Predictive Display Models for Tele-Manipulation from Uncalibrated Camera-capture of Scene Geometry and Appearance

Keith Yerex, Dana Cobzas and Martin Jagersand
Computing Science, University of Alberta, Canada
`www.cs.ualberta.ca/˜ {keith,dana,jag}`

***Abstract*— In tele-robotics delayed visual feedback to the human operator can degrade task performance significantly. To improve this, predictive display, uses a scene model to estimate and render immediate visual feedback based on the operator's control commands. Traditional predictive display involves the calibration and overlay of an a-priori model with the delayed real video feedback. In this paper we present an image-based method where the scene geometry and appearance is captured using structure-from-motion by an uncalibrated eye-in-hand camera mounted on the remote robot. The model is then compressed and transmitted to the operator site, where it is used to generate immediate feedback in response to the operators movements. Calibration problems are avoided since the model is captured by the same scene camera as is being simulated in the predictive display. We show experiments where we capture the appearance of a robot hand and transmit it over the network to the operator site where the model renders scene appearance change in response to operator viewpoint motion.**

I. Introduction

An important problem in tele-robotics is to provide high fidelity visual feedback from the remote scene to the human operator. However, limitations in terms of delays are inherent to transmission over a distance. Typical round-trip delays using e.g. NASA satellite links are in the order of seconds [11]. Using low cost alternatives such as the internet introduces additional unpredictability in the time delay and instantaneous bandwidth. It has been shown that as little as half a second delay causes operators to lose direct connection between their controlling movements and the visual display [8]. Instead of performing smooth motions, inefficient “move, then wait and see” strategies are adopted.

In predictive display the delay problem is eliminated by computer-synthesizing an estimated visual feedback *immediately* in response to the operator's movements. In most systems a CAD based line drawing of the object is rendered and overlaid on the delayed real video using an a-priori calibrated transform. For a survey, see [14]. However, important applications of tele-robotics are in non-engineered settings, where precise models of objects and environments are difficult to obtain (e.g. remote exploration, emergency response, waste cleanup, space robotics). In a recent system, predictive display for a mobile robot is provided by geometrically warping the delayed video using a computer vision line-based model[1]. While a mobile robot is typically controlled using 2 dimensional steering and

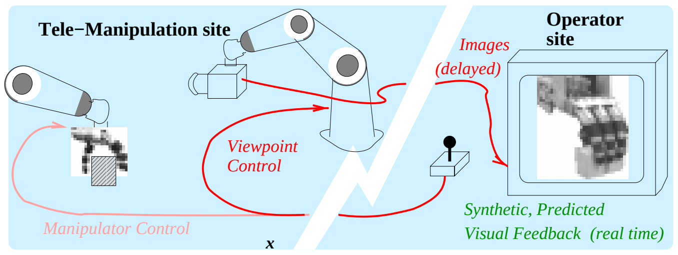

Fig. 1. Remote tele-manipulation setup.

velocity commands we develop vision-controlled robot arm and dexterous hand based tele-robotic systems[9], where the control space is from 6 to 20 dimensional, see Fig. 1. Our new technique accomodate larger viewpoint change than the case we solved before[10] by combining geometric and image-based techniques. Our main contributions include:

- We show how to merge previous work in geometry-based scene modeling[1], [15], [3], with image-based methods which generalize view-based textures from a discrete set of example textures to a continuous *dynamic texture* synthesis [2].
- We derive a new formulation of residual intensity variation capturing discrepancies between between the geometric model and the real scene and use this to compute the *dynamic texture* basis.
- We develop a HW accelerated OpenGL implementation wich allows cost-effective systems based on standard PC's and graphics cards, and in the experiments demonstrate real time model capture and predictive rendering on these.

Our method decomposes the problem of modeling the image change due to camera motion into two stages. In the first real-time visual tracking is used to capture a non-Euclidean model. The model represents an approximation of the true scene structure, and is used to approximately stabilize texture patches during tracking. Second, using image statistics, a spatial basis is constructed which captures the residual intensity variation in the stabilized texture. Both the geometry and texture basis are parameterized in pose to enable rendering of new poses. From this model, predicted images from new camera poses are rendered to the operator by warping the modulated texture to the projected geometry. The advantage of our approach is that it decomposes the difficult problem of exactly capturing geometry and aligning it with texture images into two simpler simpler tasks where the strengths and weaknesses of each of the two sub-methods complement each other.

II. Theory

Consider a tele-robotics setup as in Figure1, where an operator controls a remote robot. The remote scene is viewed by camera mounted on the robot. The scene images are shown to the operator using e.g. a video screen or head mounted display (HMD).

Let $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots)$ be a sequence of viewpoint motion commands[12] by the tele-operator. Assuming a round-trip delay d, the operator will not see the results of the current motion $\mathbf{x}_i$ until time $i + d$. We present a method where instead an estimated image $\hat{I}$ from scene viewpoint $\mathbf{x}_i$ is rendered immediately from an image-based model M. The model is generated using a sequence of (previous) images from the remote scene $(I_1, I_2 \ldots I_m)$ as training data.

1) Geometry and image-based model: In terms of geometry, image-based-rendering (IBR) techniques relate the pixel-wise correspondence between sample images $I_{1\ldots m}$ and a synthesized desired new view $\hat{I}$. This can be formulated using a warp function w_t to relate $I_t(w_t) = \hat{I}$. If $I_t \approx \hat{I}$ then w is close to the identity function. However, to relate arbitrary viewpoints, w can be quite complex, and current IBR methods generally require carefully calibrated cameras in order to have a precise geometric knowledge of the ray set [5], [12].

In our method the approximate texture image stabilization achieved using a coarse geometric model reduces the difficulty of applying IBR techniques. The residual intensity (texture) variability can then be coded as a linear combination of a set of spatial filters (Figure 2). More precisely, given a training sequence of images $I_t, t \in 1 \ldots m$ and tracked image points $[\mathbf{u}_t, \mathbf{v}_t]$, a simplified geometric structure P of the scene and a set of motion parameters $\mathbf{x}_t = (R_t, a_t, b_t)$ that uniquely characterize each frame is estimated from the tracked points using affine structure from motion (Section II-B). The re-projection of the structure given a set of motion parameters $\mathbf{x} = (R, a, b)$ is obtained by

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = RP + \begin{bmatrix} a \\ b \end{bmatrix} \tag{1}$$

From the training image I_t and the tracked points $[\mathbf{u}_t, \mathbf{v}_t]$, we flattern the triangulated image into a canonical texture I_{wt}

$$I_{wt} = I_t(w(\mathbf{u}_t, \mathbf{v}_t)) \tag{2}$$

Using the method described in Section II-A we then compute a variability basis B that captures the intensity variation caused by geometric approximations and illumination changes and the set of corresponding blending coefficients $\mathbf{y}_t$.

$$I_{wt} = B\mathbf{y}_t + \bar{I} \tag{3}$$

[1]We detail the parameterization of camera viewpoints in Section II-B

[2]We write scalars plain, vectors bold and matrices in upper case. Images flattened into column vectors are written bold upper case

2) Predictive view synthesis: To synthesize a new predicted view $\hat{I}$ we first estimate the projection of the structure P in the desired view (specified by motion parameters of the current operator viewpoint $\mathbf{x}$ in Equation 1). Then, texture modulation coefficients $\mathbf{y}$ are computed and a texture corresponding to the new view is blended (Equation 3). Finally the texture is warped to the projected structure (inverse of Equation 2). The following sections describe this process in detail.

A. Dynamic textures

The purpose of the *dynamic texture* is to allow for a non-geometric modeling and view-dependent synthesis of texture intensity variation during animation of a captured model. To motivate our development of a time varying texture, consider that using the standard optic flow constraints we can write translational image variability as the modulation of two basis functions: $I = I_0 + \frac{\partial I}{\partial u}\Delta u + \frac{\partial I}{\partial v}\Delta v$. Here the image derivatives $\frac{\partial I}{\partial u}$ and $\frac{\partial I}{\partial v}$ form a spatial basis.

Below we first extend this to 6 parameter warps representing texture transforms, with depth, non-rigidity and lighting compensation, and then we show how to stably estimate this basis from actual image variability.

1) Parameterizing image variability: Formally, consider an image stabilization problem. Under an image constancy assumption the light intensity from an object point $\mathbf{P}$ is independent of the viewing angle[6]. Let I_t be an image (patch) at time t, and I_w a stabilized (canonical) representation for the same image. In general, then there exists some coordinate remapping w s.t.

$$I_w(\mathbf{p}) = I(w(\mathbf{p}), t) \tag{4}$$

where $\mathbf{p}$ is the image plane projection of $\mathbf{P}$. Hence, I_w represents the image from some hypothetical viewing direction, and w is a function describing the rearrangement of the ray set from the current image I_t to I_w. In principle w could be found if accurate models are available for the scene, camera and their relative geometry.

In practice, at best an approximate function $\hat{w}$ can be found, which may be parameterized in time (e.g. in movie compression) or pose (e.g. in structure from motion and pose tracking). Below we develop mathematically the effects of this approximation. In particular we, study the residual image variability introduced by the imperfect stabilization achieved by $\hat{w}$, $\Delta I_t = I_t(\hat{w}) - I_w$. Let $\hat{w} = w + \Delta w$ and rewrite as an approximate image variability to the first order (dropping t):

$$\Delta I = I(w + \Delta w) - I_w = I(f) + \frac{\partial}{\partial w} I(f)\Delta w - I_w = \frac{\partial}{\partial w} I(w)\Delta w \tag{5}$$

The above equation expresses an optic flow type constraint in an abstract formulation without committing to a particular form or parameterization of w. In practice, the function 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.

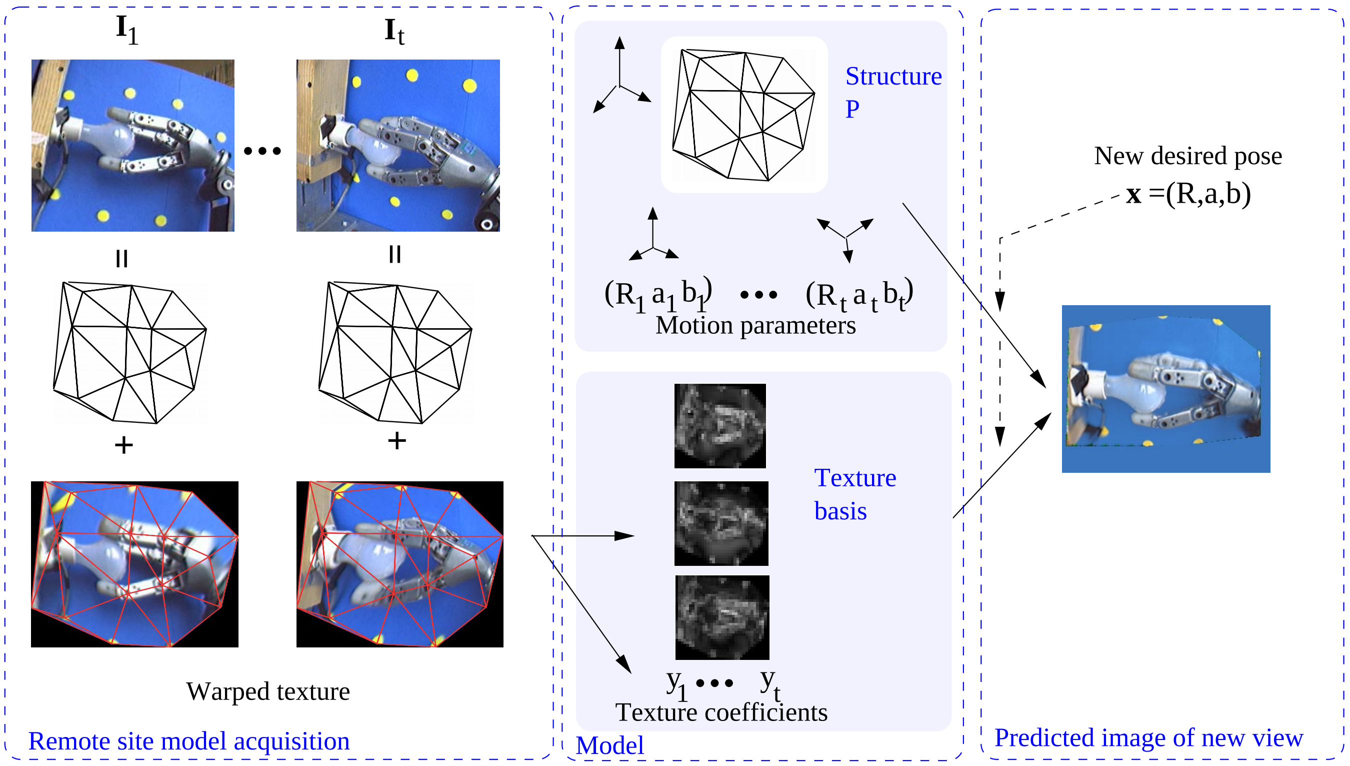

Fig. 2. A sequence of training images $I_1 \cdots I_t$ is decomposed into geometric shape information and dynamic texture for a set of triangular patches. The scene structure P and motion (r, s, a, b) is determined from the projection of the structure using a factorization algorithm. The dynamic texture is decomposed into its projection $\mathbf{y}$ on an estimated basis B. For a given desired position, a novel image$\hat{I}$ is predicted by warping new texture synthesized from the basis B on the projected structure.

2) Structural image variability: Under a weak perspective (or orthographic) camera geometry, plane-to-plane transforms are expressed using an affine transform of the form:

$$\begin{bmatrix} u_w \\ v_w \end{bmatrix} = w_a(\mathbf{p}, \mathbf{a}) = \begin{bmatrix} a_3 & a_4 \\ a_5 & a_6 \end{bmatrix} \mathbf{p} + \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \qquad (6)$$

This is also the standard image-to-image warp supported in OpenGL. Now we can rewrite the image variability Eq. 5 resulting from variations in the six affine warp parameters as:

$$\Delta I_a = \sum_{i=1}^{6} \frac{\partial}{\partial a_i} I_w \Delta a_i = \left[\frac{\partial I}{\partial u}, \frac{\partial I}{\partial v}\right] \begin{bmatrix} \frac{\partial u}{\partial a_1} \cdots \frac{\partial u}{\partial a_6} \\ \frac{\partial v}{\partial a_1} \cdots \frac{\partial v}{\partial a_6} \end{bmatrix} \Delta[a_1 \ldots a_6]^T \qquad (7)$$

Let $\{I\}_{discr} = \mathbf{I}$ be a discretized image flattened along the column into a vector, and let '$*\mathbf{u}$' and '$*\mathbf{v}$' indicate point-wise multiplication with column flattened camera coordinate u and v index vectors. Calculating the partial derivatives (Jacobian) of Eq. 6 we get a closed for Eq. 7 as:

$$\Delta \mathbf{I}_a = \left[\frac{\partial \mathbf{I}}{\partial u}, \frac{\partial \mathbf{I}}{\partial v}\right] \begin{bmatrix} 1 & 0 & *\mathbf{u} & 0 & *\mathbf{v} & 0 \\ 0 & 1 & 0 & *\mathbf{u} & 0 & *\mathbf{v} \end{bmatrix} [y_1, \ldots, y_6]^T = [\mathbf{B}_1 \ldots \mathbf{B}_6][y_1, \ldots, y_6]^T = B_a \mathbf{y}_a \qquad (8)$$

where $[\mathbf{B}_1 \ldots \mathbf{B}_6]$ can be interpreted as a variability basis for the affine transform.

Similarly, a two dimensional spatial basis $\mathbf{B}_d$ can be derived to account (to a first order) for parallax due to scene depth variation when model triangles do not perfectly correspond to physical planes. It has also been shown that a low dimensional intensity subspace $\mathbf{B}_l$ (5-9 dimensions) is sufficient for representing the light variation of most natural scenes [13], [6].

Statistical image variability In a real, imperfectly stabilized image sequence we can expect all of the above types of image variation, as well as unmodeled effects and noise $\Delta \mathbf{I}_e$. Hence, total residual image variability can be written as:

$$\Delta \mathbf{I} = \Delta \mathbf{I}_a + \Delta \mathbf{I}_d + \Delta \mathbf{I}_l + \Delta \mathbf{I}_e = B_a \mathbf{y}_a + B_d \mathbf{y}_d + B_l \mathbf{y}_l + \Delta \mathbf{I}_e = \mathbf{B}\mathbf{y} + \Delta \mathbf{I}_e \qquad (9)$$

In principle B_a can be derived from images alone using Eq. 8, while B_d requires dense and accurate object depth information. In practice, we instead estimate both from actual statistics in the training sequence using PCA. This yields a transformed basis $\hat{B}$ and set of modulation vectors for each training image $\mathbf{y}_k$ which captures (up to a linear coordinate transform) the actually occurring image variation in the true B (Eq. 9).

To estimate the texture mixing coefficients for intermediate poses, we first apply n-dimensional Delaunay triangulation over the sampled poses $\mathbf{x}_t$. Then given any new pose $\mathbf{x}$ we determine which simplex the new pose is contained in, and estimate the new texture mixing coefficients $\hat{\mathbf{y}}$ by linearly interpolating the mixing coefficients of the corner points of the containing simplex. Finally, a novel texture image is generated by adding the dynamic texture variation to the reference (mean) image $I_w = \bar{I} + B\hat{\mathbf{y}}$.

B. Geometric model

A structure-from-motion algorithm starts with a set of corresponding features (point, lines) in a sequence of images of a scene and recovers the coordinates of these features and the cameras poses relative to this representation (see Figure 3). Different algorithm have been developed depending on the camera model [4]. If the camera is approximated with an affine camera, the problem can be linearized and solved using factorization [16]. Here we extended the classical Tomasi-Kanade factorization to a weak perspective camera inspired by [17].

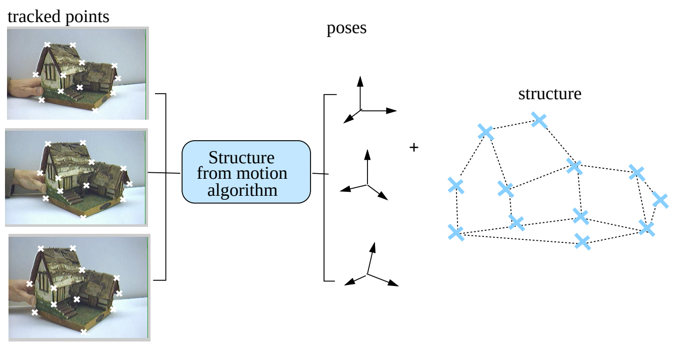

Fig. 3. A general structure from motion algorithm extracts the structure and camera poses from a set of tracked points.

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. We next describe a more detailed mathematical formulation of the problem.

Affine structure from motion 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:

$$\begin{aligned} u_{ti} &= s_t \mathbf{i}_t^T \mathbf{P}_i + a_t \\ v_{ti} &= s_t \mathbf{j}_t^T \mathbf{P}_i + b_t \end{aligned} \tag{10}$$

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 $\mathbf{t}1_t$ of the translation $\mathbf{t}_t$. Rewriting Eq. 10 for multiple points (n) tracked in several frames (m)

$$W = RP + \mathbf{t}1 \tag{11}$$

where W is a $2m \times n$ matrix contains image measurements, R represents both scaling and rotation, P is the shape and $\mathbf{t}1$ is the translation in the image plane [17].

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:

$$\hat{W} = RP \ \ where \ \ \hat{W} = W - \mathbf{t}1 \tag{12}$$

Following [16], in the absence of noise we have $\mathrm{rank}(\hat{W}) = 3$. Under most viewing conditions with a real camera the effective rank is 3. Considering the singular value decomposition of $\hat{W} = O_1 \Sigma O_2$ we form

$$\hat{R} = O_1', \qquad \hat{P} = \Sigma' O_2' \tag{13}$$

where O_1', Σ', O_2' are respectively defined by the first three columns of O_1, the first 3×3 matrix of Σ and 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:

$$R = \hat{R}Q \qquad P = Q^{-1}\hat{P} \tag{14}$$

Q can be determined by imposing constraints on the components of the scaled rotation matrix R:

$$\begin{aligned} &\hat{\mathbf{i}}_t^T QQ^T \hat{\mathbf{i}}_t = \hat{\mathbf{j}}_t^T QQ^T \hat{\mathbf{j}}_t && (= s_t^2) \\ &\hat{\mathbf{i}}_t^T QQ^T \hat{\mathbf{j}}_t = 0 && t \in \{1..m\} \end{aligned} \tag{15}$$

where $\hat{R} = [\hat{\mathbf{i}}_1 \cdots \hat{\mathbf{i}}_m \hat{\mathbf{j}}_1 \cdots \hat{\mathbf{j}}_m]^T$ This generalizes [16] from an orthographic to a weak perspective case.

To extract pose information for each frame, we estimate the scale factor s_t and rotation components $\mathbf{i}_t$ and $\mathbf{j}_t$ from R. We estimate the 3×3 full rotation matrix by completing the last row with $\mathbf{k}_t = \mathbf{i}_t \times \mathbf{j}_t$ and parametrize it with Euler angles $\mathbf{r}_t = [\psi_t, \theta_t, \varphi_t]$. Each camera pose is represented by the motion parameter vector $\mathbf{x}_t = [\mathbf{r}_t, s_t, a_t, b_t]$. The geometric structure is represented by P and its reprojection given a new pose $\mathbf{x} = [\mathbf{r}, s, a, b]$ is estimated by

$$[\mathbf{u}, \mathbf{v}] = sR(\mathbf{r})P + \left[\begin{array}{c} a \\ b \end{array} \right] \tag{16}$$

where $R(\mathbf{r})$ represents the rotation matrix given the Euler angles $\mathbf{r}$.

III. Implementation

We have implemented and tested our method on consumer grade PC's for video capture, tracking and rendering. The operator views the remote scene on a monitor, and uses the mouse to change the viewpoint and mouse buttons to switch between rotation, translation and scale (zoom) mode. We mounted the camera to a pan-tilt head using a long swing arm and aiming the camera inwards. By placing the pan-tilt center of rotation near the scene to be imaged the operator can control the viewpoint to sample the viewing sphere. This limits the viewpoints in the sample sequence to a 2D manifold, but by the re-projection property of the geometric model P we can from the model synthesize any 6D affine camera pose by varying the scaled rotation sR and translation $[a, b]$.

A. Model capture algorithm

We have implemented the remote-site tracking and capture part of our system using XVision[7]. To compute a model we process about 128-512 images of the remote site video under varying camera pose as follows:

1) The operator selects the scene region to be captured in the model by clicking on 10-20 (trackable) image points $[\mathbf{u}_1, \mathbf{v}_1]$ in image I_1. Delaunay triangulation is used to divide the region into texture patches. For subsequent video images I_t point correspondences $[\mathbf{u}_t, \mathbf{v}_t]$ are obtained from real-time SSD tracking.
2) Using HW accelerated OpenGL each frame I_t is loaded into texture memory and warped to a standard shape I_{wt} based on tracked positions. (Equations 2). The standard texture shape is chosen to be the average positions of the tracked points scaled to fit in a square region as shown in Fig. 2.
3) We estimate the geometric model P from the tracked as described in Section II-B and a texture basis B as in Section II-A

The PCA is performed separately on each color channel in YUV color space. By separating the intensity information from the color, more eigenvectors can be used for the intensity channel, and less for the color channels, saving both texture memory and rendering time. Typically the eigenvector cutoff k is chosen so the resulting composite model is about one fifth the size of the set of sample images. Hence, if bandwidth is limited it is most efficient to compute the model at the remote site, and then transfer it to the operator console.

B. Hardware Rendering

To render dynamic textures in real-time, we use NVidia graphics cards, and NVidia specific OpenGL extensions for texture blending (NV_register_combiners, NV_register_combiners2) as follows:

1) For the new desired view compute the re-projection $[u, v]$ from the pose $\mathbf{x}$ as in Equation 1.
2) Estimate texture blending coefficients $\mathbf{y}$ by interpolating the coefficients of the nearest neighbors from the coefficients, and poses from the training data.
3) Compute the new textures in the standard shape using Equation 3 and warp the textures onto the calculated geometry.

These operations are performed simultaneously in graphics hardware as follows. Basis images are stored in texture memory as 4 channel (RGBA) textures in signed 8 bit format. Hence each texture holds 4 basis images, all 4 belonging to the basis for the same channel (Y, U or V). Since the hardware we are using can access 2 (GeForce 2 card) or 4 (Geforce 3 and 4) textures per rendering pass, up to 16 basis images (or 15 and the mean image) can be combined in a single pass. Using the register combiners mechanism, the 16 basis images are multiplied by their 16 corresponding coefficients, and summed. The output is then converted to RGB by multiplying the scalar result by the row in a YUV to RGB transformation matrix corresponding to the particular channel being rendered in the current pass. Each pass is then accumulated in the frame buffer using OpenGL blending. Finally, the dynamically generated texture is stored in texture memory, and rendered onto the object geometry.

Using the HW accelerated implementation we can update the predicted image at around 50 Hz using a 1year old GeForce 3 desktop graphics card, and at about 25-30Hz running on an basic GeForce 2-to-go in a 1GHz laptop.

IV. Experimental results

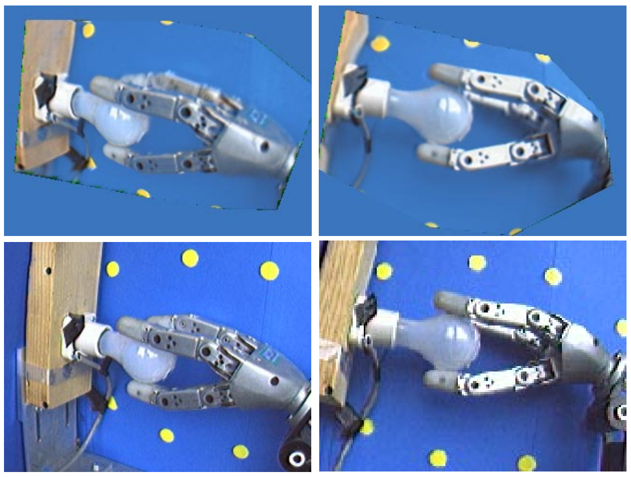

Fig. 4. Image sequence predicted for different viewpoints (top) and ground truth from corresponding (delayed) real images taken from the same viewpoints (bottom). See `www.cs.ualberta.ca/~ jag/videos/m1.mpg` for the full video

To test our method, we have used it to capture a robot hand. The hand has fairly complex geometry which would be difficult to capture completely. Some parts of the object are occluded from some angles, and therefore would be very hard to track. Using our technique, a very simple geometric model (Fig. 2) can be used including only easily trackable points, and the dynamic texturing compensates for the geometric inaccuracy (note the partially occluded finger in the back is represented mostly by the dynamic texture, and not by geometry).

To quantitatively evaluate rendering fidelity we measured geometric and intensity errors obtained with our dynamic texture and compared these with the view dependent textures used in other models from images work (e.g. in [3]). In view dependent texturing several texture images are used, and in rendering the one or a blend of several real images close to the virtual camera viewpoint are chosen to texture the model. To put the methods on equal footing each one was allowed the same amount of texture memory,

To measure appearance error we computed the differences in pixel intensities between rendered images and ground truth real images of the same scene and viewpoint, see Fig. 5. For the dynamic texture we got a smooth and low error of 0.56% (or less than 2 units on a standard 8bit/color channel quantization). The view dependent texture exhibits larger errors the further away the virtual viewpoint is from the real sample images. On average the error was 1.17%, but the high peak errors gave a visually unpleasant jumpiness in response to operator viewpoint motion.

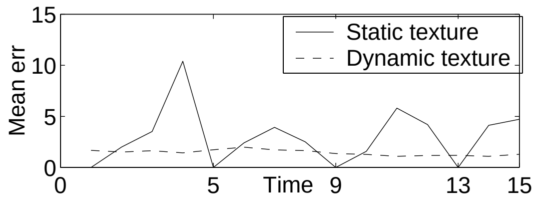

Fig. 5. Intensity pixel error in the rendered images (compared to original image)

In animation the reproduction of continuous realistic motion over a sequence of frames is as important as image sharpness. To quantify the jumpiness we captured a model from a standard camera calibration pattern, and re-animated a smooth, known viewpoint motion For each feature on the texture image we measured the actual reprojection and compared it to ground truth. The following table shows the average pixel jitter.

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

V. Discussion

We showed how to capture object models from a remote moving camera and render predicted images from new views using a new type of image-based modeling where a coarse geometric model is captured from images, and a time-varying *dynamic texture* is overlaid to compensate for errors in the coarse geometry approximation. Our technique obliviates the need for expensive range sensors and calibrated setups to capture the remote scene geometry, and instead uses inexpensive consumer web or video cameras with a standard PC's. To enable real-time rendering we take advantage of recent advances in consumer grade graphics cards which allow blending of several transparent textures.

In the future we plan to integrate the predictive display with our visual feedback control system[9]. This would enable vision-based tasks like the light bulb replacement in `www.cs.ualberta.ca/~ jag/videos/m2.mpg` to be performed in a tele-robotics setting where the operator receives real-time photo-realistic predictive display in response to simultaneous viewpoint and arm/hand motions. In the current implementation the operator has to interleave separate viewpoint selection and arm/hand motion sequences. The main obstacle in our current implementation is the the texture basis mixture coefficients are indexed in global pose. To represent independent motion of the robot hand/arm we plan to change this to modulate the *dynamic texture* based on the local triangle configuration. We are also working on providing a 3D impression using CAVE and HMD stereo displays. Generating the left and right images is simple from our model by just changing the viewing pose by the interocular distance. It remains to integrate our system with real-time pose tracking of also the operator to be able to put the predictive display into the U of Alberta 3D immersive CAVE.

References

[1] M. Barth, T. Burkert, C. Eberst, N. Stöffler, and G. Färber. Photo-realistic scene prediction of partially unknown environments for the compensation of time delays in presence applications. In *Int. Conf. on Robotics and Automation*, 2000.
[2] D. Cobzas and M. Jagersand. Tracking and rendering using dynamic textures on geometric structure from motion. In *ECCV*, 2002.
[3] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from phtographs. In *Computer Graphics (SIGGRAPH'96)*, 1996.
[4] O. D. Faugeras. *Three Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, Boston, 1993.
[5] S. J. Gortler, R. Grzeszczuk, and R. Szeliski. The lumigraph. In *Computer Graphics (SIGGRAPH'96)*, pages 43–54, 1996.
[6] 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.
[7] G. D. Hager and K. Toyama. X vision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding*, 69(1):23–37, 1998.
[8] R. Held, A. Efstathiou, and M. Greene. Adaption to displaced and delayed visual feedback from the hand. *J. Exp Psych*, 72:871–891, 1966.
[9] M. Jagersand. Image based visual simulation and tele-assisted robot control. In *IROS Workshop: New Trends in Image Based Visual Servoing*, 1997.
[10] M. Jagersand. Image based predictive display for tele-manipulation. In *Int. Conf. on Robotics and Automation*, 1999.
[11] W. S. Kim and A. K. Bejczy. 'demonstration of a high-fidelity predictive/preview display technique for telerobotic servicing. *IEEE Transactions on Robotics and Automation*, 1993.
[12] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics (SIGGRAPH'96)*, pages 31–42, 1996.
[13] A. Shashua. *Geometry and Photometry in 3D Visual Recognition*. PhD thesis, MIT, 1993.
[14] T. B. Sheridan. Space teleoperation through time delay: Review and prognisis. *IEEE Tr. Robotics and Automation*, 9, 1993.
[15] I. Stamos and P. K. Allen. Integration of range and image sensing for photorealistic 3d modeling. In *ICRA*, 2000.
[16] 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.
[17] 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.