Scene Model Acquisition for a Photo-Realistic Predictive Display

Tim Burkert[1] and Georg Passig[1]

Institute for Real-Time Computer Systems, Technische Universität München, D-80333 Munich, Germany, georg.passig@rcs.ei.tum.de,
WWW home page: http://www.rcs.ei.tum.de/

Abstract. Predictive displays have proven their suitability to compensate time delays in the visual feedback of teleoperation applications. Using camera images for texture mapping and a geometric model of the remote scene photo-realistic predictions can be achieved. The aim of the presented work is the acquisition and update of a geometric scene model of the remote scene using camera images. A hierarchical structure of the model is described, that facilitates the transformation of 3D point clouds to a polygon-based description for computer graphics. In addition to the concepts, some experimental results in scene reconstruction are presented.

1 Introduction

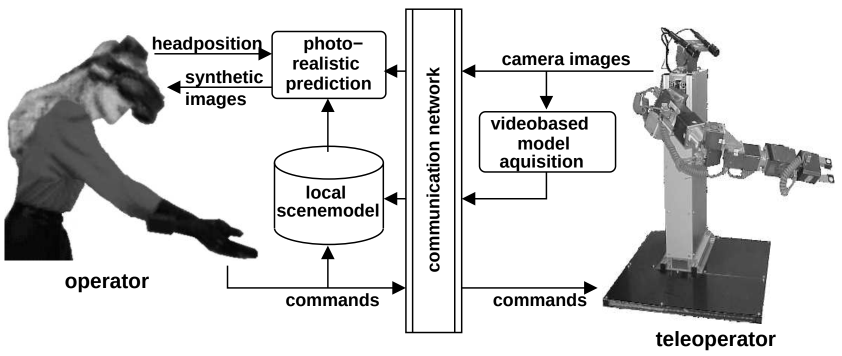

Fig. 1. Teleoperation scenario with a photo-realistic predictive display

In a common teleoperation system, actions of the human operator such as motions of his hand and head are measured by pose-tracking devices and a data-glove. These actions are mapped into commands which are sent over the communication network. The latter causes a delay before the robot executes the commands. Video images as feedback of the operation are transmitted back through the same communication channel, adding an extra delay before they can be displayed. It is the aim of this work to compensate for these disturbing delays using a photorealistic predictive display.

Several variations of *predictive displays* have been suggested, which superimpose simple [18] or complex [1, 12, 13, 25] graphics on camera images (augmented reality), or completely replace them by synthetic images (virtual reality) [9].

An overview of research related to time delays in teleoperation can be found in [22].

In contrast to image based approaches for the prediction of scenes as used in some of the above approaches and [10], an explicit polygonal mesh representation is chosen in the presented system. Only the local availability of geometry, kinematics and dynamics of the remote scene allow prediction of scene content. Simplification of generated meshes nevertheless permits fast rendering of even complex scenes. A good survey of the vast area of mesh simplification is presented in [6, 8].

The need for scene models in autonomous robotics has been a major source of research in the past years but with growing interest in virtual reality the emphasis of modeling shifts more to visual qualities of the models.

Reconstruction and rendering of real objects and scenes has been discussed for many purposes. The *Virtualized Reality*^{TM} project by Kanade et al. [11] uses a *sea of cameras* around the scene to create digitized 3D-models of real-world events. Multi-baseline-stereo, visual hull techniques or a combination of them are usually used in such systems [3, 26, 7]. In a variety of other projects visual hulls are reconstructed for virtual reality in real-time [16], some of them with hardware-acceleration on graphics cards [15, 14]. In contrast to the presented approach they rely on multiple cameras surrounding the scene of interest.

Immersive interaction in the context of video conferencing is commonly based on polynocular stereo systems for fast and accurate model generation [17, 20]. Synchronized cameras facing the user in an arc are interpreted in different combinations as bi- and trinocular stereo systems.

Systems with a single moveable camera allow the exploration of large unknown scenes [19]. Capturing textured 3D models of historical items based on camera vision is one of the leading scenarios in this area. The problem of camera registration adds complexity in those approaches.

In contrast to some of the mentioned approaches, the presented work uses a single stereo camera mounted on a pan-tilt unit. This allows later expansion of the system into a mobile platform for telepresence.

Among the most comparable systems Sequeira et al. [21] presented an autonomous robot reconstructing textured three dimensional environment models. They rely on laser range sensor information for extracting model information and video cameras as source for textures.

Section 2 introduces the overall concept and the predictive display. The acquisition of scene models is explained in section 3. The paper closes with first results in section 4 and a conclusion in section 5.

2 System Overview

Figure 1 shows the structure of a teleoperation scenario incorporating the predictive display. Actions of the human operator, such as motions of his hand and head are measured by pose-tracking devices and a data-glove. These actions are mapped into commands that are transmitted over the communication network and are executed by the robot. This first transmission induces a delay between the generation of a command and its execution due to the latency of the network. Next, the video images showing the execution of the commands are sent back over the network to the operator. This second transmission again adds a delay, consisting of the latency of the network and the transmission time of the image data.

In order to obtain an immediate visual feedback, the local scene model is continuously modified according to the commands. Based on this model, a synthetic image is generated and presented to the operator via a head-mounted display (HMD) with no noticeable time delay. In parallel, the scene model is updated by the sensor information acquired by the teleoperator.

Photo-realism is achieved by extracting textures from delayed *real* images. These are then used for rendering the *predicted* images. To keep the prediction as accurate as possible and to cope with changes of illumination in the remote scene, the textures also have to be continuously updated and verified.

2.1 Scene Prediction

The actual scene prediction is described in detail in [2]. A short overview is given here to clarify the application and the demands for the model acquisition.

Fast **extraction** of the required **textures** is one of the the main challenges. The concept is shown in figure 2-left. Textures are extracted from images taken by the teleoperator‘s cameras. Besides the image the scene model at the time the image was taken as well as the intrinsic and extrinsic camera parameters must be known. The following steps are performed for extracting textures from one camera image.

For each polygon an appropriate area in the camera image is determined by calculating the projection of its vertices using the camera parameters. The bounding boxes enclosing these areas are stored as textures.

Parts of textures contain invalid color information when a polygon is occluded by another one. This leads to parts of objects in the front being drawn on objects in the background during rendering. Therefore in a next step such areas are detected and marked as invalid in the textures.

So far a perfect camera registration and a perfectly modeled remote environment were assumed. In reality these assumptions do not hold. Wrong parts of the camera image are assigned to polygons. This causes artifacts in the textures. Therefore uncertain information is discarded.

Up to this point the textures contain the perspective of the camera image. They are now normalized, i.e. each texture is transformed into a view orthogonal

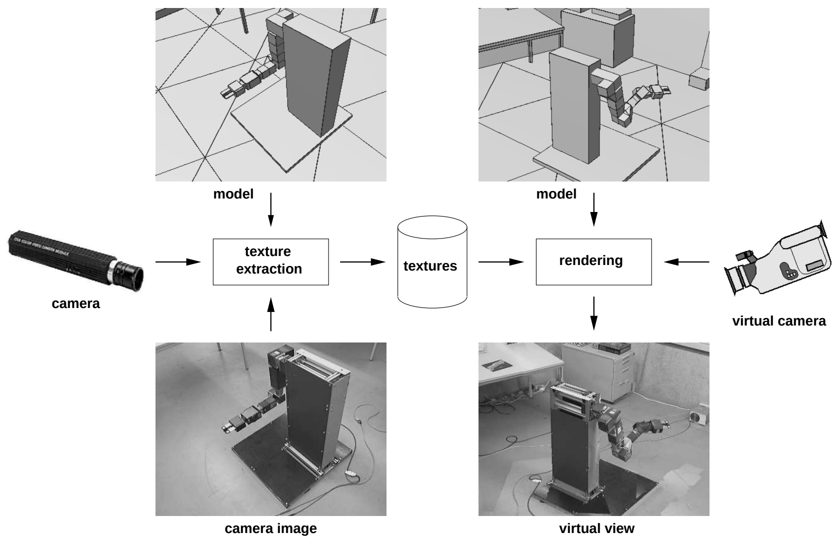

Fig. 2. Texture extraction and rendering

toward the corresponding polygon. As a result the required texture memory for a polygon is directly proportional to the polygon's size.

For each polygon there may already be a texture extracted from older camera images. This texture is merged with the new one. The result contains all color information from the current image plus the "holes" filled as far as possible with old information.

Finally, parts of textures that still contain no valid information (e.g. parts of polygons that were never visible) are filled. This is done by interpolating the color information at the border of such "holes".

To accelerate the processing large parts of the algorithms are performed directly on the graphics card using OpenGL.

The **rendering** of the predicted scene is a standard rendering algorithm using the previously extracted textures (fig. 2-right). The parameters of the calibrated HMD are used for the projection.

3 Acquisition of the Scene Model

3.1 Scene Model

A scene model for telepresence applications has to meet some special requirements. The manipulator, the tools, and the objects to be manipulated are in the center of the user's attention. Errors in the model, high update latencies, and low modeling resolution are critical factors for these objects because they disturb or even impair the user in executing his task. Information about the position of objects in this *foreground* is either known (i.e. the teleoperator) or has to be acquired by object tracking. All other parts of the scene, the user does not interact with directly, have to be modeled to improve the user's orientation and depth impression: even small movements of the user's head produce a strong depth cue when the foreground shifts against the *background*. No syntactic information about it is needed, therefore no object recognition or localization has to be performed. Low update rates are sufficient. This work focusses on the acquisition and update of the background. An overview of the image processing system is shown in figure 3.

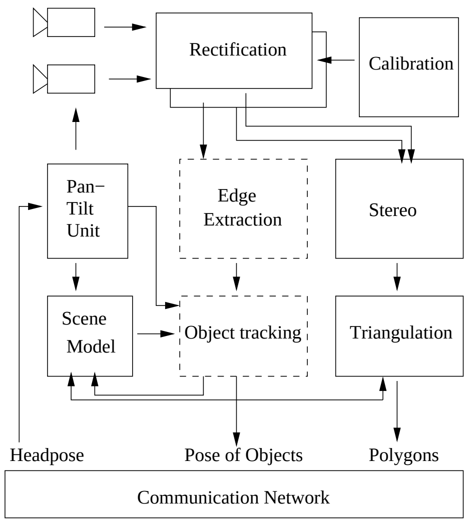

Fig. 3. Image processing system overview

The acquisition of the scene model is not a bound process. Rather than dealing with a limited set of images with associated 3D data, a sequence of image data with open end has to be processed. Every time a new image is taken, the model has to be updated accordingly. Difficult problems like the limitation of model size, error accumulation over time and fusion of new data into the existing model arise.

Following the data flow, this section will begin with some basics of image acquisition and some preprocessing steps. Extraction of depth data from stereo images leads to segmentation and triangulation of 3D voxeldata. Fusion of subsequent images into a scene model is completed by the transmission of new data over the communication channel to the predictive display.

3.2 Camera Model

Metric information about the scene has to be retrieved, so a calibration of the intrinsic and extrinsic camera parameters has to be performed. A pinhole camera model with first order radial distortion is used. *HALCON* – a commercial image processing tool [1] – is used to determine the intrinsic parameters of each camera using a calibration pattern. The *extrinsic parameters* are estimated following the concepts and using the software of Z. Zhang [27]. The resulting projection matrices are scaled to euclidian space using one stereo image of the known calibration table.

3.3 Image Rectification

To be able to run an area-based stereo algorithm efficiently *scan line correspondence* between the two images is useful. Straight and parallel epipolar lines are of course not found in practice, so rectification of the images is performed. After removing radial distortion an affine rectification transformation calculated from the real projection matrices of the cameras [5] is applied. This combined rectification transformation can be expressed as a simple look-up-table that is calculated offline after camera calibration. For each pixel in a rectified target image it holds the two coordinates in the original image from which the color of the target image should be taken (using bilinear interpolation).

3.4 Depth Extraction

Two similar stereo systems are used at the moment in the project to generate disparity images out of rectified stereo images: A home grown standard area-based stereo approach mostly following [24, 4] will be explained in detail in this section. It is competing with the commercially available Small Vision System[2].

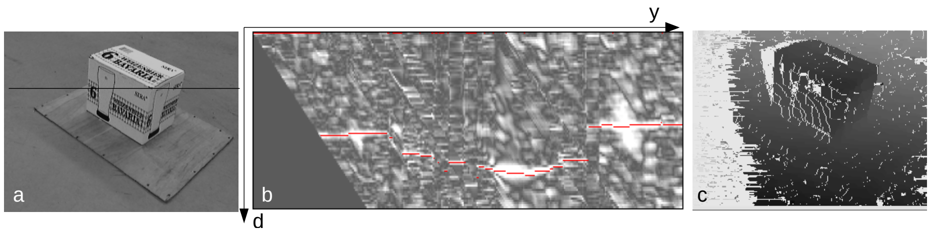

Fig. 4. Left camera image (a), a slice of the cost matrix (b) of the scanline marked black in (a), and the resulting disparity image (c)

Cost Matrix. Rectangular windows in the two images are compared using a cost function. The result, i.e. the cost for a match at a certain disparity d at a

[1] www.mvtec.com
[2] www.videredesign.com

single pixel position in the reference image, is used to build a cost matrix with dimension $width * d_{max}$ for each scanline as shown in figure 4 b. Normalized cross correlation (ZNCC) is used as cost function.

Best Match. A decision on which disparity is the *correct* one for each pixel in the reference image is based on the entries in the cost matrix. For performance issues, line by line dynamic programming is preferred over an approach using the entire image for optimization[28]. It has constant calculation time and it provides good results compared to the simple maximum search. The *Extended Neighbourhood Cost function* proposed by Falkenhagen [4] is used for the calculation of the best path through the cost matrix slice.

Occlusion Detection and Confidence Test. Occluded regions in one image show a characteristic correspondence pattern in the disparity path of a scan line. Multiple pixels in one image are mapped to one single pixel in the other. These occluded areas can not be considered as trustworthy and are therefore marked as invalid in the disparity image.
For each stereo pair, one disparity image relative to each camera image is generated and compared. Pixels that do not satisfy this right to left confidence test are marked as invalid, since the two disparity images hold different information for a given point in the scene.

Subpixel Accuracy and Hole Filling in the Disparity Map. Subpixel accuracy can be obtained by referring once more to the entries of the cost matrix. The exact disparity at a pixel corresponds to the position of the maximum of a parabola matched to the entries of the cost matrix at $d-1, d$ and $d+1$.

Small invalid areas and outliers in the disparity map are filled with neighboring disparity values. To reduce the *visible* errors in the virtual model view these holes are always filled with the smallest disparity in the neighborhood.

3.5 Structure of the Scene Model

In polygonal models based on depth information, polygons serve as an abstraction layer of the 3D voxel data. Each resulting triangle is based on its own group of voxels in voxel space. The smaller triangles get, the more they are vulnerable to noise in this sensory input. As a consequence one gets significant variations of the normal vectors of triangles, even within clearly flat areas (i.e. the floor in a room). To avoid these problems an abstraction layer between voxel space and triangles is introduced: *point clouds*. The voxel data can now be segmented into regions with similar characteristics before appropriate triangulation.

3.6 Eigenbasis of Point Clouds

As a main characteristics of a point cloud M of data points $\{\underline{x}\}$ one would be interested in its “flatness”. A point cloud can be described as flat if its variance along its normal vector is small compared to its variance along the other coordinate axes. To find the *eigenbasis*, a coordinate system aligned with the directions of maximum variance, the data set has to be shifted into the origin so that $E\{\underline{x}\} = \underline{0}$ and the covariance $\underline{C}$ of $\{\underline{x}\}$ has to be calculated:

$$\underline{C} = \frac{1}{M} \sum_{k=1}^{M} \underline{x}_k \underline{x}_k^T \tag{1}$$

The eigenvalues λ_i of $\underline{C}$ correspond directly to the variance along the axes of the new coordinate system, the eigenvectors $\underline{a}_i$. Recapitulating it can be said that

$$\underline{C}\,\underline{a}_i = \lambda_i \underline{a}_i \tag{2}$$

$$\boxed{\underline{y} = \underline{A}^T \left(\underline{x} - \underline{m}\right)} \qquad \text{with } \underline{m} = E\{\underline{x}\} \tag{3}$$

$$\begin{aligned} \underline{A} &= \text{ Matrix of eigenvectors of } \underline{C} \\ \underline{A} &= (\underline{a}_1, \underline{a}_2, \underline{a}_3) \quad , \quad \lambda_1 > \lambda_2 > \lambda_3 \end{aligned}$$

3.7 Segmentation of Voxel Data into Planar Patches

The voxel data for an image has to be segmented into point clouds with different properties. Simply speaking, all planar patches in the scene should be found, because their initial triangulation is very simple and fast. Once small flat patches are found, they can be grown iteratively, but initially finding them is the first challenge. When many uniformly distributed patches are used as seeds for the evaluation of their "flatness" their size is a critical factor. A tradeoff between reliability of their eigenbase (patches are too small) and processing cost has to be found. A good guess for starting regions is therefore desirable to considerably speed up the detection of planar patches.

The process of selection of seed patches, evaluation and iterative growth is done twice: Using 2D region growing once on the original camera images and once on the depth map. Each point cloud corresponding to a patch is transformed into its eigenbase. Comparing each point's coordinates to limits based on the variance along the respective axis, outliers are removed and the eigenbase is recalculated. Different quality criteria are now evaluated for the patches to decide if they are flat:

- **Unstable Orientation:** To check the stability of the new eigenbase, the procedure of reducing outliers is repeated once more. If the normal vector changes for more than a small angle during this second reduction, the patch can neither be very reliable nor flat.
- **Unbalanced Eigenvalues:** If the second smallest eigenvalue is not considerably larger than the smallest, one can not be sure of the correct normal direction of the patch. This condition is more likely to occur further away from the camera, since the position uncertainty along the camera axis grows with the distance.
- **Large Eigenvalues** indicate non-flat areas.

The remaining patches are now extended with points in their vicinity whose normal distance to the plane is small enough. The eigenbase of the growing plane is recalculated, whenever the number of points has considerably increased. Growth stops when there are no more matching points, or when a patch starts getting too frayed. Figures 5 and 6 show an image, its disparity map and the patches before and after selection and growth.
The described process is now repeated for newly created seed patches this time based on the disparity image.

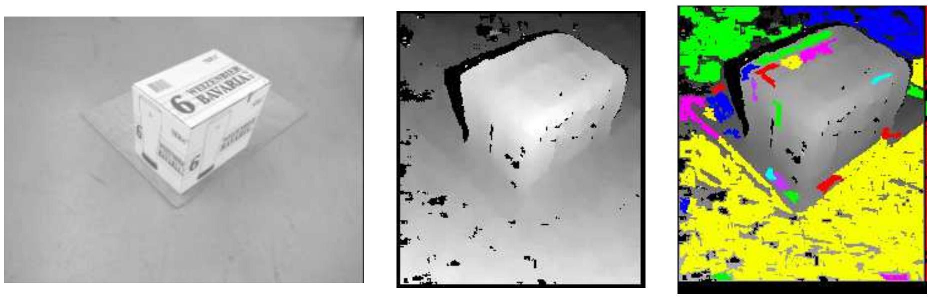

Fig. 5. A rectified camera image, its disparity map and a set of seeds for flat patches

Fig. 6. Seeds for planar patches after selection and growth and the final result

3.8 Model Structure

The scene model is designed in three layers: Point clouds, voxel data, and triangle-vertex information.

Point clouds "own" their voxels and manage the data associated with them: The covariance matrix, a 2D region corresponding to the projection using the actual camera pose, boundary data, and a list of triangles. The triangle mesh itself is topologically connected through pointers as depicted to the right.

$$\begin{tikzcd}
& \text{Triangle} \arrow[dl, shift right, "3"'] \arrow[dr, "3"] & \\
\text{Edge} \arrow[ur, shift right, "2"'] \arrow[rr, "2"] & & \text{Vertex}
\end{tikzcd}$$

3.9 Triangulation

In a typical fine to coarse mesh generation, one usually starts by transforming the disparity data into a very fine triangle mesh. For a typical disparity image (320x240, sparse), this method creates about 38,000 triangles. The subsequent mesh thinning would spend most of its processing time reducing this information to 1/10th. By choosing an adapted initial triangulation this unnecessary calculation time can be reduced at the cost of a slightly higher overall error in the final mesh.
All point clouds are triangulated independently. Constrained 2D *delaunay* triangulation is used to generate a mesh out of non-uniformly distributed vertices with connecting edges in a patch. Three different sources of constraining input data for the triangulation can be identified:

- The outer borders of patches are approximated by line segments. Their endpoints are used as vertices, the edges are used as triangle edges (fig. 7, left).
- Borders of neighboring patches are input as constraints into the triangulation.
- Borders of holes within a patch are treated just like outer borders.

The result are vertex points, edge segments, and holes that are passed to the constrained 2D-Delaunay triangulation. The software package "Triangle" of Jonathan Shewchuk [23] is used for this triangulation with great success.
Point clouds can exist in planar and unplanar versions, so special measures have to be taken, to triangulate unplanar areas correctly. To achieve this, the depth image is filtered for depth discontinuities (using a *kirsch* filter mask). The resulting 2D *constraint map* is used within the 2D triangulation as a constraint for the triangle size. Summing up the (squared) entries in the constraint map within the triangle area gives a criterion for the decision if a triangle has to be subdivided further by adding *steiner points* to the mesh. Figure 7, right, shows a triangulation composed solely out of unconnected flat patches as seen from a different camera pose.

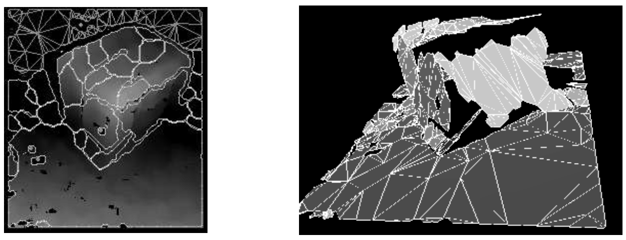

Fig. 7. Triangulation

3.10 Mesh Optimization

Mesh optimization is always a tradeoff between the number of triangles, the quality of the mesh, and processing time. Operations for mesh optimization can roughly be split into three groups. Operations like **edge swapping** and **moving vertices** do not change the number of triangles, but can reduce the error. Decimation techniques like **removing a vertex** or **collapsing an edge to a point** reduce the number of triangles and increase the overall error in the mesh. Refinement and subdivision methods reduce the error but increase the number of triangles. A **vertex is created** by splitting an edge into two at an arbitrary point. Most operations are capable of introducing irreversible errors in the logical structure of the mesh when applied to special triangle configurations. Special care has to be taken to detect such operations.

The mesh optimization should reach an optimum between calculation time and quality of the result. All methods mentioned above can be freely combined when approaching this aim, but no guideline on how to find an optimal combination in terms of processing time and resulting mesh quality can be found. In the presented system a starting mesh with adaptive medium resolution is generated as described before. It is optimized with one pass of edge swapping and decimated with edge collapses until a specified maximum mesh error is reached. The edge collapse is applied to the edge whose collapse would create the lowest increase in the error of all neighboring triangles.

3.11 Fusion of Different Views

Each new view of the scene must be used to update the model information. This is done by direct comparison of the new disparity map with the existing model. To facilitate this process, a simulated disparity map is created from the existing model using the new camera position. This allows a pixel by pixel decision on the new disparity map. To allow the use of meaningful decision limits, the pixels are back-projected into 3D voxels using the disparity value, the pixels coordinates (x,y), and the calibration data of the rectified camera system. Depending on where a voxel is found relative to the existing model, changes in the model are initiated. If a voxel is found behind a triangle, the triangle is removed and the point cloud is updated accordingly. If it is close to a point cloud, it is associated to it. A new point cloud is created in all other cases. This decision is done on a voxel by voxel basis, but changes in the model are only initiated, if a sufficient number of voxels imply a change. Figure 9 shows the fusion of two images for a simplified case without point clouds.

4 Results

4.1 Experimental Setup

For the measurement of the operator's movements, the information of an *Ascension Flock of Birds* magnetic sensor system is used. Sensors are mounted in a gripper and on the *Sony HMD800* head-mounted display that displays the currently predicted view. The position of the user‘s hand controls a 7-joint modular robotic system in an anthropomorph, right-hand configuration. A pan-tilt unit completes the mechanic system. Two *Sony DFW-V500* firewire cameras with 8mm lenses mounted in an approximately parallel configuration 0.12m horizontally apart from each other or a *MEGAD* stereo camera from *Videre-Design* are used as image sensors. Polarizing lenses reduce reflections in the scene. The camera system can also be mounted on a tripod to allow translational movements. Latencies of the transmission channel are simulated by adding a delay to the unidirectional control flow from the tracking system to the manipulator and the pan-tilt head.

The computer graphics calculations are performed on a standard PC with an *AMD Athlon* Processor with 1GHz and 512MB DDR-RAM. The graphics card is based on an *NVidia Geforce 4 Ti4600 Processor* with 128MB memory. OpenGL 1.3 is used as graphics API on a Linux system. Image processing is also done on a Linux system based on a *Intel PIII* with 733MHz and 256MB RDRAM.

4.2 Current Implementation Status

Photo-realistic scene prediction without noticeable time delay can already be shown in the system when the display is based on a manually acquired model (like the model in fig. 2). Most of the algorithms concerning memory and calculation intensive textures are executed directly on the graphics hardware. Reasonable execution times are achieved even on consumer graphics cards. The predicted scene is rendered with about 100 frames per second. The time for extracting textures depends on the model. In the example from figure 2 a scene with about 2,000 polygons this time is usually about 500ms. An mpeg-video of a motion in a predicted scenario can be found at **http://www.rcs.ei.tum.de/research/rovi/tele/**.

The execution time of the image processing part is currently not sufficient for a fast model update. For images with the resolution of about 320x240 and a maximum disparity of 45 pixels the calculation of the disparity is performed in 4.5s including right-to-left confidence test and subpixel accuracy. Segmentation of the disparity image is done in about 15s, but especially this module is not implemented in an efficient way. The execution time for the creation of constraining edges around the patches and their 2D triangulation averages out at 1s but is depending on the number and complexity of patches. Decimation of very fine meshes takes about 25s (reducing the number of triangles from 2,800 to 800) for a completely unknown scene. When only parts of the scene have to be retriangulated all mentioned times after stereo processing reduce linearly. Until now no effort was put into a speed-optimized implementation of the segmentation and triangulation.

Closed-loop operation of the system with fast computer graphics and slower model update has already been tested. A stable tracking of objects in the foreground and registration of the camera in the scene are missing steps to complete integration of the system. The process of triangulation, decimation and fusion of triangle meshes is illustrated in figures 8 and 9. As the image processing system undergoes a major redesign at the moment, these examples are still based on a software version without planar patches and with simple triangle constraints.

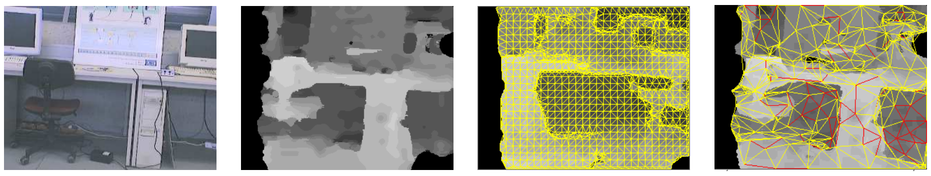

Fig. 8. A rectified camera image, its disparity map, the initial triangulation and the decimated result mesh

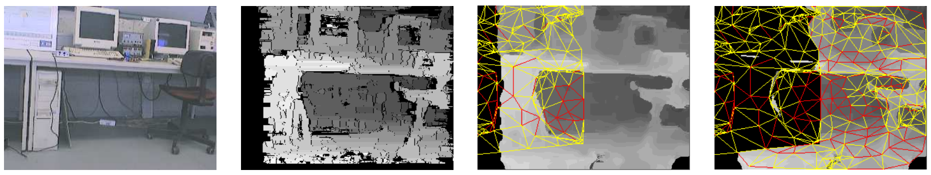

Fig. 9. A rectified camera image with different camera pose, its unfiltered disparity map, the filtered map with the existing model and the result mesh

5 Conclusion and Future Work

This paper described the concepts and an implementation of a system for automatic scene model acquisition in the context of telepresence. A fast low-cost scene reconstruction based on stereo images, and a *photo-realistic* visualization based on texture mapping which uses low-cost 3D graphics cards and OpenGL were presented.

Future work will focus on completion and speedup of the image processing system and the integration of different 3D data types into one model. Further experiments will be conducted with different robotic systems and scenarios including endoscopic surgery and mobile robotics.

In the sector of texture extraction, we are working on an algorithm for facilitating texture extraction in parallel to rendering.

6 Acknowledgement

This work was supported in part by the German Research Foundation (DFG) within the Collaborative Research Centre SFB 453 on “High-Fidelity Telepresence and Teleaction”.

References

[1] Antal K. Bejczy, Won S. Kim, and Steven C. Venema. The Phantom Robot: Predictive Displays for Teleoperation with Time Delay. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'90)*, pages 546–551, Cincinatti, Ohio, 1990.

[2] Tim Burkert, Jan Leupold, and Georg Passig. A photo-realistic predictive display. *TO APPEAR IN: Presence*, 2003.

[3] Kong Man Cheung, Simon Baker, and Takeo Kanade. Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.

[4] Lutz Falkenhagen. Depth estimation from stereoscopic image pairs assuming piecewise continuous surfaces. In *Proc. of European Workshop on combined Real and Synthetic Image Processing for Broadcast and Video Production*, Hamburg, Germany, 1994.

[5] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. Rectification with unconstrained stereo geometry. In A. F. Clark, editor, *Proceedings of the British Machine Vision Conference*, pages 400–409, September 1997.

[6] Michael Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics 99 – State of the Art Reports*, pages 111–131, 1999.

[7] Markus Gross, Stephan Wrmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, and Andrew Vande Moereand Oliver Staadt. blue-c: A spatially immersive display and 3d video portal for telepresence. In *SIGGRAPH*. ACM, 2003.

[8] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. In *Multiresolution Surface Modeling Course Notes*, SIGGRAPH. ACM, 1997.

[9] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl. ROTEX – The First Remotely Controlled Robot in Space. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'94)*, volume 3, pages 2604–2611, Los Alamitos, CA, USA, May 1994. IEEE Computer Society Press.

[10] Martin Jägersand. Image-based predictive display for high d.o.f. uncalibrated tele-manipulation using affine and intensity subspace models. *Advanced Robotics*, 14(8):683–701, February 2001.

[11] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 4(1):34–47, 1997.

[12] Won S. Kim. Virtual Reality Calibration and Preview/Predictive Displays for Telerobotics. *Presence*, 5(2):173–189, 1996.

[13] Won S. Kim, Donald B. Gennery, Eugene C. Chalfant, Lucien Q. Junkin, Ivan M. Spain, and Suzie B. Rogers. Calibrated Synthetic Viewing. In *Proc. of the ANS 7th Topical Meeting on Robotics and Remote Systems*, volume 1, pages 596–602, Augusta, Georgia, April 1997. American Nuclear Society.

[14] Ming Li, Marcus Magnor, and Hans-Peter Seidel. Hardware-accelerated visual hull reconstruction and rendering. In *Proc. of Graphics Interface*, 2003.

[15] Benjamin Lok. Online model reconstruction for interactive virtual environments. In *Proc. 2001 Symposium on Interactive 3D Graphics*, pages 69–72, March 2001.

[16] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *SIGGRAPH 00 Conf. Proc.*, 2000.

[17] J. Mulligan, N. Kelshikar, X. Zabulis amd, and K. Daniilidis. Stereo-based environment scanning for immersive tele-presence. *IEEE Trans. on Circuits and Systems for Video Technology, Special Issue on Immersive Telepresence*, 2003.

[18] M. V. Noyes and T. B. Sheridan. A Novel Predictor for Telemanipulation Through a Time Delay. In *Proc. of the 20th Annual Conf. on Manual Control*, NASA Ames Research Center, Moffet Field, CA, 1984.

[19] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Automated reconstruction of 3D scenes from sequences of images. *ISPRS Journal Of Photogrammetry And Remote Sensing*, pages 251–267, 2000.

[20] Radim Ŝára. Accurate natural surface reconstruction from polynocular stereo. In R.Bajcsy A. Leonardis, F. Solina, editor, *Proceedings NATO Advanced Research Workshop: Confluence of Computer Vision and Computer Graphics*, number 84 in High Technology, pages 69–86. Kluwer Academic Publishers, 2000.

[21] Vitor Sequeira, K. Ng, E. Wolfart, J.G.M. Gonçalves, and D. Hogg. Automated reconstruction of 3D models from real environments. *Jounal of Photogrammetry & Remote Sensing*, 54(1):1–22, February 1999.

[22] Thomas B. Sheridan. Space teleoperation through time delay: Review and prognosis. *IEEE Transactions on Robotics and Automation*, 9(5):592–606, October 1993.

[23] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[24] Changming Sun. A fast stereo matching method. In *Digital Image Computing: Techniques and Applications*, pages 95–100, Massey University, Auckland, New Zealand, December 1997.

[25] G. Thomas, T. Blackmon, M. Sims, and D. Rasmussen. Video engraving for virtual environments. *Electronic Imaging*, 1997.

[26] Stephan Würmlin, Edouard Lamboray, Oliver G. Staadt, and Markus H. Gross. 3D Video Recorder. In *Proc. of the 10th Pacific Conference on Computer Graphics and Applications*, 2002.

[27] Zhengyou Zhang. A new multistage approach to motion and structure estimation: From essential parameters to euclidean motion via fundamental matrix. Technical Report No.2910, INRIA Sophia-Antipolis, France, June 1996.

[28] C. Lawrence Zitnick and Takeo Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000.