system we implement an efficient algorithm for silhouette carving using an orthogonal ray set and the Marching Intersections[25] algorithm. This decreases storage cost and improves geometric precision (by recording silhouette intersections exactly on the rays) compared of the conventional discrete voxel representation.

For the examples in this chapter, a rough 3D geometry has been obtained using either SFM or SFS as indicated. Independent of how the geometry was obtained, but central to image-based modeling is that this 3D structure can be reprojected into a new virtual camera and thus novel views can be rendered. Starting with a set of m images $I_1 \dots I_m$ from different views of a scene, a structure of n physical scene points $X_1 \dots X_n$, and m view projections $P_1 \dots P_m$. These project onto image points $x_{j,i}$ as

$$x_{j,i} = P_j X_i \quad i \in 1 \dots n, j \in 1 \dots m \tag{1}$$

Practically, the structure is divided into Q planar facets (triangles or quadrilaterals are used in our experiments) with the points $x_{j,i}$ as node points. For texture mapping, each one of the model facets are related by a planar projective homography to a texture image; see Fig. 1.

3 Texture basis

In conventional texture mapping, one or more of the real images are used as a source to extract texture patches from. These patches are then warped onto the re-projected structure in the new view.

Instead of using one image as a source texture, here we study how to relate and unify all the input sample images into a texture basis. Let $x_{T,i}$ be a set of texture coordinates in one-to-one correspondence to each model point X_i and thus also for each view j with the image points $x_{j,i}$ above. A texture warp function $\mathscr{W}$ translates the model vertex to texture correspondences into a pixel-based re-arrangement (or warp) between the texture space I_W to screen image coordinates I.

$$T(\mathbf{x}) = I(\mathscr{W}(\mathbf{x};\mu)) \tag{2}$$

where μ are the warp parameters and $\mathbf{x}$ the image pixel coordinates. For notational clarity in later derivations, we let the warp function act on the parameter space, as is common in the computer vision literature, but less often seen in Graphics.

Common such warp functions are affine, bi-linear and projective warps. The warp function W acts by translating, rotating and stretching the parameter space of the image, and hence for discrete images a re-sampling and filtering step is needed between the image and texture spaces. Details of these practicalities can be found in Moller and Haines [20].

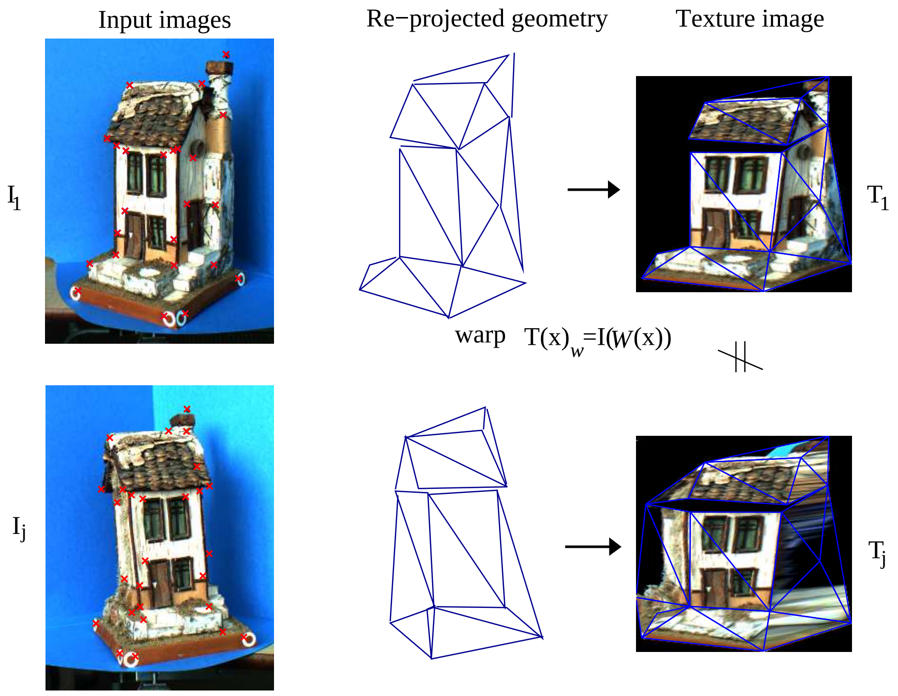


Fig. 1 Textures generated from two different images using an approximate and coarse geometry are usually different. Common problems are misalignment of texture coordinates, as visible on the right house edge, and parallax visible on windows and door.

Now if for each sample view j, we warp the real image I_j from image to texture coordinates into a texture image T_j, we would find that in general the two texture images are not identical, $T_j \neq T_k,\ j \neq k$ as illustrated in Fig. 1. Typically, the closer view j is to k, the smaller the difference is between T_j and T_k. This is the rationale for view-dependent texturing, where a new view is textured from one to three (by blending) closest sample images[7].

In this chapter we will develop a more principled approach, where we seek a texture basis B such that for each sample view:

$$\mathbf{T}_j = B\mathbf{y}_j,\ j \in 1\ldots m. \tag{3}$$

Here, and in the following, $\mathbf{T}$ is a $q \times q$ texture image flattened into a $q^2 \times 1$ column vector. B is a $q^2 \times r$ matrix, where normally $r \ll m$, and $\mathbf{y}$ is a modulation vector. The texture basis B needs to capture geometric and photometric texture variation over the sample sequence, and correctly interpolate new in-between views. We first derive a first order geometric model and then add the photometric variation. For clarity, we derive these for a single texture warp (as in Fig. 7), whereas in practical applications a scene will be composed by texturing several model facets (as in Fig. 1).

3.1 Geometric texture variation

The starting point for developing a spatial texture basis representing small geometric variations is the well known optic flow constraint, which for small image plane translations relates texture intensity change $\Delta\mathbf{T} = T_j - T_k$ to spatial derivatives $\frac{\partial}{\partial u}T, \frac{\partial}{\partial v}T$ with respect to texture coordinates $\mathbf{x} = [u,v]^T$ under an image constancy assumption[14].

$$\Delta T = \frac{\partial T}{\partial u}\Delta u + \frac{\partial T}{\partial v}\Delta v \tag{4}$$

Note that given one reference texture T_0 we can now build a basis for small image plane translations $B = [T_0, \frac{\partial T}{\partial u}, \frac{\partial T}{\partial v}]$ and from this generate any slightly translated texture $T(\Delta u, \Delta v) = B[1, \Delta u, \Delta v]^T = B\mathbf{y}$

In a real situation, a texture patch is deforms in a more complex way than just translation. This deformation is captured by the warp parameters. Given a warp function $\mathbf{x}' = \mathscr{W}(\mathbf{x}, \mu)$ we study the residual image variability introduced by the imperfect stabilization achieved by a perturbed warp $\mathscr{W}(\mathbf{x}; \hat{\mu})$, $\Delta T = T(\mathscr{W}(\mathbf{x}; \hat{\mu}), j) - T(\mathscr{W}(\mathbf{x}; \mu))$. Let $\hat{\mu} = \mu + \Delta\mu$ and rewrite as an approximate image variability to the first order (dropping j):

$$\begin{aligned} \Delta T &= T(\mathscr{W}(\mathbf{x}; \mu + \Delta\mu)) - T_W \\ &= T(\mathscr{W}(\mathbf{x}; \mu)) + \nabla T \tfrac{\partial \mathscr{W}}{\partial \mu}\Delta\mu - T_W \\ &= \nabla T \tfrac{\partial \mathscr{W}}{\partial \mu}\Delta\mu \\ &= \left[\tfrac{\partial T}{\partial u}, \tfrac{\partial T}{\partial v}\right] \begin{bmatrix} \frac{\partial u}{\partial \mu_1} \cdots \frac{\partial u}{\partial \mu_k} \\ \frac{\partial v}{\partial \mu_1} \cdots \frac{\partial v}{\partial \mu_k} \end{bmatrix} \Delta[\mu_1 \ldots \mu_k]^T \end{aligned} \tag{5}$$

The above equation expresses an optic flow type constraint in an abstract formulation without committing to a particular form or parameterization of $\mathscr{W}(\mu)$. The main purpose in the following is that Eq. 5 lets us express (small) texture pertubation due to a geometric shift, $\Delta\mu$, without the explicit pixel shifting used in [21, 9], but rather using a basis of derivative images, namely the spatial derivatives of the image texture ΔT multiplied by the Jacobian of the warp function $\frac{\partial \mathscr{W}}{\partial \mu}$. In practice, the function $\mathscr{W}$ is usually discretized using e.g., triangular or quadrilateral mesh elements. Next we give examples of how to concretely express image variability from these discrete representations.

For image-based modeling and rendering we warp real source images into new views given an estimated scene structure. Errors between the estimated and true scene geometry cause these warps to generate imperfect renderings. We divide these up into two categories, *image plane* and *out of plane* errors. The planar errors cause the texture to be sourced with an incorrect warp.[1] The out of plane errors arise when piecewise planar facets in the model are not true planes in the scene, and

[1] Errors in tracking and point correspondences when computing an SFM geometry, as well as projection errors due to differences between the estimated camera model and real camera (SFM and SFS) both cause model points to be reprojected incorrectly in images.

when rewarped into new views under a false planarity assumption will not correctly represent parallax. This can be due to the geometry being coarse (common when using SFM) and/or inaccurate (e.g. a visual hull from SFS).

Planar texture variability. First we will consider geometric errors in the texture image plane. In most IBR (as well as conventional rendering) textures are warped onto the rendered view from a source texture **T** by means of a projective homography.

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \mathscr{W}_h(\mathbf{x}_h, \mathbf{h}) = \frac{1}{1+h_7u+h_8v} \begin{bmatrix} h_1u+h_3v+h_5 \\ h_2u+h_4v+h_6 \end{bmatrix} \tag{6}$$

Rewrite Eq. 5 with the partial derivatives of $\mathscr{W}_h$ for the parameters $h_1 \dots h_8$ into a Jacobian matrix. Let $c_1 = 1+h_7u+h_8v$, $c_2 = h_1u+h_3v+h_5$, and $c_3 = h_2u+h_4v+h_6$. The resulting texture image variability due to variations in the estimated homography is (to the first order) spanned by the following spatial basis:

$$\begin{aligned} &\Delta\mathbf{T}_h(u,v) \\ &= \frac{1}{c_1}\left[\frac{\partial\mathbf{T}}{\partial u}, \frac{\partial\mathbf{T}}{\partial v}\right] \begin{bmatrix} u & 0 & v & 0 & 1 & 0 & -\frac{uc_2}{c_1^2} & -\frac{vc_2}{c_1^2} \\ 0 & u & 0 & v & 0 & 1 & -\frac{uc_3}{c_1^2} & -\frac{vc_3}{c_1^2} \end{bmatrix} \begin{bmatrix} \Delta h_1 \\ \vdots \\ \Delta h_8 \end{bmatrix} \\ &= [\mathbf{B}_1 \dots \mathbf{B}_8][y_1, \dots, y_8]^T = B_h\mathbf{y}_h \end{aligned} \tag{7}$$

Where here and thoughout the paper **y** is used for the texture modulation coefficients. Examples of the $\mathbf{B}_1 \dots \mathbf{B}_8$ derivative images can be seen in Figure 3. Similar expressions can be derived for other warps. For example, dropping the two last columns of the above Jacobian gives the variability for the affine warp.

Non-planar parallax variation In image-based modeling a scene is represented as piecewise planar model facets, but the real world scene is seldom perfectly represented by and aligned with these model planes. In rendering this gives rise to parallax errors. Figure 2 illustrates how the texture plane image T changes for different scene camera centers C. Given a depth map $d(u,v)$ representing the offset between the scene and texture plane, relief texturing [21] can be used to compute the rearrangement (pre-warp) of the texture plane before the final homography renders the new view. In image-based methods, an accurate depth map is seldom available. However, we can still develop the analytic form of the texture intensity variation as above. For a point on the model facet, let $[\alpha, \beta]$ be the angles between the normal vector and the ray pointing to the camera center C_j along the u and v axis (e.g., if $\mathbf{v} = |C_j - P| = (v_x, v_y, v_z)^T$, $\alpha = \tan^1(\frac{v_x}{v_z})$, and $\beta = \tan^{-1}(\frac{v_y}{v_z})$). The pre-warp rearrangement needed on the texture plane to correctly render this scene using a standard homography warp is then:

$$\begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \mathscr{W}_p(\mathbf{x}, \mathbf{d}) = d(u,v) \begin{bmatrix} \tan\alpha \\ \tan\beta \end{bmatrix} \tag{8}$$

As before, taking the derivatives of the warp function with respect to a camera angle change and inserting into Eq.5 we get:

$$\Delta \mathbf{T}_p(u,v) = d(u,v)\left[\frac{\partial \mathbf{T}}{\partial u}, \frac{\partial \mathbf{T}}{\partial v}\right]\begin{bmatrix}\frac{1}{\cos^2\alpha} & 0\\ 0 & \frac{1}{\cos^2\beta}\end{bmatrix}\begin{bmatrix}\Delta\alpha\\ \Delta\beta\end{bmatrix} = B_p\mathbf{y}_p \tag{9}$$

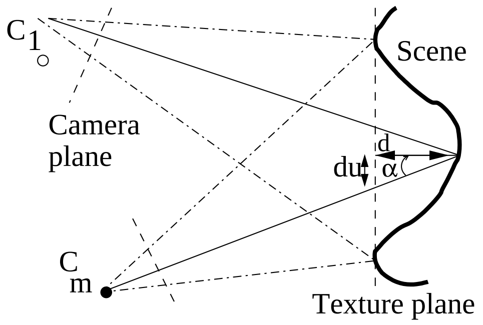


Fig. 2 Texture parallax between two views (planar representation).

3.2 Photometric variation

In image-based rendering real images are re-warped into new views, hence the composite of both reflectance and lighting is used. If the light conditions are the same for all sample images, there is no additional intensity variability introduced. However, commonly the light will vary at least somewhat. In the past decade, both empirical studies and theoretical motivations have shown that a low dimensional intensity subspace of dimension 5-9 is sufficient for representing the light variation of most natural scenes. Recently, Barsi and Jacobs [3] and Ramamoorthi and Hanrahan [22] have independently derived an analytic formula for irradiance (and reflected radiance from a convex Lambertian object) under distant illumination, explicitly considering attached shadows. They express the irradiance in terms of spherical harmonics coefficients of the illumination. An important result of their work is that Lambertian reflections acts as a low-pass filter so the irradiance lies very close to a 9D subspace.

Let (α,β) be the spherical coordinates of the distant light source and $T(\alpha,\beta,\theta,\phi)$ the intensity of the image at a point with surface normal whose spherical coordinates are (θ,ϕ). Assuming a Lambertian surface and ignoring albedo, $T(\alpha,\beta,\theta,\phi)$ can be thought as the irradiance at orientation (θ,ϕ) due to a unit directional source at (α,β). The analytical formula for T is then [3]:

$$T(\alpha,\beta,\theta,\phi) = \sum_{l=0}^{\infty}\sum_{k=-l}^{l} A_l L_{lk}(\alpha,\beta) Y_{lk}(\theta,\phi) \tag{10}$$

$$\approx \sum_{l=0}^{2} \sum_{k=-l}^{l} A_l L_{lk}(\alpha,\beta) Y_{lk}(\theta,\phi) \tag{11}$$

where $Y_{lk}(\theta,\phi)$ are the spherical harmonics, A_l is a constant that vanishes for odd $l > 1$ and $L_{lk}(\alpha,\beta)$ are the spherical harmonic coefficients of the incident illumination.

The first nine spherical harmonics and constants:

$$\begin{aligned}
(x,y,z) &= (\cos\theta\sin\phi, \sin\theta\sin\phi, \cos\phi) \\
Y_{00}(\theta,\phi) &= \sqrt{(4\pi)^{-1}} \\
(Y_{1-1};Y_{10};Y_{11})(\theta,\phi) &= \sqrt{\frac{3}{4\pi}}(y;z;x) \\
(Y_{2-2};Y_{2-1};Y_{21})(\theta,\phi) &= \sqrt{\frac{15}{4\pi}}(xy;yz;xz)
\end{aligned}$$

$$Y_{20}(\theta,\phi) = \sqrt{\frac{5}{16\pi}}(3z^2-1) \;\;,\;\; Y_{22}(\theta,\phi) = \sqrt{\frac{15}{16\pi}}(x^2-y^2)$$

$$A_0 = \pi \quad A_1 = \frac{2\pi}{3} \quad A_2 = \frac{\pi}{4}$$

Using single index notation

$$\begin{aligned}
Y_1,Y_2,Y_3,Y_4 &= Y_{00},Y_{1-1},Y_{10},Y_{11} \\
Y_5,Y_6,Y_7,Y_8,Y_9 &= Y_{2-2},Y_{2-1},Y_{20},Y_{21},Y_{22} \\
\hat{A}_j = &\begin{cases} A_0 & \text{if } j = 1 \\ A_1 & \text{if } j \in 2,3,4 \\ A_2 & \text{if } j \in 5\ldots9 \end{cases}
\end{aligned}$$

and defining $B_j(\theta,\phi) = \hat{A}_j Y_j(\theta,\phi)$, $j = 1\ldots9$, we can rewrite Eq. 10 for all the pixels in the images as:

$$\mathbf{T} = [\mathbf{B}_1 \ldots \mathbf{B}_9][L_1 \ldots L_9]^T \tag{12}$$

The image difference caused by light change can be then expressed as:

$$\Delta\mathbf{T}_l = [\mathbf{B}_1 \ldots \mathbf{B}_9][y_1 \ldots y_9]^T = B_l \mathbf{y}_l \tag{13}$$

3.3 Estimating composite variability

In textures sampled from a real scene using an estimated geometric structure we expect that the observed texture variability is the composition of the above derived planar, parallax and light variation plus other unmodeled errors. Hence we can write the texture for any sample view k, and find a corresponding texture modulation vector $\mathbf{y}_k$:

$$\mathbf{T}_k = [\mathbf{T}_0, B_h, B_p, B_l][1, y_1, \ldots, y_{19}] = B\mathbf{y}_k \tag{14}$$

where T_0 is the reference texture. Textures for new views are synthesized by interpolating the modulation vectors from the nearest sample views into a new $\mathbf{y}$, and computing the new texture $T_{new} = B\mathbf{y}$

Since this basis was derived as a first order representation it is valid for (reasonably) small changes only. In practical image-based modeling the geometric point misalignments and parallax errors are typically within 3-5 pixel, which is small enough.

Often in IBR neither dense depth maps nor light is available. Hence analytically B_p, and B_l cannot be directly analytically computed using Eq's 9 and 10. Instead the only available source of information are the sample images $I_1 \ldots I_m$ from different views of the scene, and from these, the computed corresponding textures $\mathbf{T}_1 \ldots \mathbf{T}_m$.

However, from the above derivation we expect that the effective rank of the sample texture set is the same as of the texture basis B, i.e. r$ank[\mathbf{T}_1, \ldots, \mathbf{T}_m] \approx 20$. Hence, from $m \gg 20$ (typically 100-200) sample images we can estimate the best fit (under some criterion) rank 20 subspace using e.g. PCA, SVD, or ICA. This yields an estimated texture basis $\hat{B}$ and corresponding space of modulation vectors $\hat{\mathbf{y}}_1, \ldots \hat{\mathbf{y}}_m$ in one-to-one correspondence with the m sample views. From the derivation of the basis vectors in B we know this variation will be present and dominating in the sampled real images. Hence, the analytical B and the estimate $\hat{B}$ span the same space and just as before, new view dependent textures can now be modulated from the estimated basis by interpolating the $\hat{\mathbf{y}}$ from y corresponding to the closest sample views and modulating a new texture $\mathbf{T} = \hat{B}\hat{\mathbf{y}}$.

3.4 Experimental comparison for analytical and PCA basis

For validating the equivalence between the analytical formulation of the texture basis (Sections 3.1,3.2) and the statistically estimated one (Section 3.3) we perform several experiments where we isolated different types of texture variability (planar, parallax, light) and show that the analytical basis is contained in the estimated PCA subspace.

Planar texture variation. The planar variation is usually caused by tracking inaccuracies that are about 1-5 pixels. For replicating this variability we selected a planar region from a toy house (Fig. 3 (a)) and warped it using a homography warp to a 128×128 texture (Fig. 3 (b)). We then perturbed the corners randomly with 1-5 pixels and generated another 200 textures. We calculated the analytical texture basis using Eq. 7 with the initial texture (see Fig. 3 (c1,d1,e1)) and the PCA basis for the perturbed textures. We projected the analytical basis in a subspace of PCA basis. Fig. 3 (c2,d2,e2) illustrates the recovered analytical textures from the PCA subspace. The average intensity pixel error between the original and recovered basis was 0.5%.

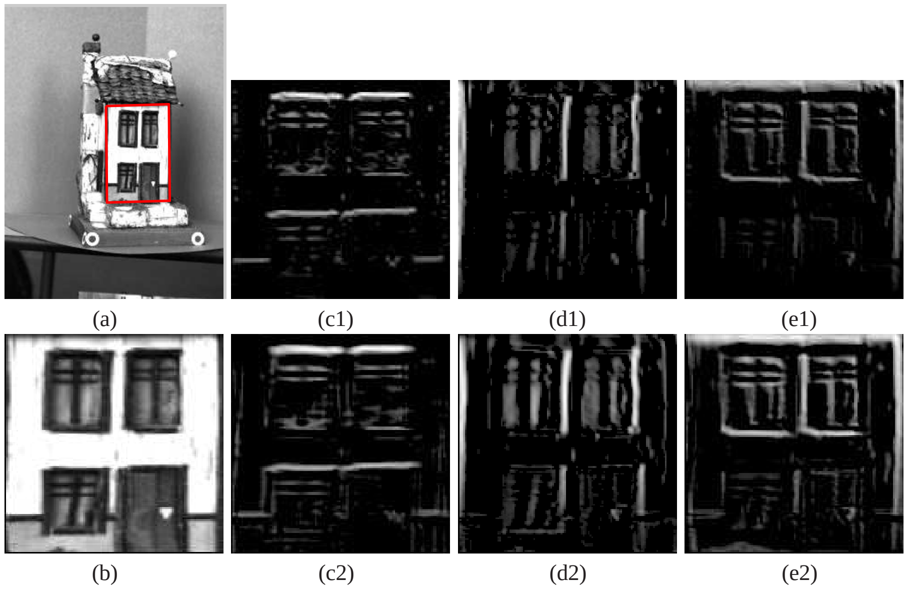


Fig. 3 Comparison of analytical and PCA basis for planar variability. (a) original quadrilateral; (b) warped texture; (c1),(d1),(e1) analytical basis (1st,4th,7th from Eq. 7); (c2),(d2),(e2) corresponding recovered basis using PCA subspace

Non-planar texture variation (parallax). The parallax variability is caused by a non-planar facet in the geometric model. For simulating this variability we captured 90 images from different pan angles of a non-planar wall from the same toy house while tracking four corners of a quadrilateral region. The corners are used to warp each quad into a standard shape for generating the texture images. Choosing a reference texture (see Fig. 4 (a)), we manually inputed the depth map (see Fig. 4 (b)) and calculated the analytical texture basis using Eq. 9. From the other sample textures we estimated a PCA subspace and projected the analytical basis into this space. Fig. 4 (c1) shows the original analytical basis (B1) and Fig. 4 (c2) shows the recovered basis from the PCA subspace.

Photometric texture variation Photometric variation is caused by changing light conditions or object rotation relative to the light source. We simulated this variability by moving a toy house on a pivot rig relative to incoming sunlight. Other forms of variation were avoided by attaching the camera to the pivot rig (i.e., the projection of the house in the sequence is fixed). A laser-scanned model was then aligned to the image sets, and the spherical harmonic functions were computed for this geometry. Figure 5 shows the first few of these analytical spherical harmonic basis functions and their reconstruction from a PCA basis that was computed from the image sequence. The similarity of the harmonics reconstructed from the PCA basis to the analytic harmonic functions illustrate that the emperical basis sufficiently encodes light variation.

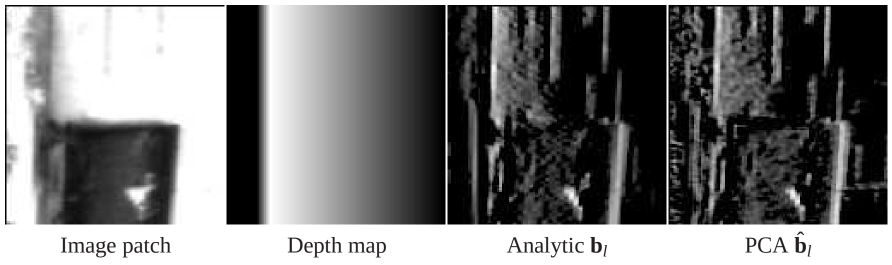


Fig. 4 Comparison of analytical form (Eq. 9) of $\mathbf{b}_l$ and the estimated PCA basis $\hat{\mathbf{b}}_l$ for parallax variability. Image patch is from the right side wall of the house, see Fig. 1 top row.

4 Rendering system implementation

Computing the texture basis involves reprojecting input images using the object or scene geometry. The geometry is usually computed from the same images, but could be obtained in some other way. We developed a software integrating the steps from images to model into a procedure taking only a few minutes in most cases, see Video 1[1] . The software is downloadable; see [2]. To quickly capture views from all sides of an object we use a rotating platform (Radio Schack TV stand). Our SW can take live video from an IEEE1394 camera, (we use a Unibrain web cam and PtGrey Scorpion 20SO) or import digital image files from a still camera. Camera calibration is obtained with a pattern, and object silhouettes through bluescreening. Light variation can be implicitly captured with viewpoint direction in a lit texture, or light direction can be separately parameterized using the image of the specular ping-pong ball in Fig. 6. The geometry is then computed as in Sect. 2. Alternatively, a separately obtained geometry can be imported. To stay with the camera-based paradigm we have used KU Leuven's 3D Webservice[27].

Texture coordinates While in computer vision it is common to texture directly from images, in applications a unified texture space is desired and often necessary. To automatically compute texture coordinates, the object geometry is first split along high curvature regions, then each region is flattened using a conformal mapping[19], and packed into an OpenGL texture square (the GUI screenshot in Fig. 6 illustrates an example of this mapping). In the dynamic texture basis computation, all input images are transformed into and processed in this space.

Texture basis generation The projection of the estimated structure into the sample images, x_j, is divided into planar facets (triangles). Practically, using HW accelerated OpenGL each frame I_j is loaded into texture memory and warped to a standard shape texture T_j based on the texture coordinate atlas. We next estimate a texture basis $\hat{B}$ and a set of texture coefficients $\hat{\mathbf{y}}_j$ by performing PCA on the set of zero mean textures $[T_j - \bar{T}]$, $j \in 1 \ldots m$.

Final model. The DynTex basis is the largest component of a model. However it compresses well using jpeg, and model storage sizes of 50kB-5MB are typical. (size is proportional to size and number of basisvectors.) The complete model of

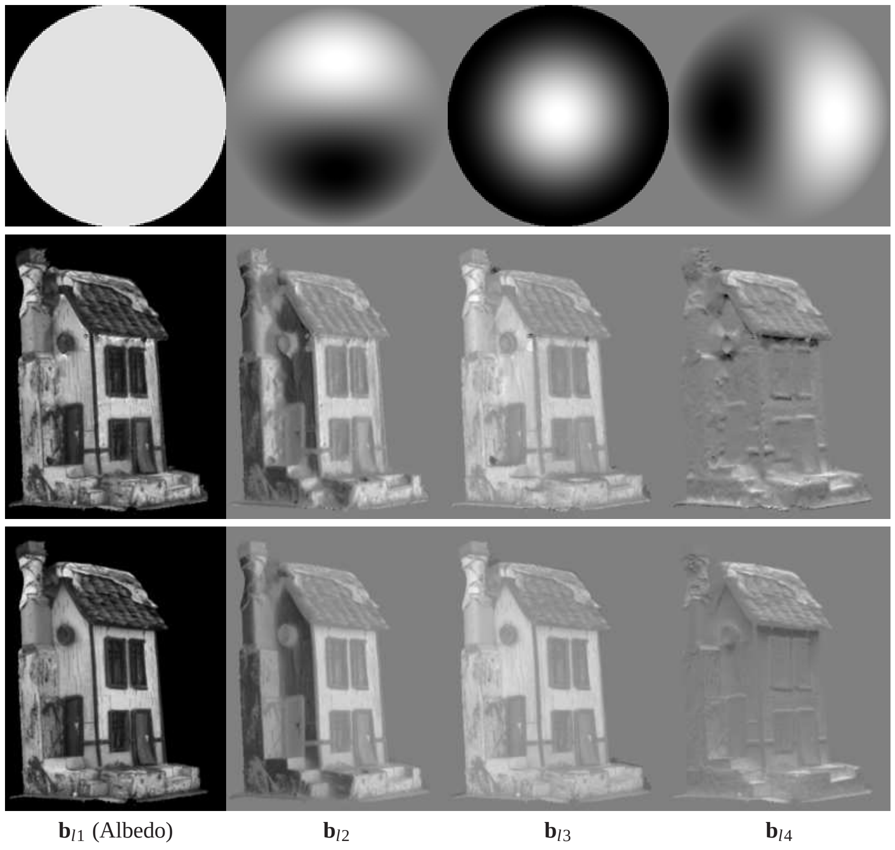
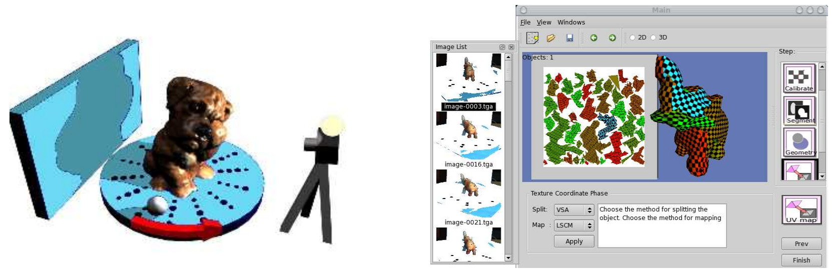


Fig. 5 Comparison of analytical form (Eq. 9) of $\mathbf{b}_l$ and the estimated PCA basis $\hat{\mathbf{b}}_l$ for light variability. Top row: Angular map of the spherical harmonics. Middle: Analytic spherical harmonic basis. Bottom: Corresponding light basis computed by PCA.

geometry and texture basis can be exported, either for inclusion in Maya or Blender, for which we have written a dynamic texture rendering plugin, or direct real-time rendering. An example of several object and people captured separately using our capture system and incorporated into a scene from Edmonton (location near Muttart conservatory) can be seen in using a cylindrical panorama of a city as backdrop, Fig. 8, Video 1[1].

Rendering. A desired view is given by the projection matrix P with the camera direction $\mathbf{v}$. For calculating the texture blending $\mathbf{y}$ we first apply 2-dimensional Delaunay triangulation over the camera viewing directions in the training set. Then we determine which simplex the new camera direction is contained in, and estimate the new texture modulation coefficients by linearly interpolating the coefficients associated with the corner points of the containing simplex. The new texture is generated from the basis textures and then the geometric model X is rendered into the desired



Texture Coordinate Phase

Split: VSA

Map : LSCM

Choose the method for splitting the object. Choose the method for mapping

Fig. 6 Left: experimental capture-setup. Right: GUI for our capture system. the screenshot shows the texture coordinate step.

pose. The most computationally demanding part of rendering is blending the texture basis. Hardware accelerated blending helps to achieve real time rendering. Depending on the graphics hardware capabilities one of several methods are choosen. On old, modest graphics cards, multipass rendering is used to blend the basis textures. On newer graphics hardware a shader program is used to directly blend the textures. If graphics hardware acceleration is unavailable, a SIMD MMX routine performs the texture blending. Rendering our textures on midrange HW, with shader programs, single objects render at well over 100Hz using 20 512×512 resolution basis textures per object. Ten dynamically textured objects in a scene still render at over 30Hz.

Example renderings. The first example illustrates the difference between a modulated texture and standard image texturing. A wreath made of natural straw and flowers was captured and processed into a texture basis. In Fig. 7 a rotating quadrilateral is textured with the image of the wreath. Using only one image, the texture appears unnatural from all but the capture direction as illustrated in the top row. On the other hand, modulating the view dependent texture, the fine scale variability from the wreath physical geometric texture as well as photometric properties is realistically reproduced (bottom row).

As mentioned, we are not limited to small objects. We can import geometries from 3Dwebservice[27]. In Video 3[1] and Fig. 9, we show a preachers chair captured in situ from the Seefeld church.

5 Experiments

Rendering quality of textures can be judged subjectively by viewers and evaluated numerically by comparing to ground truth images. Unlike comparisons of geometry alone, numeric errors are not indicative of perceptual quality. Furthermore, a static image does not show how light and specularities move. Therefore we rely mainly on the video renderings to argue photo-realistic results. As far as we know there is no commonly accepted standard for a perceptually relevant numerical measure.

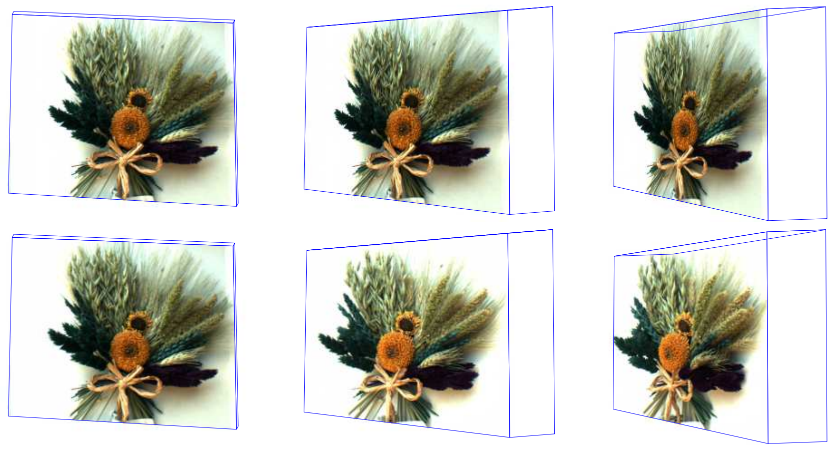


Fig. 7 Texturing a rotating quadrilateral with a wreath. Top: by warping a flat texture image. Bottom: by modulating the texture basis B and generating a continuously varying texture which is then warped onto the same quad.

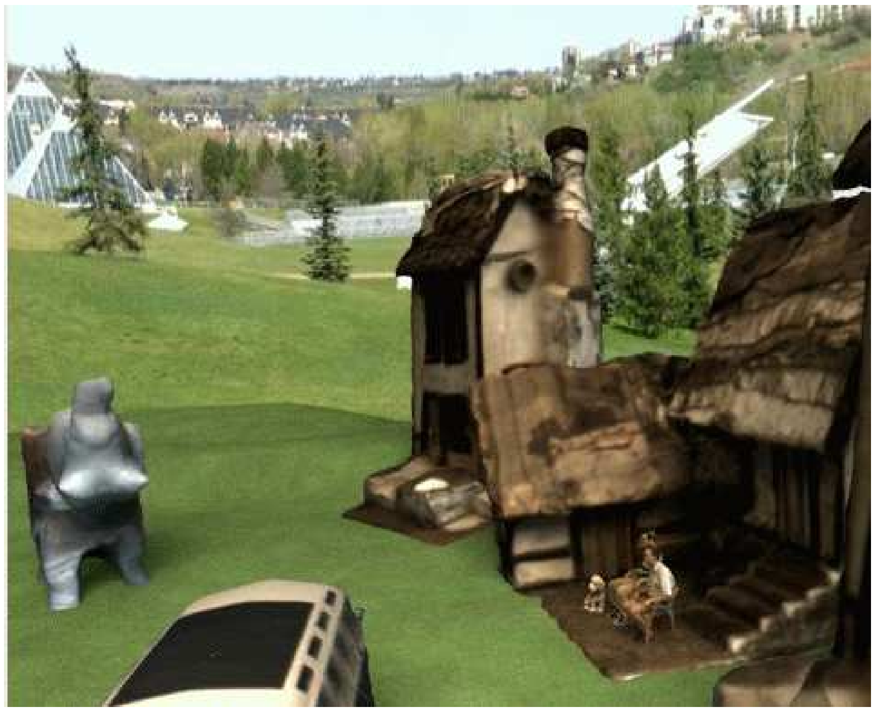


Fig. 8 Several objects and persons composed in Blender, **Video 1**

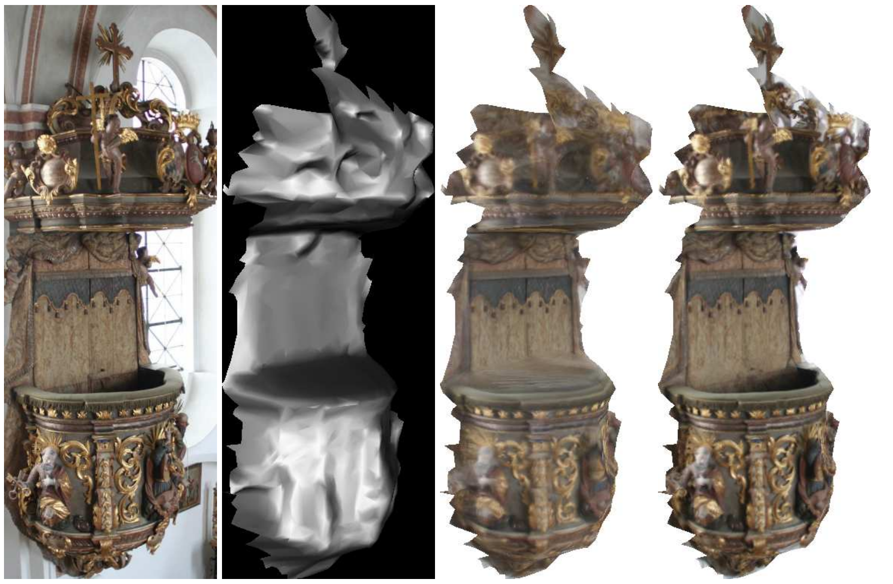


Fig. 9 Seefeld Kanzel: input image, geometry, static and dynamic texture rendering

We use just the mean pixel intensity difference between the rendered model and a real image (from a pose not used in the capture data to compute the model). For each experiment, a set of input images were acquired using the turntable setup. Half were used to compute the model, and the other half (from different viewpoints) were used as reference in the comparison videos and intensity error computation. For the three sequences below captured in our lab (house, elephant,and wreath) a PtGrey Scorpion camera at 800x600 resolution was used. Due to the calibration pattern taking up image space, the effective object texture resolution is however closer to web-cam VGA (640x480) resolution.

Four algorithms compared To evaluate the subspace-based dynamic texture we compared it to several other popular texturing methods in the literature. As a base case we use standard single texturing with the pixel values of the single texture computed to minimize the reprojection error in all training views. Next we choose the popular view dependent texturing (VDTM)[7], and our final comparison is against ray-based "Untructured Lumigraph" rendering[4]. While more methods have been published, many are variations or combinations of the four we compare. To put the methods on an equal footing we use 20 basis vectors in the dynamic texture. The VDTM texturing by blending textures sourced from 20 input images. In the lumigraph, for each texturespace pixel a list of ray color and u,v index is stored. The lumigraph is then computed on the geometry by picking the 20 rays per texture pixel that minimize the reprojection error over all input images. (Note: Unlike the VDTM and DynTex basis, jpeg compression does not work for the ray indices yielding in practice a larger data representation).

Selection of four test data sets. Depending on the complexity of the scene or object to be captured and rendered, different texturing methods can be used. In the following evaluation we choose four data sets of increasing complexity to challenge the texturing methods.

For a comparison to existing literature, we start with the downloadable temple scene from[23] (Fig. 10, I.) A close approximation to the true geometry is computed using SFS by our system, with 90% reconstructed within 1.7mm of ground truth, a further geometry refinement improves this to 1.1mm. Our geometry is not quite as good as Hernandez et al. (0.5mm)[10], but comparing texture renderings for the initial SFS model with those of the refined model, there is next to no perceptual difference. Likewise for this simple BRDF we find little perceptual or numerical error difference between using just a conventional static single texture or any of the view dependent textures (see Video 5[1]).

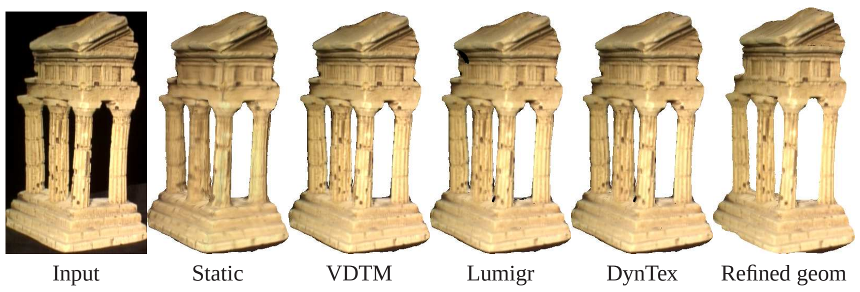


Fig. 10 Renderings of the temple from Middelbury multiview stereo image set. Texture is simple, and renders well with any texturing method.

Our second data set is of a house, with wood, bark and moss materials, and a more complex structure. For the house there a significantly difference between the SFS visual hull geometry and the underlying true geometry (particularly in the middle inside corner). Also can be seen in Video 6[1] and Fig. 11 now the static texture on compares badly to the view dependent ones, while there is little difference between.

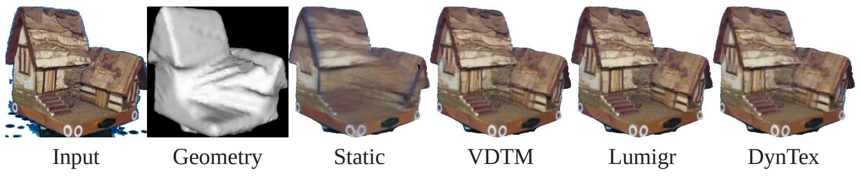


Fig. 11 Renderings of a textured SFS house model. The static texture is blurreddue to averaging colors on different rays, while the other textures are sharp with indistinguishable quality differences.

Third we try an elephant carved in jade. This object has a complex reflectance with both specularities and subsurface scattering. Here a single texture gives a dull flat appearance. VDTM is perceptually better, but a close analysis shows that some specularities are missing (e.g. on ears in Fig.12), and others have incorrect gradients. The DynTex and unstructured lumigraph show better results both visually and numerically for difficult (particularly specular) views, with a max intensity error of 6% compared to 10% for the standard view dependent texture and 19% for a static texture, Fig. 12 (Video 7[1]).

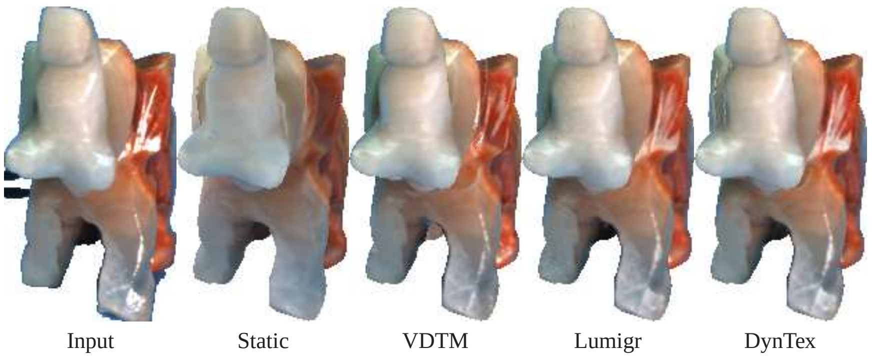


Fig. 12 A Jade elephant with complex reflectance. Static and VDTM textures are dull and completely miss the specularity on the ear. DynTex and Lumigraph capture the light and material more faithfully.

Finally, we show an example of a straw wreath, where obtaining a good geometry is very difficult (Fig. 13 IV, Video 8[1]). Here, a purely image-based method can represent a dense sample of the rayset, but at a huge storage (gigabytes) cost. We used a rough visual hull proxy geometry. The static texture is blurred out. The VDTM looks sharper because input images are used directly, but a close inspection shows somewhat jumpy transitions in the video, and during these transitions two input images are blended on top, creating a wreath with more straws. Both the DynTex and Unstructured Lumigraph code view dependency in texture space in different ways. These instead blur detail somewhat but give an overall lower error as explained below.

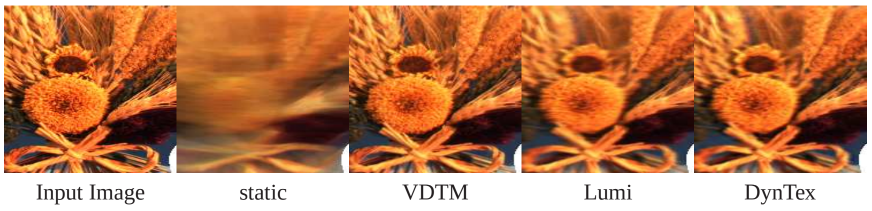


Fig. 13 Detail crops showing results for the Wreath with complex micro-geometry rendered on a rough proxy geometry.

error (variance)	temple	house	elephant	wreath
Static texture	10.8 (1.5)	11.8 (1.2)	19.0 (1.4)	28.4 (2.8)
VDTM	8.3 (1.9)	9.8 (1.3)	10.1 (1.9)	21.4 (3.5)
Lumigraph	10.8 (2.5)	9.8 (1.2)	5.9 (0.7)	14.3 (1.3)
DynTex	7.3 (1.0)	9.4 (1.0)	6.6 (0.7)	13.4 (1.2)

Table 1 Numerical texture intensity errors and (variance). %-scale.

Summarizing the experiments we find that for simple reflectance and geometry, any texturing method works well, while for more complex cases, view-dependent appearance modeling helps, and for the two most complex cases the DynTex has a better performance than VDTM. Both of these can be rendered in hardware using simple texture blending. The unstructured lumigraph has similar performance to the DynTex, but at a much higher storage cost, and would require a complicated plugin to render in Maya or Blender. The maximum image errors and error variance are summarized in Table1. The variance indicates smoothness of texture modulation over viewpoint changes. Perceptually a high value manifests itself as a jumpy appearance change. An example of viewpoint error variation can be seen in Fig. 14. The jumpy appearance of the VDTM is due to it working better when close to an image in the reference set.

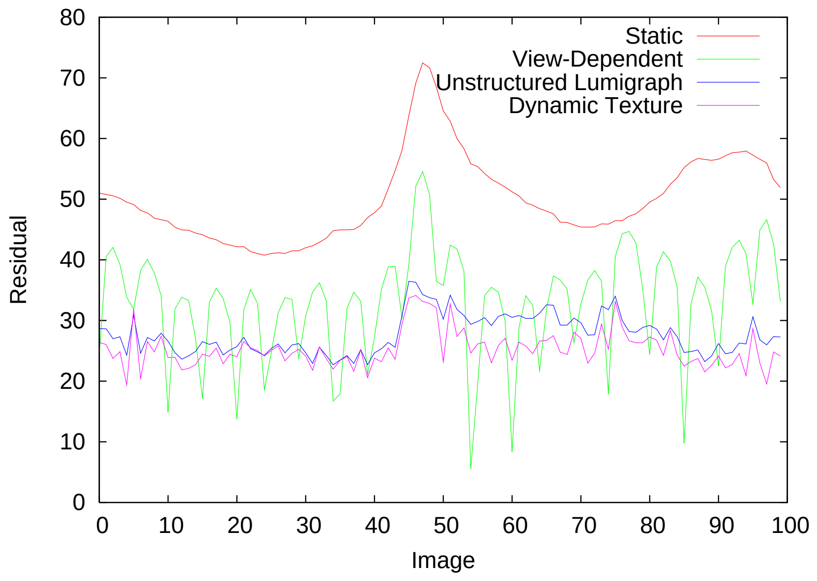


Fig. 14 Viewpoint variation of rendering error for the wreath

6 Discussion

We have presented a texturing method where for each new view a unique view-dependent texture is modulated from a texture basis. The basis is designed so that it encodes a texture intensity spatial derivatives with respect to warp and parallax parameters in a set of basis textures. In a rendered sequence the texture modulation plays a small movie on each model facet, which correctly represents the underlying true scene structure to a first order. This effectively compensates for small (up to a few pixels) geometric errors between the true scene structure and captured model.

The strength of our method lies in its ability to capture and render scenes with reasonable quality from images alone. Hence, neither a-priori models, expensive laser scanners nor tedious manual modeling is required. Only a PC computer and a camera is needed. This can potentially enable applications of modeling from images such as virtualized and augmented reality in the consumer market.

References

1. Movies of the experiments are available on http://www.cs.ualberta.ca/∼vis/scalesmod
2. Software is downloadable from http://www.cs.ualberta.ca/∼vis/ibmr
3. Barsi, R., Jacobs, D.: Lambertian refrectance and linear subspace. In: IEEE International Conference on Computer Vision, pp. 383–390 (2001)
4. Buehler, C., Bosse, M., McMillan, L., Gortler, S.J., Cohen, M.: Unstructured lumigraph rendering. In: Computer Graphics (SIGGRAPH), pp. 43–54 (2001)
5. Cobzas, D., Yerex, K., Jagersand, M.: Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. In: Eurographics (2002)
6. Dana, K.J., van Ginneken, B., Nayar, S.K., Koenderink, J.J.: Reflectance and texture of real-world surfaces. ACM Trans. Graph. **18**(1), 1–34 (1999)
7. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In: SIGGRAPH
8. Doretto, G., Chiuso, A., Wu, Y.N., Soatto, S.: Dynamic textures. Int. J. Comput. Vision **51**(2), 91–109 (2003)
9. Eisemann, M., Decker, B.D., Magnor, M., Bekaert, P., de Aguiar, E., Ahmed, N., Theobalt, C., Sellent, A.: Floating Textures. Computer Graphics Forum (Proc. Eurographics EG'08) **27**(2), 409–418 (2008)
10. Esteban, C.H., Schmitt, F.: Silhouette and stereo fusion for 3d object modeling. CVIU **96**(3), 367–392 (2004)
11. Freeman, W.T., Adelson, E.H., Heeger, D.J.: Motion without movement. In: SIGGRAPH
12. Furukawa, R., Kawasaki, H., Ikeuchi, K., Sakauchi, M.: Appearance based object modeling using texture database: acquisition, compression and rendering. In: Proceedings of the 13th Eurographics workshop on Rendering, pp. 257–266. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2002)
13. Gortler, S.J., Grzeszczuk, R., Szeliski, R.: The lumigraph. In: Computer Graphics (SIGGRAPH'96), pp. 43–54 (1996)
14. Hager, G.D., Belhumeur, P.N.: Efficient region tracking with parametric models of geometry and illumination. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(10), 1025–1039 (1998)
15. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2000)

16. Heckbert, P.: Fundamentals of Texture Mapping. Msc thesis. Technical Report No. UCB/CSD 89/516, University of California, Berkeley (1989)
17. Jagersand, M.: Image based view synthesis of articulated agents. In: CVPR (1997)
18. Levoy, M., Hanrahan, P.: Light field rendering. In: Computer Graphics (SIGGRAPH'96), pp. 31–42 (1996)
19. Levy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. ACM Trans. Graph. pp. 362 – 371 (2002)
20. Möller, T., Haines, E.: Real-time Rendering. A.K. Peterson (2002)
21. Oliveira, M.M., Bishop, G., McAllister, D.: Relief texture mapping. In: Computer Graphics (SIGGRAPH'00) (2000)
22. Ramamoorthi, R., Hanarahan, P.: On the relationship between radiance and irradiance: determining the illumination from images of a convex lambertian object. J. Optical Soc. Am. A **18**(10), 2448–2459 (2001)
23. Seitz, Curless, Diebel, Scharstein, Szeliski: A comparison of multiview stereo reconstruction algorithms. In: CVPR (2006)
24. Slabaugh, G.G., Culbertson, W.B., Malzbender, T., Schafer, R.W.: A survey of methods for volumetric scene reconstruction from photographs. In: International Workshop on Volume Graphics (2001)
25. Tarini, M., Callieri, M., Montani, C., Rocchini, C.: Marching intersections: an efficient approach to shape from silhouette. In: Proceedings of VMV 2002 (2002)
26. Vasilescu, M.A.O., Terzopoulos, D.: Tensortextures: multilinear image-based rendering. ACM Trans. Graph. **23**(3), 336–342 (2004)
27. Vergauwen, M., Gool, L.V.: Web-based 3d reconstruction service. Mach. Vision Appl. (17), 411–426 (2006)
28. Wood, D.N., Azuma, D.I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D.H., Stuetzle, W.: Surface light fields for 3d photography. In: Computer Graphics (SIGGRAPH'00) (2000)