Predictive Display System for Tele-manipulation using Image-Based Modeling and Rendering

Zhenyuan Deng and Martin Jägersand
Dept. Computing Science, University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
Email: {zdeng, jag}@cs.ualberta.ca

***Abstract*— Using a combination of techniques from visual tracking, image-based rendering, distributed programming, and robot motion control, we present a predictive display system to improve local operator's performance in tele-manipulations. More precisely, we construct a networked distributed system allowing the robot motion control and predictive display function to be implemented in a single PVM (Parallel Virtual Machine) program running on both the operator and remote scene computer. We also integrate our system with real-time pose tracking of the operator to allow 3D rendering in HMD.**

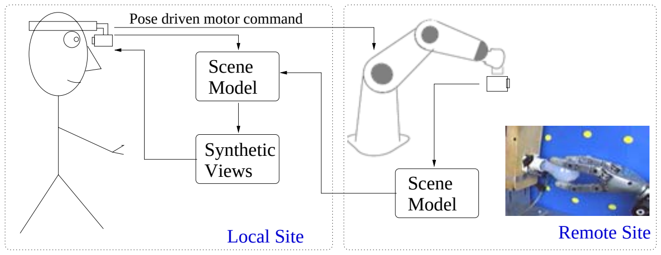

Fig. 1. Gaze Control Loop

I. Introduction

In tele-operation tasks such as space robotics, toxic or radioactive cleanup, human operators observe a remote scene through cameras, while tele-manipulating a robot. The operator response is based on the latest feedback images from the cameras (see Figure.1). A main challenge is that typical network and switching delays in the feedback path start in the order of seconds and increase with transmission distance. As a basis for comparison, the psychophysics literature indicates that delays as short as 0.5 second can impair the performance of an operator involved in a teleoperation task. In fact, the visuomotor tracking capabilities of humans and primates have often been compared to an asynchronous system with intermittent sample-data control [1]. Furthermore, in [2], the authors mention that human tracking patterns are similar to intermittent sample and hold patterns for which the rate of corrections is controlled by the delays of the visual feedback. Hence, it would appear that longer delays decrease the motor correction rate. Also, other factors such as limited bandwidth contribute to low-resolution and/or low frame rate video streams and impair human performances even more. [3]

Predictive display has been defined as using a computer for extrapolating displays forward in time [4]. This definition could well be generalized to generating a display without necessarily relying on time extrapolation or delayed feedback. With this in mind, a local model of the remote scene can be used to predict and render the remote scene in response to operator motor commands. It replaces the delayed video feedback with immediate synthesized images and enables local operators to perform operations normally. In most predictive display systems a CAD-based line drawing of the scene and objects is rendered and overlaid on the delayed video using an *apriori* calibrated transform [4]. However, many applications in tele-robotics are in non-engineered environments, where precise 3D models of objects and scenes are difficult to obtain.

Capitalizing on recent developments in both computer vision and computer graphics it is now possible to develop visual models of a robotic work-site without needing a-priori geometry [5]. Here we extend this work by a distributed systems implementation and evaluation. First an image-based model of the remote scene is captured by moving (eye-in-hand) camera. The model is then transmitted to the operator site and used to generate various viewpoints of the scene in response to operator head movements using image-based rendering(IBR). This avoids both time-consuming modeling and cumbersome calibration of the model-camera alignment. Particular features of our system are:

- A combination of real-time visual tracking, structure-from-motion geometric modeling and dynamic textures allows models to be captured by the remote scene camera without the need for special calibration or cumbersome manual modeling.
- Implementation of a networked distributed system allowing the robot motion control and model acquisition functions to be implemented in a single Parallel Virtual Machine(PVM) [6] program running on both the operator and remote scene computer.
- Integration and implementation of an user interface in which the remote robot motion and scene rendering are parameterized by the pose of the operator (orientation of the head).
- All parts of the system are commodity (consumer level) hardware, and the PVM allows execution over both intra-nets and the widely available Internet, potentially allowing low cost tele-operation and inspection between almost any two sites.

The rest of the paper is organized as follows. Section II briefly reviews the functionalities of predictive display system. Section III presents the system components in details. Section IV shows experimental results. Concluding remarks are drawn in Section V.

II. System Overview

Consider a tele-robotics setup (see Figure.1) 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 $(\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_i, \ldots)$ be a sequence of viewpoint motion commands by the tele-operator. Assuming a round-trip delay d, the operator will not see the results of the current motion $\mathbf{m}_i$ until time $i + d$.

Combining techniques from visual tracking, view synthesis, networked programming, and robot motion control, we present a predictive display system which by removing much of the time delay improves local operator's performance in tele-manipulation. While controlling the robot remotely, instead of seeing the real video image, the operator observes an estimated image $\hat{I}$ from scene viewpoint $\mathbf{x}_i$ 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. To support realistic high fidelity rendering we replace the standard texture image with a time varying dynamic texture [7] which like in mpeg movie compression compensates for errors in the motion prediction.

Networked Program

A predictive display system consists of several individual software components mapping onto computers at both the operator and robot site. Efficient communication has been studied in the field of distributed programming. PVM [6] is a software system that allows the utilization of heterogeneous network of parallel and serial computers as a single computational resource. We integrate the components via PVM as one single networked program operating on both the operator site and remote site. Briefly, we go over the components in the following:

A. Operator Site

In the operator end, after the operator sends each robot motor command to the remote robot, the scene renders a synthetic view immediately for the operator. The system consists these components: a head tracker for head pose acquisition, a scene model for rendering synthetic view.

- Head Tracker
  Being able to sense the human operator's movement is of great interest in the field of tele-manipulation. In our setup, when the operator moves (e.g. rotates) his/her head the motion is sensed by a head tracker. The head tracker continuously provides the 3D (orientations) parameters of the operator's head via PVM to both the local scene rendering and remote robot site.
- Scene Rendering
  As the head tracker updates the operator's head pose through PVM broadcast, the scene model renders the view with respect to the operator's head, and shows it to operator immediately.

B. Remote Site

While in the remote end, the networked system consists the following components: a robot control process, a eye-in-hand camera mounted on the robotic arm for real scene capturing and a visual tracking process for scene modeling. The scene model can be constructed off-line or on-line at the remote site, and then compressed and transmitted to the operator site. In our system, we adapt the dynamic texture rendering model by Cobzas et al [7]. The model merges a coarse geometry-based modeling and image-based methods which generalized view-based textures from a discrete set of example textures to a continuous texture synthesis. The scene geometry and appearance is captured using structure-from-motion (SFM) [8] by an uncalibrated eye-in-hand camera mounted on the remote robot.

- Robot/Camera motion control process
  With a eye-in-hand camera mounted on it, the robot is controlled remotely via PVM to replicate the operator's viewpoint changes. Through PVM, the robot receives the motor command, and moves accordingly, which updates the camera's view point so that the camera watches the scene from the same view point as the operator in the other end.
- Visual Tracking process
  As described above, the scene geometry and appearance is captured using SFM by an uncalibrated eye-in-hand camera. Visual tracking is needed for feature points correspondences in SFM. Through PVM, the operator drives the robot to different joint configurations, while the tracking records the image coordinates of the features points. Extended from Cobzas et al [7], we index the views by the robot joint configuration as well , which enables us to recover the joint motor command given view parameters of the scene.

III. System Parts

As pointed out earlier, our method has two stages. The first stage is run offline and automatically captures a model of the remote scene. The second stage is run on-line and uses the captured model to synchronously render predicted views of the scene in response to operator motion.

The following sections will detail the system parts and functions of our system.

A. Video

Until recently transmitting video over a distance required expensive high bandwidth point-to-point connections. Recently, with improvements to Internet capacity, and the availability of high speed internet connections in most parts of the country, developing inexpensive systems based on commodity PC hardware and networking has become a possibility. Consumer applications of this include video-phoning and teleconferencing software such as "virtual meeting" and "gnome meeting".

Our system uses the same type of hardware for the video acquisition. Recent consumer web cams based on the IEEE 1394 camera communication standard on a 400Mb/s digital network allows real-time (30Hz) capture of reasonable quality digital video. We have successfully used Pyro1394 and iBot web cams (each about US$100). The IEEE 1394 network has a maximum cable length of 4.5m and hence the uncompressed video has to be processed at the remote site. The remote computer is a standard 1.3GHz consumer PC running Linux. Using public domain software libraries for IEEE 1394 network and camera drivers and the XVision2 [9] real-time tracking routines, we implement a video pipeline and visual tracking on the remote computer.

B. Tracking

The visual tracking processes the real-time video and in each frame f calculates[1] the image plane projection $\mathbf{y}_f = (u_{n,f}, v_{n,f})^T$ of a set of n fiducial points. Briefly, the tracking is a dynamic system implementation of an SSD patch tracker [10]. Given a small reference image patch E, the current image frame I_f and an estimate of its approximate location, here we use where it was last seen, $\mathbf{y}_{f-1}$. the patch new position is calculated by solving for a correction $\Delta\mathbf{y}$ and updating the current state $\mathbf{y}_f$:

$$\mathbf{I}_f(\mathbf{y}) - \mathbf{E} = -\left(\frac{\partial \mathbf{T}}{\partial u}, \frac{\partial \mathbf{T}}{\partial v}\right)\Delta\mathbf{y}$$

$$\mathbf{y}_f = \mathbf{y}_{f-1} + \Delta\mathbf{y}$$

Here $\mathbf{I}_f(\mathbf{y})$ is the sub-image corresponding to E cut out from the current frame $\mathbf{I}_f$ at coordinates $\mathbf{y}$ and stacked into a column vector. The above update uses a linearized model of image variability and is iterated two to three times for each frame for convergence.

Current PC's can track a set of small patches in real time, but are not fast enough process the whole frame tiled into patches at 30Hz. Practically instead, the user selects a set of salient features by marking several fiducial points in the initial (static) video image of a scene. The points are placed to capture the geometric variation in the scene, and the convex hull of the points define the area of interest to be encoded in the visual model.

C. Visual Model

To capture and build a visual model of the remote scene, the video images and corresponding tracked image points are used as input to a modeling algorithm. The modeling algorithm builds a geometric 3D structure of scene points $X = [\mathbf{x}_i],\ i \in 1 \ldots n$ from the image coordinates $[\mathbf{y}_{i,f}]$ of the set of tracked points over several video frames $f \in 1 \ldots m$ using a structure-from-motion algorithm. In the past ten years several such algorithms have been published for both non-linear perspective and linear affine camera models [11]. For performance reasons we choose to implement a linear factorization algorithm based on a weak perspective camera [8]. Under this camera model, the input image points and structure can be related in each video frame through the weak perspective camera equation:

$$\mathbf{y}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = sR(\mathbf{r})\mathbf{x}_i + \begin{bmatrix} a \\ b \end{bmatrix}$$

where $R(\mathbf{r})$ is the camera-scene rotation matrix parameterized in Euler angles $\mathbf{r}$, $[a,b]^T$ is the translation and s scale.

In conventional graphics textures are represented as an image on the triangulated surface of the 3D structure X. This assumes that the geometry accurately represents the true underlying scene. In image-based modeling and rendering it has proven hard to automatically extract and align a scene geometry with video [12]. Here we instead model appearance with a time-varying *dynamic texture* [13], [14]. During the model capture and tracking, key frames from which to code the texture variation are picked at about 5Hz. For each key frame the projection of the geometric structure $\mathbf{y}_{i,f}$ is used to warp the image $\mathbf{I}_f$ into a texture $T_{w,f}$ with canonical coordinates w chosen as the mean image projection of the structure $\mathbf{w}_i = \frac{1}{m}\sum_f \mathbf{y}_{i,f}$. If the estimated geometric structure had accurately represented the real scene this texture image would be near constant for different viewpoints. In practice, the estimated structure is at best a coarse approximation of the scene, and the texture image varies for different viewpoints. However this variation can be smoothly parameterized using a basis B computed by principal component analysis from the sample image textures as

$$T_{w,f} = B\mathbf{a}_f,$$

where $\mathbf{a}_f$ is a view dependent texture modulation coefficient vector. It has been shown that the modulation coefficient $\mathbf{a}(\mathbf{r})$ varies smoothly with viewing angle r [15] and B captures the texture variability [7] up to a first order model of true intensity variation.

Hence a complete model is extracted from the video using m sample frames (typically 128-512) and is represented as a 3D geometry X, set of sample poses $\mathbf{p}_f = (\mathbf{r}, a, b, s)$, corresponding modulation coefficients $\mathbf{a}_f$ and texture basis B.

D. Human Machine Interface

To enhance operator's immersive sensation, we integrate and implement the user interface in which the robot motion is parameterized by the pose of the operator.

1) HMD Tracking:

To simulate the presence of the user in the remote environment the human computer interface must let the user move around. The variations in the pose of the operator are converted into commands for the manipulator and specify a view of the model.

[1]Here and in the following vectors are written bold and matrices capitalized.A bold capital indicates an image that has been flattened into a column vector

The current setup uses a magnetic tracker, a built-in compass-like mechanical device in the i-glasses HMD (from Virtual I-O), to determine the pose (roll, pitch, yaw) of the operator. The head tracker continuously sends its latest pose via an RS-232C serial interface to its host computer at about 250 HZ. The roll,pitch and yaw readings are converted to linear values where +/-16384= +/-180 degree. [16].

2) *Remote Motion:*

The remote robot mimics the operator's head motions. In our system setup, we employ a Surveyor's "Transit RCM" unit, a robotic camera mount system. [17] The Pan-Tilt has only two DOF, pan and tilt. It is based on "dead reckoning" in the sense that we specify the pan and tilt values, and the head points in whatever it considers to be that direction. We can't directly specify its velocities or accelerations. Despite of all the constraints, it is sufficient to replicate the operator head's yaw and pitch rotations. Referring to Fig.2, since the camera is mounted on the pan-tilt is always pointing approximately inwards to rotation axis of the pan-tilt, it allows the camera to move to different viewpoints on a sphere centered near the scene.

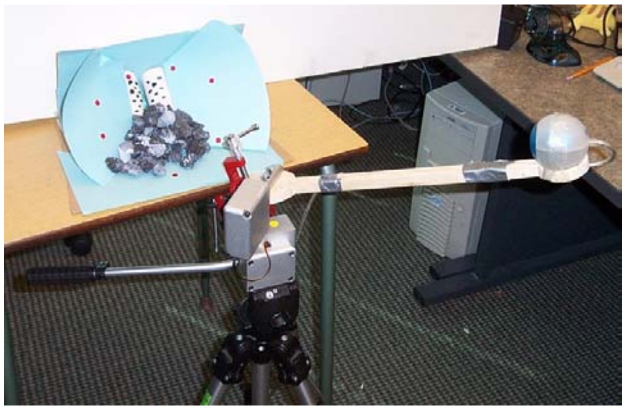

Fig. 2. Our prototype robot setup: a Pan-Tilt with an iBot camera mounted on an extended linkage (eye-in-hand).

E. *Rendering*

Instead of watching the delayed video streams from the remote site, the operator is shown the synthetic view immediately in response to head movements. The scene model renders a corresponding synthetic view of the remote scene as the head tracker continuously updates its pose. In our system, the appearance and the coarse geometry of the scene model are both indexed by a set of pose parameters $\mathbf{p}_f = (\mathbf{r}, a, b, s)$ obtained from SFM. A new view is rendered by first modulating a new texture corresponding to this view

$$T_w = B\mathbf{a}_f,$$

then reprojecting the geometric scene structure into the current camera frame, and the texture mapping onto it.

A stereo view is rendered from the model by a pose offset between the left and right HMD eye display. This is efficiently implemented using line stippling techniques in openGL, with even lines of the view rendered to one eye and odd lines rendered to the other eye of the HMD simultaneously.

IV. Experiments

A. *Prototype system*

To practically develop and evaluate our predictive display system, we set up a simple prototype. At one end of the lab we mount an standard web-cam on the pan-tilt, with motion control using a serial interface to a standard PC. At the other end of the lab, the operator is wearing an HMD connected to another PC. The PCs are connected via departmental Ethernet.

Since our camera motion device, the Pan-Tilt, only takes open loop "dead reckoning" commands, we can't directly specify the velocities/accelerations or acquire the exact completion time. In order to measure the remote system response time, we have to measure it indirectly.(Here by system response time, we mean the difference between the time we send the command and the Pan-Tilt finish the movement.) We track a feature point in the scene as soon as the Pan-Tilt starts it motion, and record the trajectory of the point's X-Y coordinates while it moves. As soon as the trajectory stabilizes, we pick that exact moment as the Pan-Tilt stops its movement. In the following, we adapt the notation (k, l) to indicate the Pan-Tilt's configuration where k indicates the pan, l indicates the tilt.

The profiles of two runs of the indirect measurement is illustrated in Fig.3. Starting from the same configuration (80, 90), the top profile shows the Pan-Tilt moves to (10,90) while the bottom one shows it moves to (70, 90). The stabilization in the bottom is faster than the top one because the Pan-Tilt rotates more degrees in the latter case. Because the movements of the Pan-Tilt is discretized, we can see curve is "jerky". Also, for the bottom profile, it only pans 10 readings(14 RAD). From start till stabilization, it takes about 1.4 seconds. The Pan-Tilt starts the motor and then slows quickly to adjust the overshoots, which contributes to the bumpy looking of the profile.

To study the relation between degrees of pan and the response time, we conduct an experiment: starting from the same configuration, we drive the Pan-Tilt to rotate 10, 20, 30, 40, 50, 60, and 70 readings for 20 runs each. We compute the average system response time for each trial. Showing in Fig.4 , it can be characterized as a linear system:

$$Time = Kd + T_0$$

Where d is the reading of the rotation. In this experiment, using linear least square fitting, we get K= 0.0217 and T_0 =1.3411. T_0 is the time required for transmission of the Pan-Tilt motion command and starting of Pan-Tilt's motor.

To study the step response time of our predictive system, we have a user wear the HMD, and perform view point re-orientations. The user waits until the image being shown is stabilized then turn to another view point. First, we provide the user with only the delayed video stream, and record the degrees of rotation and time (See Top graph of Fig.5). The flat/horizontal segments in the graph indicate that the user

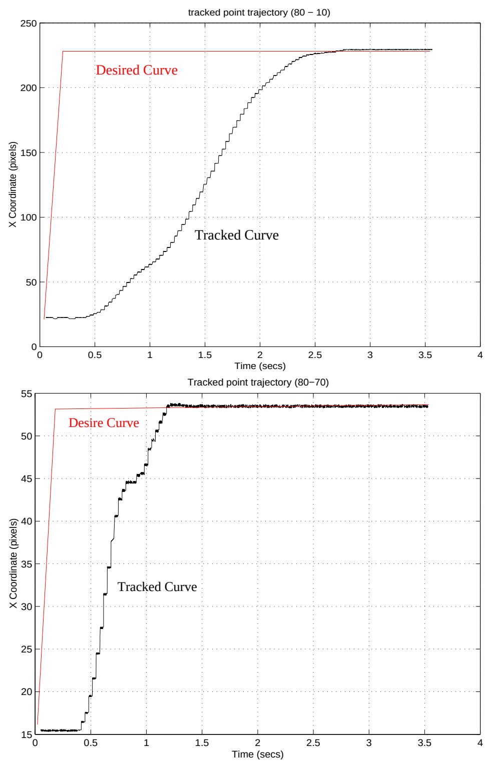

Fig. 3. Step response of the camera motion head measured by visual tracking of scene feature points. Top: Pan-Tilt moves from angles (80,90) to (10, 90). Bottom: Pan-Tilt moves from (80,90) to (70,90).

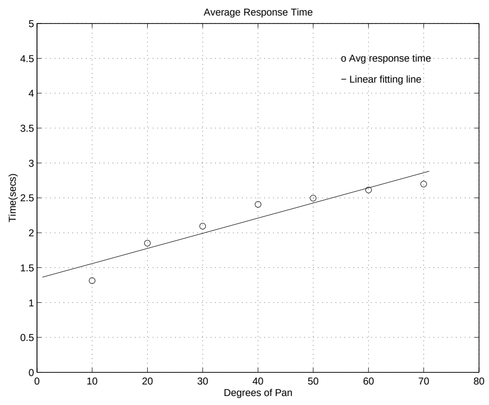

Fig. 4. System response time and degree of rotation can be linearly fitted to a straight line

employs "move-and-wait" strategy as he turns slowly. The user is able to accomplish 2-3 view point changes in 30 seconds. In the second experiment, we provide the user with predictive display and record the time and yaw readings from the HMD (See Bottom graph of Fig. 5). The user turns at high speed at first, slows down, and then speeds up again. As the user stops to observe the image, overshoot in the readings occurs due to the pose tracker in the HMD. We see that there is larger overshoot when the user turns at higher speed and smaller overshoot at lower speed. Referring to Fig.5, the step response time for the predictive display is approximately (or less than) one second, which is an improvement over the performance of 10-15 seconds with only delayed video stream.

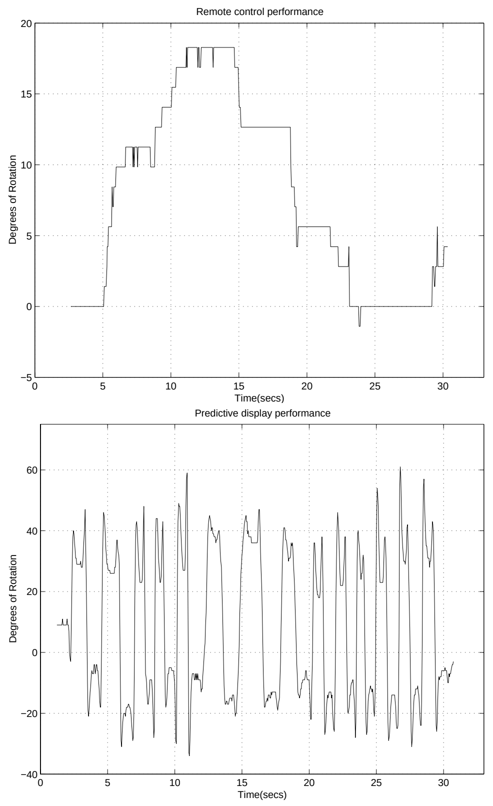

Fig. 5. An user wears the HMD and performs view point re-orientation. We record the degree of rotation and the time.Top: Remote control performance provided with only delayed video stream; Bottom: Performance with predictive display

B. Visual certification: modeling and rendering an oilsand crusher

The province of Alberta, Canada is rich in oil. Natural oils are stored in solid form "oilsand" instead of liquid. The first step in extractions involves using crushers, (see Fig.6) to mechanically achieve a uniform size distribution. A problem causing significant production delays is when foreign objects, such as the broken shoveler tooth in the picture gets stuck somewhere in the crusher.

We construct a scale model of this scene to certify that the technique of image-based modeling and rendering can be applied in predictive display. (See Fig.2) Instead of a shoveler's tooth, we use a small LED light placed in various places inbetween and partially occluded behind the crusher rollers, see Fig.7. The scene has fairly complex geometry which would be difficult to capture completely in a an a-priori graphics model. Using the $dynamic texture$ techniques, a very simple geometric model can be used including only easily trackable points, and the dynamic textures compensates for the geometric inaccuracy (note the LED is represented mostly by the dynamic texture, and not by geometry). In the figure rendered and real images can be compared.

Fig. 6. Close-up of a shoveler's missing tooth stuck in the oilsand crusher. Image from Syncrude Inc.

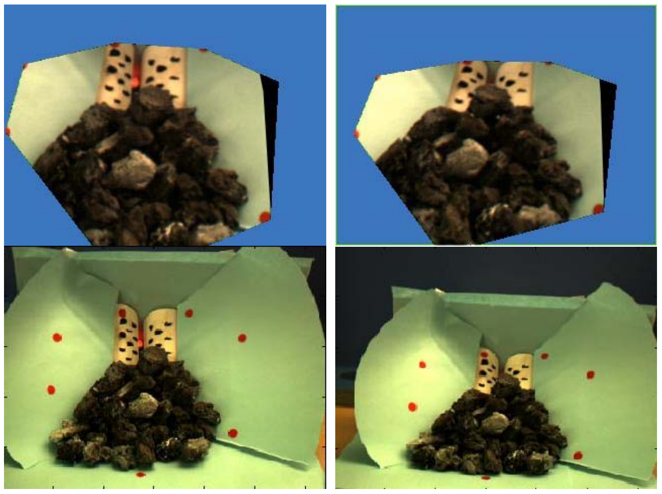

Fig. 7. Image sequence predicted for different viewpoints (top) and ground truth from corresponding (delayed) real images taken from the same view points(bottom). Images(left) with the led light detected, and image (right) with led light blocked.

V. Discussion

Being able to solve the problem of system delays in visuomotor tasks is essential for developing advanced man machine interfaces for teleoperation and increasing operator's performance. Yet, delays are unavoidable in any communication. We presented a predictive display system that overcomes the visual feedback delays by locally using an image-based model of a remote scene. Image-based models are graphics models created from images that can be used the same way as the more conventional 3D models. The model is created remotely by the manipulator.

Making use of the recent advances from various research areas in computer science, we implement and integrate a predictive display system, which is operated in two separate stages. The first stage is run offline and automatically captures a texture model of the remote scene. The second stage is run on-line and uses the captured model to synchronously render predicted views of the scene in response to the operator motion.

Our system is built from moderately priced consumer HW (about $350 for camera and Pan-Tilt, $700 for Virtual IO HMD), and it uses a common Internet connection instead of dedicated high speed video links.

Our results from the pan-tilt prototype experiment indicate that there is still a minor delay in our predictive display. But comparing with the network delays and robot system response delays which are in order of seconds, the system promises improvements on the operator's performance. The experiment of modeling and rendering oil sand crusher shows that using image-based methods allows us to render predicted/synthetic view in real-time without sacrificing the quality of the image.

In the future, we will use the 6 DOF PUMA robot to replace the Pan-Tilt in our current setup, and put the predictive display into the UofA 3D immersive CAVE.

References

[1] D. Miall, R. Weir and J. Stein, "Intermittency in human manual trackign tasks," *Journal of Motor Behavior*, vol. 25, no. 1, pp. 53–63, 1993.

[2] R. Miall, D. Weir, and J. Stein, "Visuomotor tracking with delayed visual feedback," *Neuroscience*, vol. 16, no. 3, pp. 511–520, 1985.

[3] A. Bejczy, S. Venema, and W. Kim, "Role of computer graphics in space telerobotics: Preview and predictive displays," in *Cooperative Intelligent Robotics in Space, pages 365–377*, November 1990. [Online]. Available: citeseer.nj.nec.com/bejczy90role.html

[4] T. B. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, October 1993.

[5] K. Yerex, D. Cobzas, and M. Jagersand, "Predictive display models for tele-manipulation from uncalibrated camera-capture of scene geometry and appearance," in *IEEE International Conference on Robotics and Automation(ICRA)*, Taiwan, May 2003.

[6] G. Geist and V.S.Sunderam, "Network-based concurrent computing on the pvm system." *j-CPE*, vol. 4, no. 4, pp. 293–311, June 1992.

[7] D. Cobzas, K. Yerex, and M. Jagersand, "Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion," in *EuroGraphics 2002*, September 2002.

[8] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography-a factorized method," *Int'l Journal of Computer Vision*, vol. 9, pp. 137–154, 1992.

[9] G. Hager and T. K., "X vision: A portable substrate for real-time vision applications." *Comuter Vision and Image understanding*, vol. 69, no. 1, pp. 23–37, 1998.

[10] G. Hager and P. Belhumeur, "Efficient region tracking with parametric model of geometry and illumination," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, 1998.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[12] D. Cobzas and M. Jagersand, "A comparison of non-euclidean image-based rendering." in *Graphics Interface 2001 (poster session)*, Ottawa, Canada, June 2001.

[13] M. Jagersand, "Image based view synthesis of articulated agents," in *Computer Vision and Pattern Recognition*, 1997.

[14] S. Soatto, G. Doretto, and Y. Wu, "Dynamic Textures," July 2001, pp. 439–446. [Online]. Available: citeseer.nj.nec.com/soatto01dynamic.html

[15] S. Murase and S. Nayar, "Visual learning and recognition of 3-d objects from appearance," *International Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[16] *i-glasses! Tracker software development Tookit, Version 1.2*. Virtual I-O, 1995.

[17] *Transit RCM, Robotic Camera Mount User Guide*. Surveyor Corporation.