EURASIP Journal on Applied Signal Processing 2002:10, 1067–1087

Real-Time Multi-Step View Reconstruction for a Virtual Teleconference System

B. J. Lei

Information and Communication Theory Group, Department of Mediamatics, Faculty of Information Technology and Systems (ITS), Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
Email: B.J.Lei@its.tudelft.nl

E. A. Hendriks

Information and Communication Theory Group, Department of Mediamatics, Faculty of Information Technology and Systems (ITS), Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
Email: E.A.Hendriks@its.tudelft.nl

Received 29 August 2001 and in revised form 20 February 2002

We propose a real-time multi-step view reconstruction algorithm and we tune its implementation to a virtual teleconference application. Theoretical motivations and practical implementation issues of the algorithm are detailed. The proposed algorithm can be used to reconstruct novel views at arbitrary poses (position and orientation) in a way that is geometrically valid. The algorithm is applied to a virtual teleconference system. In this system, we show that it can provide high-quality nearby virtual views that are comparable with the real perceived view. We experimentally show that, due to the modular approach, a real-time implementation is feasible. Finally, it is proved that it is possible to seamlessly integrate the proposed view reconstruction approach with other parts of the teleconference system. This integration can speed up the virtual view reconstruction.

Keywords and phrases: teleconference, 3D vision, view reconstruction, real-time, virtual.

1. INTRODUCTION

There is a recent trend in computer graphics and computer vision to merge with and benefit from each other [1]. Computer graphics is good at providing high-quality 3D perception with the help of complex 3D models [2]. However, human eyes are still able to easily distinguish a computer-generated picture out of real-life photographs [3]. On the other hand, in the computer vision field we have photorealistic images, from which various kinds of meaningful information could be extracted [4]. But until now, unless in a restricted environment, the information we can get is not sufficient to automatically construct very accurate 3D models [5]. Modelling complex, nonrigid objects like the human body and face [1], which is important for telepresence video conference systems, is even more difficult. Approaches like the *Image Based Modeling* (IBM) [6] and the *Image Based Rendering* (IBR) [7] have been proposed in order to efficiently bridge the gap between these two fields. This exchange of knowledge and experience is proved to be very promising [8].

Within the Fifth European Framework project VIRTUE (VIRtual Team User Environment) [9], a 3-party virtual teleconferencing system, is working steadily in order to realize the convergence of computer graphics and computer vision in a real application. In this system the participants are given a convincing impression of presence in a semi-immersive environment. It is required that this environment is characterized by eye-to-eye contact, gaze awareness, direct body language, life-sized portraits, and shared work space (Figure 1). It was found that, in order to fulfill these requirements, it is essential to adapt the perceived views of all conferees to their viewpoint change. For example, in Figure 2, if conferee A wants to be given a looking around feeling about conferee C (in order to have a telepresence of C at the location of A), the correct view about C corresponding to the viewpoint of A should be reconstructed in real time from the two fixed views coming from cameras 1 and 2 at site 2. (For more detailed discussion about VIRTUE, see Appendix A.)

Real-time novel view reconstruction is the generation of arbitrary novel views from limited number of known views and it is crucial for the success of VIRTUE. Not only the quality of the reconstructed novel view should be realistic enough, but also the broadcasting and the view reconstruction should be done in real time (e.g., 25 frames/second). The real-time broadcasting issue has been addressed in [10].

Figure 1: The mock-up of one VIRTUE station where the real table is extended seamlessly into the virtual table in display. The full-size remote participants are rendered as arbitrary 2D video objects and their synthesized looks will change in line with the local participant's head position. The two cameras mounted on the left and top-left side of the screen provide two video streams for 3D analysis and view synthesis for the left viewer in the display; likewise for the two right-hand side cameras. The eye-to-eye contact, normal habitual hand gesturing, and gaze awareness are expected to be maintained.

This paper mainly deals with the view reconstruction issue, by following the aforementioned IBR approach.

Due to the symmetry in the system, we concentrate only on reconstructing a novel view of conferee C (*remote participant*) at site 1 (*local site*) from the information provided by cameras 1 and 2 at site 2 (*remote site*) and by the viewpoint of conferee A (*local participant*). Henceforth, cameras 1 and 2 at the remote site are referred to as the left (C_L) and the right (C_R) camera, respectively. These two cameras form a *stereo setup* for the view reconstruction.

Each time, at the remote site, the fixed stereo setup acquires two images. After segmentation, the pair of stereo views, containing only the remote participant without background, is broadcasted to the local site. Locally, based on the information about the stereo setup, the local display, and the *pose* (position and orientation) of the viewpoint of the local participant, these two views are used to reconstruct a novel view (telepresence) of the remote participant that is adapted to the current local viewpoint. The reconstructed novel view is then combined with a man-made uniform virtual environment in order to give the local participant the perception of being in a local conference with the remote participant.

The paper is organized as follows. In Section 2, an overview of work related to IBR is given. The processing chain of the whole VIRTUE system is introduced in Section 3. In Section 4, we conduct a theoretical analysis of the proposed view reconstruction algorithm. Furthermore, the algorithm's relationship with previous work is examined. Based on this analysis, the real-time implementation issues are considered in Section 5. Two possible schemes are investigated and compared with each other. The best option is chosen for real-time realization in VIRTUE. Experiments are done in Section 6 in order to demonstrate the quality and the speed of the implemented algorithm. Comparisons are made with two other well-studied approaches. Finally, we draw conclusions in Section 7.

2. RELATED WORK

Traditionally, the view reconstruction problem was solved by first constructing a 3D model out of the acquired information. The 3D model is then projected into a virtual camera by combining the texture mapping in order to obtain the desired view. The first step is usually named 3D reconstruction in computer vision. The second step is either called rendering in computer graphics or simply projection in computer vision. However, since real-time 3D reconstruction has been proved to be very difficult [11] and often ill-posed [12], IBR has intensively been studied as an alternative [7]. One big advantage of the IBR scheme is its image-size-proportional complexity (independent of the scene complexity). This property makes the stable real-time implementation possible [13].

Below we briefly address the IBR method. More general and detailed discussions can be found in [3, 7, 12].

Following the IBR approach, different kinds of representations of the 3D scene have been proposed.

Collection of pure 2D images

A collection of pure 2D images from either video sequences [14] or multiple cameras at different poses [15] is directly used for reconstructing novel views.

Layer representation

The whole 3D scene is decomposed into multiple layers according to different purposes. These layers provide faster and more robust rendering possibilities. Several kinds of layers have been proposed in the literature: motion consistent layers in [16], occlusion derived layers in [17, 18, 19], and planar layers in [20].

Plenoptic function

This function was introduced by Adelson and Bergen [21] in order to characterize the complete flow of light for all possible wavelengths, at any time, from each direction of a 3D scene, and at every pose. Afterwards many simplifications have been attempted on it [7]. No geometry information is needed to produce a new view, due to its complete description of the 3D scene. However, it inevitably relies on the oversampling, which is both time and labor consuming.

Panorama view

Hand-created or automatically generated view on a circular space from which a viewer can inspect his surroundings in all directions [22]. This technology has gained great success in real applications such as the QuickTime VR system [23]. But the position of the viewer is constrained to a fixed point and the capturing of the needed views should be done in a special way (all focal centers coincide with each other).

Mosaic

All images are warped into a uniform image coordinate system [24]. It can produce a very realistic immersive environment map. Recently, stereo cue starts to be added into it [25]. However, manual work cannot be avoided.

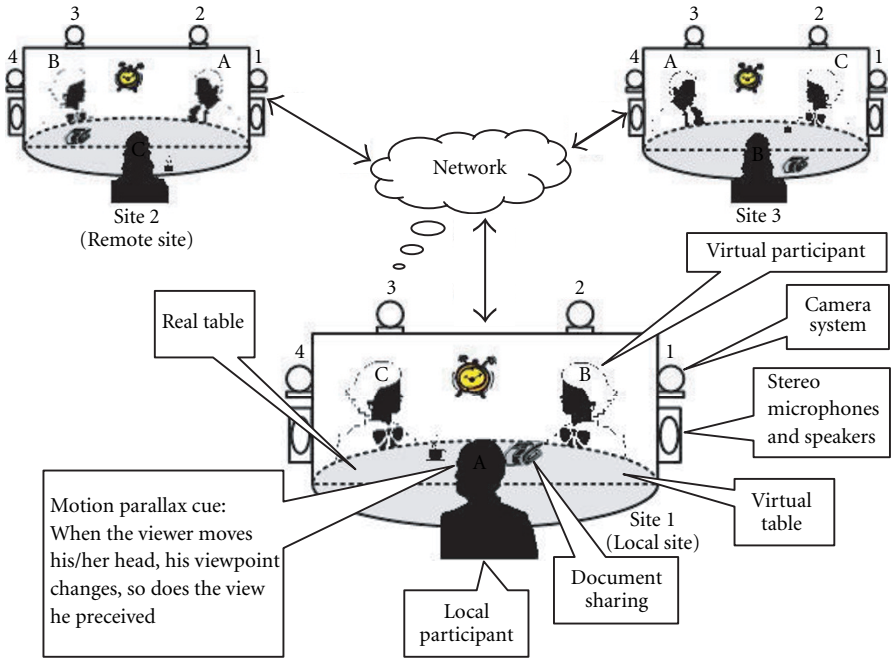


Figure 2: Illustration of the networked VIRTUE station in a three-way telepresence video conferencing session. Assuming accurate 3D scene analysis, the technique behind the novel view generation is the focus of the current paper.

Tour into a picture

By incorporating some human knowledge and observation about the 3D info embedded in an image, Youichi et al. [26] can even represent a 3D scene by one single picture (also see [27]). Moreover, by taking into account more constraints, a panorama can even be created from only one available view [28]. Again, manual work is needed to provide the rough 3D info.

All representations mentioned above have provided certain nice properties for reconstructing new views. However, except for the first two, all of them are very difficult, if not impossible, to be done automatically. The layered representation of 3D scenes offers a very nice opportunity to handle occlusions and redundancy in the stereo views. But it is rather complicated to segment the images according to certain consistency (e.g., motion) and later group the regions into different layers [18]. Therefore, the first representation with pure stereo views is chosen as the starting point for the VIRTUE view reconstruction.

In order to reconstruct a new view directly from a discrete set of images at different poses, four main approaches are currently followed.

Hybrid 3D models and image rendering

By combining simplified 3D structure models and realistic photographs, Debevec [8] has made amazing visual products (see http://www.debevec.org/). However, due to the use of 3D models, manual works cannot be avoided and thus the whole process (including modelling and rendering) is impossible to be automatic.

Visual hull

Starting from the visible silhouette information embedded in the given views, this approach approximates the geometry of the considered scene with either multiple visual hulls (the intersections of the silhouette come from all the given views) [29] or a single convex hull [30]. Performing all computations in the 2D image space instead of in the 3D space, as the traditional discrete volumetric representations did [31], real-time rendering may be guaranteed [29]. However, the visual hull of an object does not match the object's exact geometry and in particular it cannot represent concave surface regions [32].

Geometry driven methods

Geometric relations between known views are first completely recovered and then used for transferring known views to the new pose. Two schemes have been investigated in this category.

(1) *Using the fundamental matrix*: Laveau and Faugeras [33] by using fundamental matrix, tried to construct a new virtual view from N views. Each pixel in the desired novel view is mapped from a pixel in one known view based on the fundamental matrix between them. An attractive idea in [33] is the use of the ray tracing technique for handling the occlusion areas.

(2) *Using the trifocal tensor*: trying to avoid the singularity situations (the optical centers of the two known views and the virtual view are collinear) that may occur in the fundamental matrix, Avidan and Shashua [34] explored the usage of trifocal tensors (also called trilinearity) in view reconstruction.

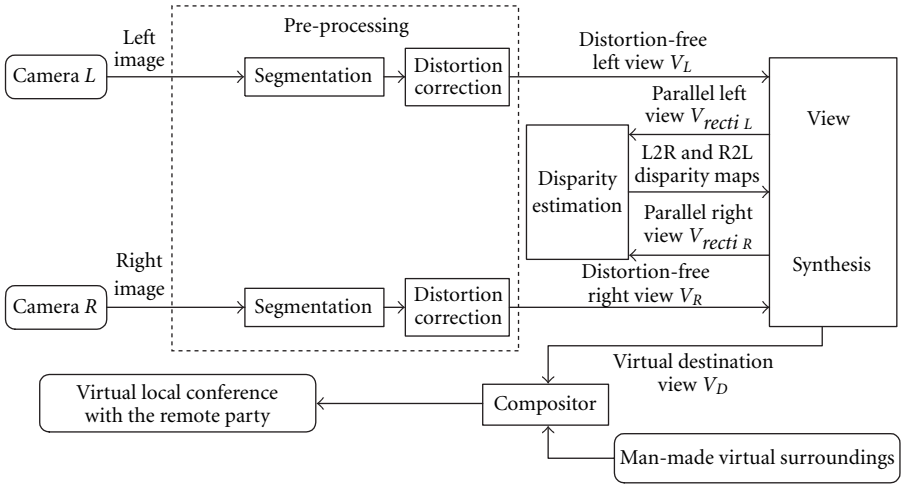


Figure 3: The processing chain for adapting the synthesized look of one participant in line with the viewpoint change of another participant. Based on a pair of stereo sequences, the "virtually" perceived view should be reconstructed and integrated seamlessly with the man-made uniform environment in real time.

The solution in [34] is theoretically perfect for stereo setup. It can geometric-validly recover arbitrary viewpoints from a discrete set of stereo images. Even more, by replacing the trifocal tensor with higher order tensors (e.g., quadrifocal tensor [35]), the geometry of the new view can directly be reconstructed from multiple (> 2) discrete known views.

However, there are two main disadvantages in this approach: (1) occlusion areas are very difficult to be taken into consideration; (2) the calculation of the trifocal tensor is rather complex. Although a simple equation is derived in [34] to transform the trifocal tensor in line with the viewpoint changing, all its 27 elements must be updated each time. This hinders the possibility of real-time processing.

Interpolation based methods

Chen and Williams [36] postulated a possible interpolation scheme for intermediate views from a stereo pair. Later this method was applied successfully in a vision system Quicktime-VR [23]. Seitz and Dyer [15, 37] investigated further along this line and proposed the physical-valid view interpolation method (pre-warping followed by interpolation plus post-warping). From this method arbitrary views (with valid geometry) on the baseline between a pair of stereo views can be reconstructed. Later on, Scharstein [12] added more freedoms into the possible *pose* of the desired novel view. Therein the only constraint is that the trifocal plane formed by the three focal centers of the two original views and the novel view should not intersect with the concerned 3D scene (see [12, page 147]).

One major advantage of this approach is that it performs very well when processing relative complex scenes.

Two other nice properties of this interpolation based approach that have not been addressed so far are the following.

Modular arrangement

This gives us much space to speed up the whole view reconstruction process by either rearranging processing components or simplifying the complex parts without affecting the quality of the final novel view. Thus a stable real-time realization may be guaranteed.

Parallel setup

By applying this setup computations may be simplified from 2D space to multiple 1D spaces, allowing parallel processing. Therefore, it can help to keep the latency very low. Further, this parallel setup allows us to handle the occlusion areas in a very intuitive way.

For our virtual conference system with telepresence perception, the interpolation based approach seems to be the most promising one. Our objective is to put this into a practical application which has seldom been studied except in rather constrained environments like the one described in [23]. Modular arrangement and parallel setup will be explored in Section 5.

In the following section, an introduction on the processing chain of the whole VIRTUE system will be given. Each stage in the process will be briefly addressed.

3. VIRTUE SYSTEM PROCESSING CHAIN

The intended processing chain of VIRTUE is depicted in Figure 3. It can be divided into four stages: pre-processing, disparity estimation, view reconstruction, and composition.

3.1. Pre-processing

The original images coming from the fixed stereo setup contain various kinds of imaging distortions [38] and background that should be removed or neglected. Some preprocessing is needed to get rid of these unnecessary information.

3.1.1 Segmentation

For the segmentation purpose, the background is first imaged by all employed cameras for several seconds. These images are processed to build a Gaussian model at every pixel for each camera by approximating the scene background with a texture surface. This Gaussian model is then used to distinguish a foreground pixel from a background pixel in the conference session. It has been shown that this kind of change detection scheme is very flexible and feasible for real-time implementation and thus able to meet the requirement of VIRTUE [39].

3.1.2 Distortion correction

Camera distortions [38] are compensated in this step to keep all following operations linear in the projective space. Camera distortions are nonlinear. However, if the "imaging distortion" definition is employed [38], then there is a one-to-many correspondence between the distorted image coordinates and the distortion-free coordinates. This enables us to construct beforehand, a backward mapping lookup table for distortion correction that is attractive for real-time implementation.

3.2. Disparity estimation

Based on a pair of rectified parallel views, the disparity maps (both left-to-right and right-to-left) [40] are estimated to represent implicitly the 3D information contained in them.

A hybrid block- and pixel-recursive approach has been chosen for VIRTUE. The main idea of this new algorithm is to combine the advantages of the block-recursive disparity estimator and a pixel-recursive optical flow estimator into one common scheme, leading to a fast, hybrid, recursive disparity estimation [41].

Although the disparity estimation is very important for the quality of the overall result, it is not within the scope of this paper. For a detailed discussion on the disparity estimator used for VIRTUE, see [42].

3.3. View reconstruction

With a pair of pre-processed "clean" views V_L (the left view) and V_R (the right view), the view reconstruction will generate a virtual view V_D in accordance to the actual viewpoint of the local participant. This is the topic of this paper and will be discussed in detail.

3.4. Composition

In the compositor, the 3D model of a man-made uniform environment map is combined with the reconstructed virtual view of the remote participants in order to construct the final virtual conferencing environment. This composition can efficiently be realized using current off-the-shelf graphics card with texture handling capability [2].

The final composite view is put on a life-size display to give the local participant the impression of being in a conference room with the other participants.

4. MULTI-STEP VIEW RECONSTRUCTION

In the above described VIRTUE system, the view reconstruction stage plays a vital role. In this section, we generalize the theory of the interpolation-based IBR to accommodate arbitrary novel viewpoints. We base this generalization on the known camera geometry. We call the generalized method *multi-step view reconstruction*. We will show that, without assuming any constraints on either the 3D scene or the camera geometry, our multi-step view reconstruction maintains the geometry validity of the reconstructed view.

4.1. Problem analysis

In brevity, our problem is to reconstruct an arbitrary novel view V_D from a pair of stereo views V_L (the left view) and V_R (the right view), which came from two fixed-*pose* cameras C_L and C_R, respectively. Correspondingly, we can treat V_D as coming from a virtual camera C_D (in correspondence with the current viewpoint, its geometry can be calculated from the system configuration, the stereo setup, the local display, and the pose of the viewpoint). So the problem to be solved is simply: *to compute V_D as coming from C_D, given V_L from C_L and V_R from C_R*.

Considering a single 3D-scene point W, the problem can be further simplified as: *given P^L (the projection of W into C_L) and/or (in case of occlusion) P^R (the projection of W into C_R), compute P^D (the projection of W into C_D)*.

Two things should be noticed here: (1) we only address a fully calibrated situation [4], which means that the geometry details of C_L, C_R, and C_D are all known in advance with high accuracy; (2) distortions due to imaging system have been removed by pre-processing. Thus only linear relations will be taken into account henceforth.

4.2. Notation

We denote by V_* the 2D view generated from the camera C_*, where $*$ stands for, for example, L, R, D, *recti* L, *recti* R, X, Y, Z. P^* is the projection of the 3D-scene point W into C_*. The intrinsic parameters of C_* are f_* (*focal length*), s_{x*} and s_{y*} (the effective pixel distance in the horizontal (x-axis) and vertical (y-axis) direction, determined by the sampling rates for Vidicon cameras or the sensitive distance for CCD or CID cameras, respectively), x_{0*} and y_{0*} (the pixel coordinates of the principal point with respect to the image plane coordinate system). The extrinsic parameters of C_* are $\mathbf{R}_{c*}$ and $\mathbf{t}_{c*}$ (rotation matrix and translation vector of the CCS (camera coordinate system) with respect to the WCS (world coordinate system)). The projection matrix is $\tilde{\mathbf{P}}_* = \tilde{\mathbf{K}}_* \tilde{\mathbf{E}}_*$, where $\tilde{\mathbf{K}}_*$ is the intrinsic matrix and $\tilde{\mathbf{E}}_*$ is the extrinsic matrix [38].

In the CCS of C_*, the coordinate of W is $\mathbf{w}^*$ (the projective correspondence is $\tilde{\mathbf{w}}^*$). In the WCS, the coordinate of W is $\mathbf{w}$ (the projective correspondence is $\tilde{\mathbf{w}}$). The coordinate of P^* in view V_* is $\mathbf{p}^*$ (the projective correspondence is $\tilde{\mathbf{p}}^*$).

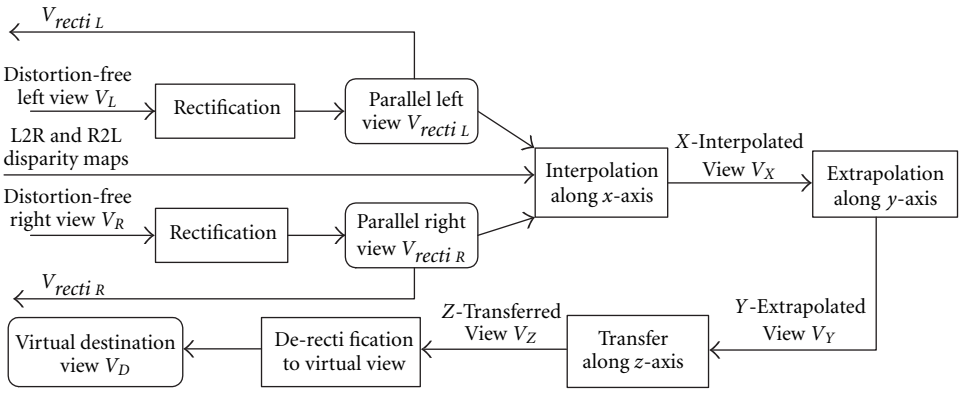


Figure 4: The multi-step view synthesis framework for VIRTUE. Multiple separate steps work together to eliminate three major differences between the final novel view V_D and the two original views V_L and V_R: (1) photometric differences such as focal length and aspect ratio etc.; (2) position in 3D space (x, y, z); (3) orientation.

Further

$$\mathbf{t}_{c*} = \begin{bmatrix} x_{c*} \\ y_{c*} \\ z_{c*} \end{bmatrix},$$

$$\mathbf{p}^* = \begin{bmatrix} x_p^* \\ y_p^* \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}, \quad \mathbf{w}^* = \begin{bmatrix} x_w^* \\ y_w^* \\ z_w^* \end{bmatrix}, \tag{1}$$

$$\tilde{\mathbf{p}}^* = \begin{bmatrix} x_p^* \\ y_p^* \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{w}} = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{w}}^* = \begin{bmatrix} x_w^* \\ y_w^* \\ z_w^* \\ 1 \end{bmatrix}.$$

Then we can write [38]

$$\begin{aligned} \lambda^* \tilde{\mathbf{p}}^* &= \tilde{\mathbf{P}}_* \tilde{\mathbf{w}} = \tilde{\mathbf{K}}_* \tilde{\mathbf{E}}_* \tilde{\mathbf{w}} \\ &= \begin{bmatrix} -\dfrac{f_*}{s_{x*}} & 0 & x_{0*} \\ 0 & -\dfrac{f_*}{s_{y*}} & y_{0*} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{c*}^T & -\mathbf{R}_{c*}^T \mathbf{t}_{c*} \end{bmatrix} \tilde{\mathbf{w}}, \end{aligned} \tag{2}$$

where λ^* is the depth of the point W in the CCS of C_*.

The line connecting the focal points of C_L and C_R is usually called baseline $\mathbf{b}$. Its length is simply noted as b.

4.3. Multi-step view reconstruction process

Without loss of generality, we can specifically select the WCS to be such that

$$\mathbf{t}_{cL} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{t}_{cR} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{t}_{cD} = \begin{bmatrix} x_{cD} \\ y_{cD} \\ z_{cD} \end{bmatrix}. \tag{3}$$

This means that the x-axis of the WCS is on the baseline $\mathbf{b}$ and its direction is from C_R to C_L. The origin of the WCS stays at the middle point of $\mathbf{b}$. And the unit of the WCS is $b/2$.

In the general case, the multi-step view reconstruction process can be divided into five steps (Figure 4).

(1) *Rectification*. We transform the stereo views V_L and V_R into a pair of new views $V_{recti\,L}$ and $V_{recti\,R}$, respectively. The two virtual cameras $C_{recti\,L}$ and $C_{recti\,R}$, which generate those two new views, are parallel to each other and share the same image plane. This process is known as stereo rectification [43]. It is intended to eliminate the photometric and orientation differences between the two source cameras, to simplify the correspondence estimation into a 1D search problem along the scan line and at the same time to provide parallel processing possibility for latter steps.

(2) *X-interpolation*. Given the necessary disparity information, the two parallel views $V_{recti\,L}$ and $V_{recti\,R}$ are combined by either interpolation [44] or extrapolation to produce another parallel view V_X. The corresponding camera C_X is located at $[x_{cD} \;\; 0 \;\; 0]$ with the same rotation and intrinsic parameters [4] as $C_{recti\,L}$ and $C_{recti\,R}$. Through this step, the x position difference from the known views $V_{recti\,L}$ and $V_{recti\,R}$ to the final view V_D is eliminated.

(3) *Y-extrapolation*. The X-interpolated view V_X is extrapolated [12] by pixel shifting in the Y direction to produce the view V_Y, which is coming from a virtual camera C_Y located at $[x_{cD} \;\; y_{cD} \;\; 0]$ with the same rotation and intrinsic parameters as C_X. Through this step, the y position difference from V_X to the final view V_D is eliminated.

(4) *Z-transfer*. The Y-extrapolated view V_Y is transferred along the Z direction to generate either a "closer" or "further" look V_Z. In the same way, the corresponding camera C_Z is located at $[x_{cD} \;\; y_{cD} \;\; z_{cD}]$ with the same rotation and intrinsic parameters as C_Y. Finally, the z position difference from V_Y to the final view V_D is eliminated.

(5) *De-rectification*. The Z-transferred view V_Z is rotated and scaled to get the final view V_D.

In the following subsections, we go through these five steps and discuss in detail all the formulas involved. Note that through the whole transformation process the geometric validity is maintained.

4.4. Rectification

To obtain a parallel configuration, C_{rectiL} and C_{rectiR} should have the same set of intrinsic parameters and the same orientation in space [12]. Further, in order to simplify the notation we may assume, without loss of generality, that C_{rectiL} and C_{rectiR} have the same orientation as the WCS

$$\tilde{\mathbf{K}}_{rectiL} = \tilde{\mathbf{K}}_{rectiR} = \tilde{\mathbf{K}} = \begin{bmatrix} -\frac{f}{s_x} & 0 & x_0 \\ 0 & -\frac{f}{s_y} & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_{c\,rectiL} = \mathbf{R}_{c\,rectiR} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

where the intrinsic transformation matrix $\tilde{\mathbf{K}}$ [38] and its associated intrinsic parameters are employed just for notation simplicity.

Based on the above notations, as both $\tilde{\mathbf{K}}_L$ and $\tilde{\mathbf{K}}_R$ are invertible, we have [45]

$$\begin{aligned} \lambda^{rectiL}\tilde{\mathbf{p}}^{rectiL} &= \lambda^{L}\tilde{\mathbf{K}}\mathbf{R}_{cL}\left(\tilde{\mathbf{K}}_L\right)^{-1}\tilde{\mathbf{p}}^{L} + \tilde{\mathbf{K}}\left(\mathbf{t}_{cL} - \mathbf{t}_{c\,rectiL}\right), \\ \lambda^{rectiR}\tilde{\mathbf{p}}^{rectiR} &= \lambda^{R}\tilde{\mathbf{K}}\mathbf{R}_{cR}\left(\tilde{\mathbf{K}}_R\right)^{-1}\tilde{\mathbf{p}}^{R} + \tilde{\mathbf{K}}\left(\mathbf{t}_{cR} - \mathbf{t}_{c\,rectiR}\right). \end{aligned} \tag{5}$$

However, we do not know the depth of the point W yet. In order to make the above two equations independent of the depth information, we have only one choice [43], which is to set

$$\mathbf{t}_{c\,rectiL} = \mathbf{t}_{cL}, \qquad \mathbf{t}_{c\,rectiR} = \mathbf{t}_{cR}. \tag{6}$$

In this case, we can construct two homographies [4] $\mathbf{T}_L$ and $\mathbf{T}_R$ such that

$$\tilde{\mathbf{p}}^{rectiL} = \frac{\lambda^{L}}{\lambda^{rectiL}}\mathbf{T}_L\tilde{\mathbf{p}}^{L}, \qquad \tilde{\mathbf{p}}^{rectiR} = \frac{\lambda^{R}}{\lambda^{rectiR}}\mathbf{T}_R\tilde{\mathbf{p}}^{R}, \tag{7}$$

where

$$\mathbf{T}_L = \tilde{\mathbf{K}}\mathbf{R}_{cL}\left(\tilde{\mathbf{K}}_L\right)^{-1}, \qquad \mathbf{T}_R = \tilde{\mathbf{K}}\mathbf{R}_{cR}\left(\tilde{\mathbf{K}}_R\right)^{-1}. \tag{8}$$

It should be noted that here we need not know the exact value of $\lambda^L/\lambda^{rectiL}$ and $\lambda^R/\lambda^{rectiR}$, because they have been determined implicitly by the definition that the third components of $\tilde{\mathbf{p}}^{rectiL}$, $\tilde{\mathbf{p}}^{L}$, $\tilde{\mathbf{p}}^{rectiR}$, and $\tilde{\mathbf{p}}^{R}$ are all 1.

After the rectification in (7) is performed, it can be verified that $\mathbf{p}^{rectiL}$ and $\mathbf{p}^{rectiR}$ have the same y coordinates but different x coordinates [43]. Therefore, we can compute, using a disparity estimation algorithm [40], two disparity maps D_{LR} (left-to-right, based on the view V_{rectiL}, the disparity value at P^{rectiL} is denoted as d_p^{LR}) and D_{RL} (right-to-left, based on the view V_{rectiR}, the disparity value at P^{rectiR} is denoted as d_p^{RL}) where

$$d_p^{LR} = x_p^{rectiR} - x_p^{rectiL}, \qquad d_p^{RL} = x_p^{rectiL} - x_p^{rectiR}. \tag{9}$$

4.5. X-interpolation

Because C_X, C_{rectiL}, and C_{rectiR} are parallel to each other with the x-axis on the baseline $\mathbf{b}$, the projections of W into them have the same y-coordinates but different x-coordinates

$$y_p^X = y_p^{rectiL} = y_p^{rectiR} = -\frac{f}{s_y}\cdot\frac{y_w}{z_w} + y_0, \tag{10}$$

$$x_p^{rectiL} = -\frac{f}{s_x}\cdot\frac{x_w - 1}{z_w} + x_0, \tag{11}$$

$$x_p^{rectiR} = -\frac{f}{s_x}\cdot\frac{x_w - (-1)}{z_w} + x_0, \tag{12}$$

$$x_p^X = -\frac{f}{s_x}\cdot\frac{x_w - x_{cD}}{z_w} + x_0. \tag{13}$$

By subtracting (11) from (12), we obtain

$$z_w = -\frac{2f}{s_x\left(x_p^{rectiR} - x_p^{rectiL}\right)} = -\frac{2f}{s_x d_p^{LR}} = \frac{2f}{s_x d_p^{RL}}. \tag{14}$$

By subtracting (12) from (13), we get

$$x_p^X = x_p^{rectiR} + \frac{f}{s_x}\cdot\frac{x_{cD} + 1}{z_w}. \tag{15}$$

By substituting (14) into (15) we get

$$x_p^X = \underbrace{\left(x_p^{rectiL} + x_p^{rectiR}\right)/2}_{\text{Middle view}} - \underbrace{\left(x_p^{rectiR} - x_p^{rectiL}\right)}_{\text{Disparity}}\cdot\frac{x_{cD}}{2}. \tag{16}$$

Equation (16) is just the interpolation equation derived by Seitz and Dyer in [44]. However, it should be noted that, in order to obtain (16), we did not add the ordering constraint [46].

By rearranging (16) we have

$$x_p^X = x_p^{rectiL} + \frac{1 - x_{cD}}{2}d_p^{LR} \tag{17}$$

and/or (in case of occlusion)

$$x_p^X = x_p^{rectiR} + \frac{1 + x_{cD}}{2}d_p^{RL}. \tag{18}$$

To keep the 3D info that was recovered from the stereo data, a new disparity map D_X is constructed wherein (based on the view V_X, the disparity value at P^X is denoted as d_p^X)

$$d_p^X = -\frac{d_p^{RL}}{2} = -\frac{f}{s_x z_w} \qquad \text{or} \qquad d_p^X = \frac{d_p^{LR}}{2} = -\frac{f}{s_x z_w}. \tag{19}$$

4.6. Y-extrapolation

As C_X and C_Y are parallel to each other with aligned y-axes, the projections of W into them should have the same x-coordinates but different y-coordinates

$$x_p^Y = x_p^X = -\frac{f}{s_x}\cdot\frac{x_w - x_{cD}}{z_w} + x_0, \tag{20}$$

$$y_p^X = -\frac{f}{s_y} \cdot \frac{y_w}{z_w} + y_0, \tag{21}$$

$$y_p^Y = -\frac{f}{s_y} \cdot \frac{y_w - y_{cD}}{z_w} + y_0. \tag{22}$$

By subtracting (21) from (22) we have

$$y_p^Y = y_p^X - \frac{f}{s_y} \cdot \frac{y_{cD}}{z_w} = y_p^X - y_{cD} \cdot \frac{s_x}{s_y} \cdot d_p^X. \tag{23}$$

Scharstein has derived similar results as (17), (18), and (23) (see [12, page 45]). One major difference is that Scharstein neglected the functioning of s_x and s_y on the extrapolation.

To retain the 3D information, another disparity map D_Y is constructed wherein (based on the view V_Y, the disparity value at P^Y is denoted as d_p^Y)

$$d_p^Y = -\frac{f}{s_x z_w}. \tag{24}$$

Although it looks like we repeatedly transfer the same info here, it is really necessary because the position information (content of $\mathbf{p}^*$ ($*$ stands for, e.g., X, Y)) is implicitly encoded in the view itself.

4.7. Z-transfer

When we move along the depth direction, both x and y coordinates are changed accordingly

$$x_p^Z = -\frac{f}{s_x} \cdot \frac{x_w - x_{cD}}{z_w - z_{cD}} + x_0, \tag{25}$$

$$y_p^Z = -\frac{f}{s_y} \cdot \frac{y_w - y_{cD}}{z_w - z_{cD}} + y_0, \tag{26}$$

by substituting (20) and (24) into (25) we get

$$x_p^Z = (x_p^Y - x_0) \cdot \frac{f}{f + s_x z_{cD} d_p^Y} + x_0. \tag{27}$$

Similarly for y_p^Z

$$y_p^Z = (y_p^Y - y_0) \cdot \frac{f}{f + s_x z_{cD} d_p^Y} + y_0. \tag{28}$$

4.8. De-rectification

Up to this stage, the view has been moved to the destination position $\mathbf{t}_{cD} = [x_{cD} \;\; y_{cD} \;\; z_{cD}]$ with orientation $\left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right]$. In C_Z and C_D, we have

$$\lambda^Z \tilde{\mathbf{p}}^Z = \tilde{\mathbf{K}}(\mathbf{w} - \mathbf{t}_{cD}), \qquad \lambda^D \tilde{\mathbf{p}}^D = \tilde{\mathbf{K}}_D \mathbf{R}_{cD}^T (\mathbf{w} - \mathbf{t}_{cD}). \tag{29}$$

As both $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{K}}_D \mathbf{R}_{cD}^T$ are invertible [45], we can easily write down for the de-rectification step

$$\lambda^D \tilde{\mathbf{p}}^D = \lambda^Z \mathbf{T}_D \tilde{\mathbf{p}}^Z, \tag{30}$$

where

$$\mathbf{T}_D = \tilde{\mathbf{K}}_D \mathbf{R}_{cD}^T (\tilde{\mathbf{K}})^{-1}. \tag{31}$$

This is similar to the rectification (see (31)).

4.9. Summary

The multi-step reconstruction process, as described in this section, is a generalization of the interpolation-based approach. It combines existing methods and generalizes them to accommodate arbitrary novel viewpoints.

(1) In the case of parallel setup, in order to obtain an intermediate virtual view, Chen and Williams [36] proposed a view interpolation method that is the same as step 2 here.

(2) In the arbitrary stereo configuration case, in order to maintain the geometric validity of the reconstructed view, Seitz and Dyer [44] added the rectification (step 1) and the de-rectification (step 5) into the view interpolation system.

(3) In order to get a novel view at arbitrary camera pose confined in a plane, Scharstein [12] added step 3.

(4) Based on the complete camera geometry, we formally generalize the interpolation-based approach for arbitrary stereo configurations and novel views at arbitrary poses.

5. IMPLEMENTATION CONSIDERATIONS

To make the whole view reconstruction process suitable for real-time applications such as VIRTUE, two critical issues should carefully be addressed: occlusion handling and computation efficiency. In this section, these two issues will be elaborated along with a detailed analysis of the realization.

Instead of reconstructing a novel view in a complex single step like the Trifocal tensor approach in [34], the multi-step scheme decomposes the whole view reconstruction process into two parts: one part (the rectification and the de-rectification) deals with the photometric and orientation difference and the other (the middle three steps) handles the position component. First we discuss the implementation issues for these two parts. Fast processing potentials will be explored in detail. For the translation handling part two simplifying possibilities are proposed and compared with each other. The best is chosen for VIRTUE. After this, the view reconstruction process will be integrated with other stages within the intended VIRTUE framework.

To show the process in each step, we use a stereo pair (Figure 5, image size 720×576) from the real VIRTUE setup as an example.

5.1. Photometric and orientation difference handling

Rectification is very important for two main reasons. First, two views can be interpolated easily if they came from two cameras that are parallel to each other and these two cameras share the same image plane. Second, the parallel setup significantly facilitates the correspondence estimation task, in which case the correspondence search in 2D space is simplified along 1D scanlines [40].

(a) Original left view V_L.

(b) Original right view V_R.

FIGURE 5: An example stereo pair from the cameras 1 and 2 of the VIRTUE setup as shown in Figure 10. Notice the large occlusion areas around the right arm of the participant and the converging effect of the setup.

(a) Rectified left view V_L.

(b) Rectified right view V_R.

FIGURE 6: The stereo views are rectified without considering the epipolar geometry. To retain all the information (including the black area) from the original views, they may have different sizes. In this stereo setup, the size of the rectified left view is 1012×955 and that of the rectified right view is 848×811.

From (7), since $\tilde{\mathbf{K}}$, $\mathbf{R}_{cL}$, and $\mathbf{R}_{cR}$ are all invertible, we can write down

$$\tilde{\mathbf{p}}^L = \frac{\lambda^{rectiL}}{\lambda^L}\mathbf{T}_L^{-1}\tilde{\mathbf{p}}^{rectiL}, \qquad \tilde{\mathbf{p}}^R = \frac{\lambda^{rectiR}}{\lambda^R}\mathbf{T}_R^{-1}\tilde{\mathbf{p}}^{rectiR}. \tag{32}$$

This means that for every pixel in the rectified view, there exists a corresponding pixel in the corresponding original view and it enables us to employ the backward mapping technique (see Appendix B). The backward mapping can be realized very fast by a simple lookup table, which can be constructed beforehand and it may stay the same as long as the setup is fixed. Two fast ways to assign intensities in the rectified view are zero-order (nearest neighbor) and first-order (bilinear interpolation). Through extensive experiments we found that for the VIRTUE setup the nearest neighbor is three times faster than the bilinear interpolation, while the image quality generated by both is comparable to each other. Thus we adopt the nearest neighbor for the rectification process.

During the rectification, in order to retain all information we have from the original images and at the same time to keep the rectified image size as small as possible, the rectified image should be shifted and cut to keep only the useful information. Further, due to the epipolar constraint [4], two constraints should be applied:

(a) Rectified left view V_L.

(b) Rectified right view V_R.

FIGURE 7: The stereo views are rectified considering the epipolar geometry. Thus two constraints can be applied to make them have the same smallest possible size 1012×811.

(1) the width of the rectified stereo pair should be the larger one of the widths of the two rectified images (left and right);
(2) the height of the rectified stereo pair should be the smaller one of the heights of the two rectified images (left and right).

Figure 6 shows the rectification applied to the left and right view separately. The size of the rectified left view is 1012×955 and that of the rectified right view is 848×811.

By applying the two constraints mentioned above, we get a pair of rectified stereo pair of size 1012×811 (shown in Figure 7).

Since the de-rectification is similar to the rectification process (compare (30) with (7)), it can be implemented by a nearest neighbor based lookup table as well

$$\tilde{\mathbf{p}}^Z = \frac{\lambda^D}{\lambda^Z}\mathbf{T}_D^{-1}\tilde{\mathbf{p}}^D. \tag{33}$$

From Section 4.4, we note that there are two freedoms in choosing the geometric properties of C_{rectiL} and C_{rectiR}: (1) the intrinsic parameter transformation matrix $\tilde{\mathbf{K}}$; (2) the z-axis of the CCSs (implicitly the WCS). The intrinsic transformation matrix $\tilde{\mathbf{K}}$ can be determined by the final display, in this case C_D. In our current implementation we choose $\tilde{\mathbf{K}} = (\tilde{\mathbf{K}}_L + \tilde{\mathbf{K}}_R)/2$ and the z-axis of C_{rectiL} and C_{rectiR} to be at the middle position of the z-axis of C_L and C_R (for detailed equations involved, see [38, Appendix A.2.5]). The advantage of this choice is that the rectification homographies T_L and T_R are fixed for a fixed stereo setup. Thus two fixed lookup tables can be built beforehand.

5.2. Position difference handling

Although translating the view in three separate steps along three orthogonal directions sounds straightforward, there is one major difficulty: in the Z-transfer step, at each pixel, we have to calculate a different factor c_p

$$c_p = \frac{f}{f - s_x z_{cD} d_p^Y}. \tag{34}$$

This makes the Z-transfer time-consuming, not only because of the computation load but also of the random 2D memory access. In addition, (27) and (28) can only be implemented by the forward mapping technique, in which case the remaining 2D holes have to be filled in afterwards, which, again, is computational expensive. (Note that although there exist forward mapping techniques that do not leave holes [47], we found out that real-time implementation for them is even more difficult, if not impossible.)

Fortunately, there exist two schemes that can be explored here to ease the processing burden. One or the other of these two schemes could be employed, according to different situations.

5.2.1 Scheme one: one optimal uniform scale factor

One scheme is to replace c_p by a constant $\overline{c}$ for every pixel in the destination view. A possible choice is

$$\overline{c} = \frac{f}{f - s_x z_{cD} \overline{d}}, \tag{35}$$

where $\overline{d}$ is a constant but optimal disparity value. "Optimal" here means optimal with respect to the minimal perspective distortion.

In this case, (27) and (28) will become

$$\begin{aligned} x_p^Z &= \overline{c} \cdot x_p^Y + (1 - \overline{c}) \cdot x_0, \\ y_p^Z &= \overline{c} \cdot y_p^Y + (1 - \overline{c}) \cdot y_0. \end{aligned} \tag{36}$$

A nice property of the two equations above is that backward mapping (see Appendix B) can be applied if we rewrite them into

$$\begin{aligned} x_p^Y &= \frac{x_p^Z - (1 - \overline{c}) \cdot x_0}{\overline{c}}, \\ y_p^Y &= \frac{y_p^Z - (1 - \overline{c}) \cdot y_0}{\overline{c}}, \end{aligned} \tag{37}$$

where $\overline{c}$ can be computed in advance by locating $\overline{d}$ as the disparity corresponding to the peak in the disparity histogram.

Thus

$$\tilde{\mathbf{p}}^Y = \mathbf{Z}_Z \tilde{\mathbf{p}}^Z, \tag{38}$$

where

$$\mathbf{Z}_Z = \begin{bmatrix} \frac{1}{\overline{c}} & 0 & \left(1 - \frac{1}{\overline{c}}\right) \cdot x_0 \\ 0 & \frac{1}{\overline{c}} & \left(1 - \frac{1}{\overline{c}}\right) \cdot y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{39}$$

As this simplification in fact changes the Z-transfer step into a zooming effect of the camera, we call it *Z-zooming*, to be distinguished from the general Z-transfer process.

Of course, the simplification from Z-transfer to Z-zooming will introduce some perspective distortions. However, from our extensive experiments on the real VIRTUE setup, we did not find any noticeable artifacts yet. The quality of the results from the two cases are comparable. The visual quality of the result from Z-zooming is even better, since with the optimal disparity we get rid of minor disparity artifacts (see Figure 8).

(a) Z-transfer. (b) Z-zooming.

Figure 8: Both the Z-transfer and the Z-zooming are applied on the same Y-extrapolated view. In the Z-transfer, each pixel has a different transform coefficient. While in the Z-zooming, all pixels are transformed by a uniform optimal scaling factor. The uniform factor is chosen to minimize the possible perspective distortions.

The above simplification holds well for the current VIRTUE setup. In other circumstances, for example, when the depth difference in the scene is too large, we can segment the view into several layers by analyzing the disparity histogram in advance. Then we apply the above technique to each layer separately. Finally, a warping technique [47] can be applied to stitch them together to get the "Z-transferred" view. This "multiple scale factors" proposal would produce more geometrically accurate novel views than using one single scale factor. But inevitably it would involve more computations.

5.2.2 Scheme two: two 1D transfer steps

On the other hand, if we take a closer look at (27) and (28), we note that the two calculations in x and y are independent from each other. Thus it is possible to integrate the calculations in (27) into the X-interpolation step and the calculations in (28) into the Y-extrapolation step. If so, the translation difference handling part can be reduced from three steps to two 1D operation steps:

(1) X-transfer

$$x_p^X = \left(x_p^{rectiL} + \frac{1 - x_{cD}}{2} d_p^{LR} - x_0\right) \cdot \frac{f}{f + s_x z_{cD} \cdot d_p^{LR}/2} + x_0 \tag{40}$$

or

$$x_p^X = \left(x_p^{rectiR} + \frac{1 + x_{cD}}{2} d_p^{RL} - x_0\right) \cdot \frac{f}{f - s_x z_{cD} \cdot d_p^{RL}/2} + x_0. \tag{41}$$

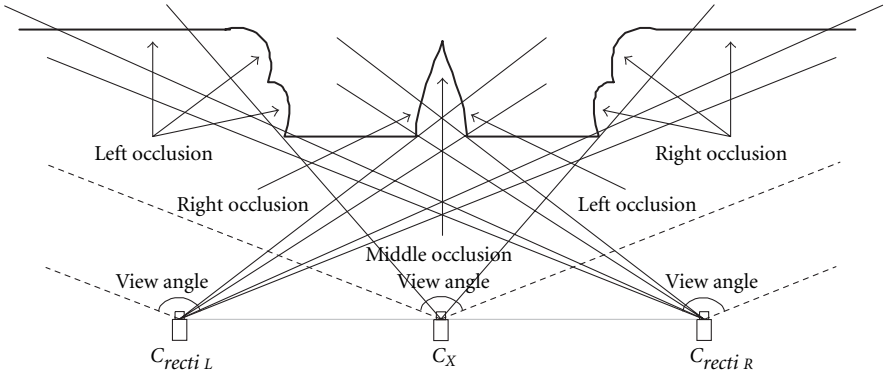


Figure 9: Top view about three parallel cameras imaging a 3D scene. The bold curve indicates the scene surface from the viewpoint of the three cameras $C_{recti\,L}$, $C_{recti\,R}$, and C_X. The left occlusion areas appear in $V_{recti\,L}$ but not in $V_{recti\,R}$. The right occlusion areas appear only in $V_{recti\,R}$ but not in $V_{recti\,L}$. The complete 3D areas appear both in $V_{recti\,L}$ and in $V_{recti\,R}$. These three types of info may or may not appear in V_X. While the middle occlusion areas only appear in V_X but neither in $V_{recti\,L}$ nor in $V_{recti\,R}$.

(2) Y-transfer

$$y_p^Y = \left(y_p^X + y_{cD} \cdot \frac{s_x}{s_y} \cdot d_p^X - y_0\right) \cdot \frac{f}{f - s_x z_{cD} d_p^X} + y_0. \quad (42)$$

In this manner, only two 1D operations are needed to transfer the rectified stereo views into the destination *pose*. Compared with the 2D operator like the original Z-transfer, the 1D operations have two advantages: (1) straight-forward hole filling; (2) more regular memory access (combining information only from the same row or column).

5.2.3 Comparison

Although the "two 1D transfer steps" simplification is geometrically valid and sounds more promising, the adoption of one optimal uniform scale factor is more practical for a real-time application mainly due to three reasons:

(1) as the disparity range in a stereo setup is limited, three small lookup tables can be built up in advance for $((1-x_{cD})/2)d_p^{LR}$, $((1+x_{cD})/2)d_p^{RL}$, and $y_{cD} \cdot s_x/s_y \cdot d_p^X$ in (17), (18), and (23), respectively. While this is impossible for (40), (41), and (42).

(2) The simplified Z-zooming function can be integrated together with the de-rectification step as one backward mapping step without adding any additional computation load such as

$$\tilde{\mathbf{p}}^Y = \left(\frac{\lambda^D}{\lambda^Z}\mathbf{Z}_Z\mathbf{T}_D^{-1}\right)\tilde{\mathbf{p}}^D. \quad (43)$$

(3) The visibility issue[1] could be solved implicitly in the X-interpolation and Y-extrapolation operations but it is rather complicated in the X-transfer and the Y-transfer [48].

Based on these three considerations, the "one optimal uniform scale factor" scheme is chosen for the current VIRTUE realization. The "multiple scale factors" proposal will be investigated for the further extension. The "two 1D transfer steps" possibility is reserved for a future improvement when more advanced hardware are available.

5.2.4 Realization using one optimal uniform scale factor

Considering the occlusion and visibility issues, our current realization based on the "one optimal uniform scale factor" scheme can be summarized as follows.

X-interpolation

From (17) and (18), it seems that we only need one disparity map plus one view to construct the X-interpolated view V_X perfectly. However, in practice, we always suffer from the occlusion problem[2] [10].

In general, four cases in the 3D scene can be distinguished (Figure 9) [10]:

(i) *Complete 3D info*. The info that can be viewed in $C_{recti\,L}$, $C_{recti\,R}$, and C_X.

(ii) *Left occlusion*. The parts of the view that are visible in the left camera but not in the right camera.

(iii) *Right occlusion*. The parts of the view that are visible in the right camera but not in the left camera.

(iv) *Middle occlusion*. The parts of the view that are only visible in the X-interpolated view.

Suppose the estimated disparity maps contain the correct (pseudo) disparity in occluded regions. Then we can generate the X-interpolated view V_X by

(1) For complete 3D info: Apply either (17) or (18).

[1] Multiple scene points may map to the same pixel in a view, while only the nearest (to the camera) one contributes intensity to that pixel.

[2] A scene point may appear, due to the visibility issue, either only in the left view, or only in the right view. So for each pixel in the destination view, the intensity may be contributed by a pixel in either the left view or in the right view. But we do not know this information in advance. It is revealed only after the forward mapping has been applied both on the left and the right views.

(2) For left occlusion areas: Use (17).

(3) For right occlusion areas: Employ (18).

(4) For middle occlusion areas: No info is available from either $V_{recti\,L}$ or $V_{recti\,R}$. We have to approximate them, for example, by either nearest-neighbor or linear interpolation.

Assuming that the light condition between the left and the right view does not change too much, four cases can be considered to facilitate the computation:

(i) $x_{cD} \geq 1$: fetch the intensity of each pixel in V_X from the corresponding pixel in $V_{recti\,L}$;

(ii) $0 \leq x_{cD} < 1$: fetch them from $V_{recti\,L}$, except for the right occlusion areas, where the information should come from $V_{recti\,R}$;

(iii) $-1 < x_{cD} < 0$: fetch them from $V_{recti\,R}$, except for the left occlusion areas, where the information should come from $V_{recti\,L}$;

(iv) $x_{cD} \leq -1$: fetch the intensity of each pixel in V_X from the corresponding pixel in $V_{recti\,R}$.

To avoid repetition, we give below only the implementation description for the second case. The realization for the other three cases can be derived in a similar way.

(1) Two lookup tables are constructed for $((1-x_{cD})/2)d_p^{LR}$ and $((1+x_{cD})/2)d_p^{RL}$, respectively, based on the predetermined disparity range.

(2) From (17), we generate a disparity map D_{XL} which is based on the view V_X. The relations are $d_p^{XL} = d_p^{LR}/2$ and $x_p^X = x_p^{recti\,L} + ((1-x_{cD})/2)d_p^{LR}$. For solving the visibility problem, this step processes pixels from left to right [48].

(3) From (41), we generate a disparity map D_{XR} which is based on the view V_X. The relations are $d_p^{XR} = -d_p^{RL}/2$ and $x_p^X = x_p^{recti\,R} + ((1+x_{cD})/2)d_p^{RL}$. In order to solve the visibility problem, this step processes pixels from right to left [48].

(4) D_{XL} and D_{XR} are integrated to generate D_X.

(5) Consider now D_X and each scanline, for every hole area where no value has been assigned (in fact, they are middle occlusions), if the two ends of the segment are both either left available or right available, then it is filled in by linear interpolation. Otherwise it is filled in by nearest neighbor.

(6) We synthesize the view V_X by the backward mapping technique based on D_X, $V_{recti\,L}$, and $V_{recti\,R}$.

Here we first implicitly reconstruct the geometry of the view V_X and then we use the backward mapping technique. In this case, our approach is similar to the one stated in [49] and [17]. It has been demonstrated to be very fast and stable.

Note that, the X-interpolation can independently be performed row-per-row, facilitating a parallel implementation.

Y-extrapolation

From (23) we know that the Y-extrapolation is a forward mapping. The holes appearing in this process can be filled in by the linear interpolation along the y direction. The visibility problem can implicitly be solved, by either processing from top to bottom (if $y_{cD} < 0$), or from bottom to top (if $y_{cD} > 0$), or copy directly V_X into V_Y (if $y_{cD} = 0$). So the view is independently processed column-per-column. A parallel realization can also be promised.

Normally, since the image data are stored row per row, the Y-extrapolation may cause a large memory jump. Therefore several memory manipulation schemes (e.g., transposing followed by processing along x and transposing afterwards) have been investigated.

5.3. Integration with the whole system

All considerations discussed above have considerably speeded up the view reconstruction part. Further, if we consider the whole VIRTUE system (see Figure 3), another advantage is that we can combine components from the view reconstruction with other processing in the VIRTUE processing chain.

5.3.1 Integrate rectification with distortion correction

As we indicated in Section 3.1.2, if we adopt the "imaging distortion," for each pixel $\mathbf{p}$ in the distortion-free view, we can find a corresponding pixel $\hat{\mathbf{p}}$ in the distorted view by

$$\hat{\mathbf{p}} = f_{\triangle}(\mathbf{p}), \tag{44}$$

where $f_{\triangle}$ is the distortion function [38].

Then the intensity of $\hat{\mathbf{p}}$ in the distorted view can be assigned to the pixel $\mathbf{p}$ in the distortion-free view.

From Section 5.1, we know that, for each pixel $\mathbf{p}_{recti}$ in the rectified view, we may find a corresponding pixel $\mathbf{p}$ in the original view by

$$\mathbf{p} = f_{recti}(\mathbf{p}_{recti}), \tag{45}$$

where f_{recti} is the rectification function as in (32).

Combining the above two equations, we get

$$\hat{\mathbf{p}} = f_{\triangle\&\,recti}(\mathbf{p}_{recti}), \tag{46}$$

where $f_{\triangle\&\,recti} = f_{\triangle}(f_{recti})$.

Then the intensity of $\hat{\mathbf{p}}$ in the distorted view can be assigned to the pixel $\mathbf{p}_{recti}$ in the rectified view.

As the stereo setup is fixed for one conferencing session, the lookup table for $f_{\triangle\&\,recti}$ can be constructed once beforehand.

It is easy to see that the combined operation $f_{\triangle\&\,recti}$ has the same on-line calculation load (intensity assignment) as the distortion correction process $f_{\triangle}$.

5.3.2 Combining Z-zooming, de-rectification, and the compositor

By combining (38) with (33), we get

$$\tilde{\mathbf{p}}^Y = \frac{\lambda^D}{\lambda^Z}\mathbf{Z}_Z\mathbf{T}_D^{-1}\tilde{\mathbf{p}}^D. \tag{47}$$

In the compositor, to composite the uniform manmade environment with the telepresences, another transformation $\tilde{\mathbf{T}}_{\text{compo}}$ is applied on $\tilde{\mathbf{p}}^D$ to get the final displayed view pixel $\tilde{\mathbf{p}}_{\text{final}}$. As $\tilde{\mathbf{T}}_{\text{compo}}$ is invertible, we could have that

$$\tilde{\mathbf{p}}^D = \tilde{\mathbf{T}}_{\text{compo}}^{-1}\tilde{\mathbf{p}}_{\text{final}}. \tag{48}$$

Integrating the above two equations, we get

$$\tilde{\mathbf{p}}^{Y} = \left(\frac{\lambda^{D}}{\lambda^{Z}}\mathbf{Z}_{Z}\mathbf{T}_{D}^{-1}\tilde{\mathbf{T}}_{\text{compo}}^{-1}\right)\tilde{\mathbf{p}}_{\text{final}}. \quad (49)$$

Thus, except for an extra 3 by 3 matrix multiplication, the computation occurring in this equation is the same as that for the composition.

This means that if the view reconstruction is integrated into the whole VIRTUE processing chain, then only the X-interpolation and Y-extrapolation will take time, reducing the processing time further. The rectification step and the de-rectification (including Z-zooming) step have been combined into the other stages as shown above.

6. EXPERIMENTS AND COMPARISON

Based on the implementation from the "one optimal uniform scale factor" idea described in Section 5.2.1, we experimented our algorithm not only on the stereo sequences from the final VIRTUE configuration, but also on synthetic stereo views with ground-truth disparity maps. Moreover, to show the advantage of our proposal, we compared our algorithm with two other approaches (the trifocal tensor and the visual hull).

6.1. Experiments

To verify the quality (both geometry validity and visual quality for temporal and spatial continuity) and speed of our implementation, we performed experiments on the implementation of our algorithm using various stereo sequences coming from the VIRTUE setup (using cameras 1 (referred to as C_L) and 2 (referred to as C_R)) (see Figure 2).

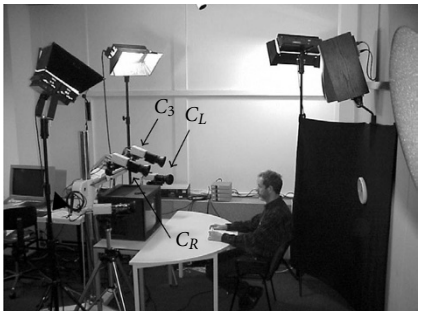
6.1.1 Geometry validity

To justify the geometry validity, a third camera C_3 (capturing real view V_3) was arbitrarily put between C_L and C_R. C_L, C_R, and C_3 are all calibrated at the same time (Figure 10).

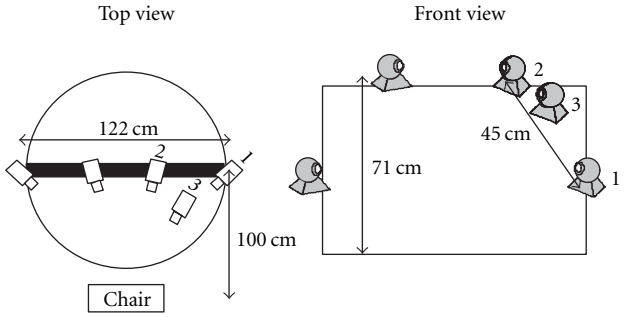
For this setup, we captured three sequences simultaneously (from C_L, C_R, and C_3, respectively) of a scene containing a talking person. After removing the background, correcting the distortion, and estimating the disparity, we choose $C_D = C_3$ in the view reconstruction process to reconstruct the novel view V_D from V_L and V_R. V_D and V_3 are compared with each other to check the geometry validity.

The three sequences have 30 frames each. In Figure 11, we show all steps of the view reconstruction for the last frame (note that all the example pictures shown in previous sections are also based on this frame). For better visibility, all images in this figure have been cut and only the interesting parts are kept. The rectification has been done together with the distortion correction. The Z-zooming has been combined with the de-rectification step. Further, using the area containing only the person as a mask, we calculate the difference image between V_3 and V_D together with the histogram of the difference in Figure 12.

Figure 14 shows another example. In this example, the two cameras in the stereo setup have been rotated a bit



(a) A recording process for the experiment.

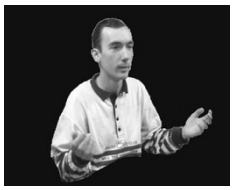


(b) The detailed configuration of the setup.

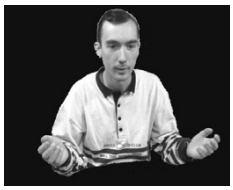
Figure 10: The real scenario of a data capturing process together with the detailed description of the setup configuration for experimenting the view reconstruction algorithm. The camera C_3 (3) is served as a virtual viewpoint to verify the final reconstruction result.

(Figure 13). With this setup, we may reduce the rectified image size to further save some processing time.

Further, to reduce the influence of the disparity map on the final results, we also synthesized several novel views from a pair of synthetic stereo views. The stereo pairs are made by ray tracing from a 3D scene that contains three spheres and a flat background. Four real images are mapped to the surfaces of the four objects, respectively. All pixels of the background have disparity value 0. This means that the background is at infinity. The left and right cameras are located at $[1\ \ 0\ \ 0]$ and $[-1\ \ 0\ \ 0]$, respectively. We employed the two ground-truth disparity maps to reconstruct novel views located at $[2\ \ 0\ \ 0]$, $[-2\ \ 0\ \ 0]$, $[2\ \ 2\ \ 0]$, and $[-2\ \ -2\ \ 0]$, respectively (Figure 15). From Figure 15, we can clearly get a feeling of viewpoint changing. Another pair of synthetic stereo views are produced in Figure 16 following the same steps described above. In contrast to the synthetic pair in Figure 15, the stereo data in Figure 16 contains large occlusion areas. The experiment on it reveals the inevitable problem of the current approach that some parts of the 3D scene may be visible in the desired novel view but they are not present in the stereo pair we have. Currently, we simply fill these areas by linear interpolation. Artifacts may appear because of this. More advanced texture analysis techniques [50] have to be employed.



(a) Original left view V_L.



(b) Original right view V_R.

(c) Rectified left view V_{rectiL}.

(d) Rectified right view V_{rectiR}.

(e) Left disparity map D_{LR}.

(f) Right disparity map D_{RL}.

(g) X-interpolated view V_X.

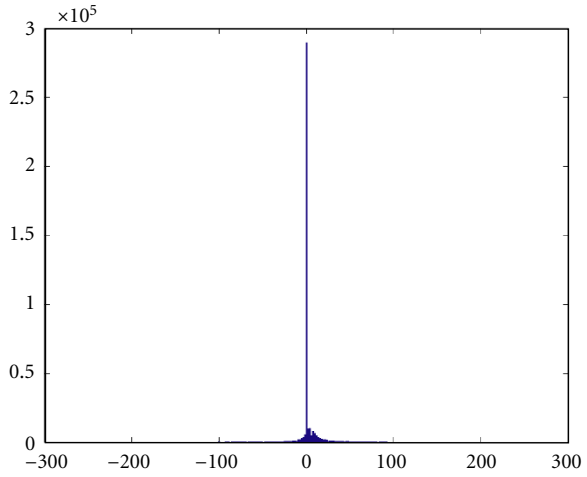
(h) Y-extrapolated view V_Y.

(i) Reconstructed view V_D.

(j) Real destination view V_3.

Figure 11: All intermediate and final results from our multi-step view reconstruction algorithm are listed above together with the disparity data. They are based on a pair of stereo frames ((a) and (b)) from the setup shown in Figure 10.

(a) Difference image.



(b) Histogram.

Figure 12: The difference image, between the view V_3 and V_D in Figure 11 considering only pixels within the area containing the talking person, together with its histogram are shown here to check the consistency of the reconstructed view with the real-perceived view at the novel viewpoint.

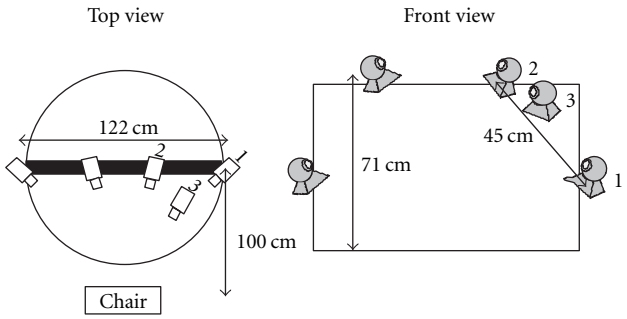
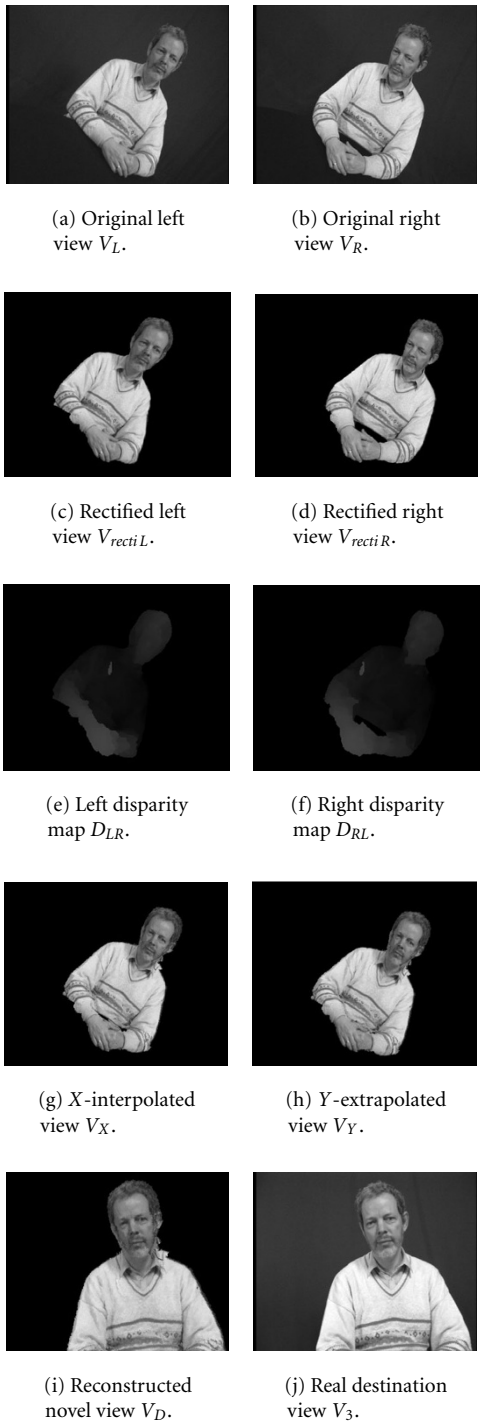


Figure 13: Setup with rotate cameras for the experiments in Figure 14. The advantage of this configuration is that the rectified stereo views may get smaller dimensions to ease the memory access load.

6.1.2 View adaptation quality

To test the visual quality of the view reconstruction, the 5th, 10th, 15th, 20th, 25th, and 30th frame of the stereo sequences acquired above are selected for experiment in two scenarios:



(a) Original left view V_L.

(b) Original right view V_R.

(c) Rectified left view V_{rectiL}.

(d) Rectified right view V_{rectiR}.

(e) Left disparity map D_{LR}.

(f) Right disparity map D_{RL}.

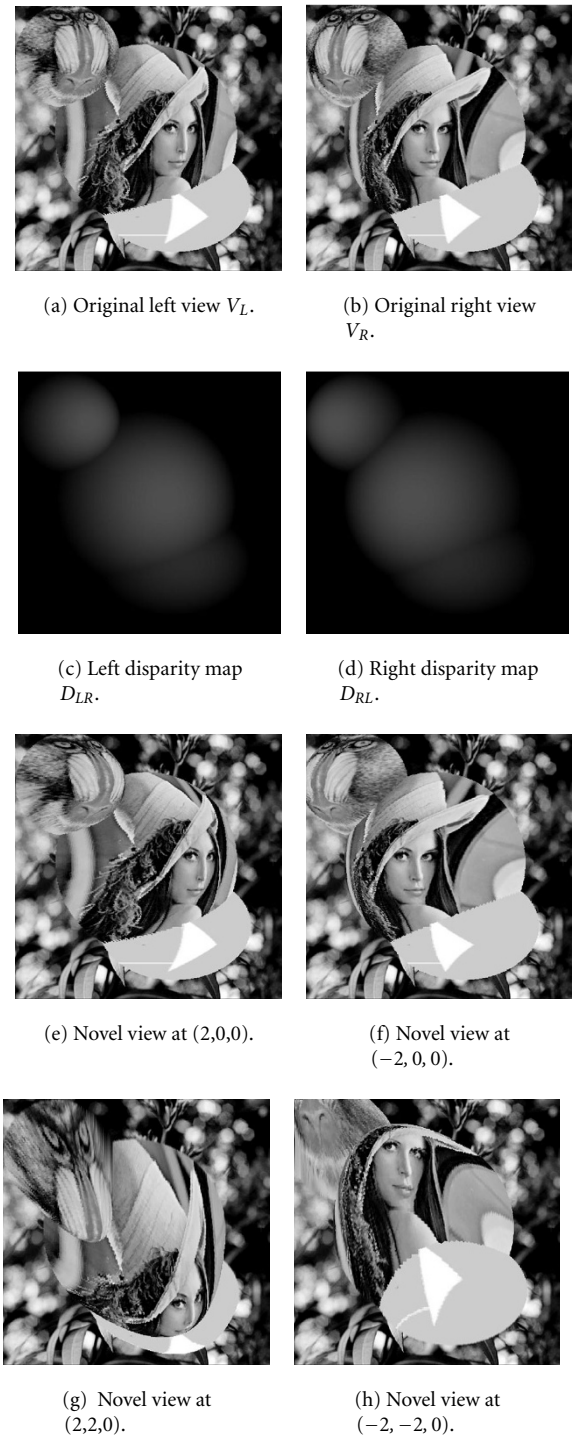
(g) X-interpolated view V_X.

(h) Y-extrapolated view V_Y.

(i) Reconstructed novel view V_D.

(j) Real destination view V_3.

FIGURE 14: All intermediate and final results from our multi-step view reconstruction algorithm are listed above together with the disparity data. They are based on a pair of stereo frames ((a) and (b)) from the setup shown in Figure 13 with a different talking person as the participant.



(a) Original left view V_L.

(b) Original right view V_R.

(c) Left disparity map D_{LR}.

(d) Right disparity map D_{RL}.

(e) Novel view at (2,0,0).

(f) Novel view at $(-2, 0, 0)$.

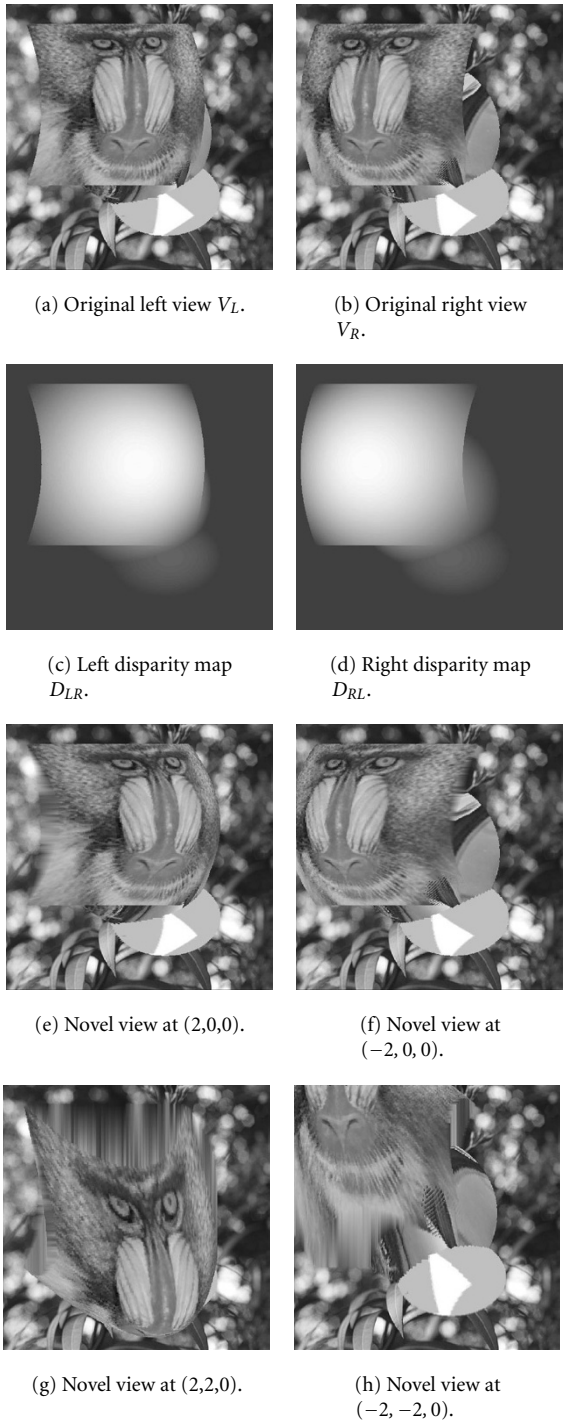
(g) Novel view at (2,2,0).

(h) Novel view at $(-2, -2, 0)$.

FIGURE 15: Novel views at difference poses reconstructed from the synthetic stereo pair (a) and (b) by our multi-step algorithm based on the ground-truth disparity maps (c) and (d). Clear feeling of viewpoint movement can be perceived.

(1) *Fixed viewpoint*. The same as above, let $C_D = C_3$ to fix the view point for all frames. This is intended to check the temporal continuity.

(2) *Dynamic viewpoint*. The view point C_D moves along a circle with the diameter coincident with the baseline **b** from $[-1.5 \;\; 0.1 \;\; 0.1]$ to $[1.5 \;\; 0.1 \;\; 0.1]$ in the WCS described in



(a) Original left view V_L.

(b) Original right view V_R.

(c) Left disparity map D_{LR}.

(d) Right disparity map D_{RL}.

(e) Novel view at (2,0,0).

(f) Novel view at $(-2, 0, 0)$.

(g) Novel view at (2,2,0).

(h) Novel view at $(-2, -2, 0)$.

Figure 16: Novel views at difference poses reconstructed from the synthetic stereo pair (a) and (b) by our multi-step algorithm based on the ground-truth disparity maps (c) and (d). It reveals the inevitable problem of the current approach that some parts of the 3D scene may be visible in the desired novel view but they are not present in the stereo pair we have.

Section 4.3. At the same time the view direction continuously changes for each different frame. This is used for checking the spatial continuity of the view point.

Table 1: The average time needed for the implementation based on the "one optimal uniform scale factor" idea on a Pentium III 550 MHz.

Function	Time	Frame size	Total time
X-interpolation	27 ms	1012×811	54 ms
Y-extrapolation	27 ms	YUV 4 : 2 : 2	
X-interpolation	10 ms	516×484	20 ms
Y-extrapolation	10 ms	YUV 4 : 2 : 2	

Table 2: The average time needed for the implementation based on the "two 1D transfer steps" idea on a Pentium III 550 MHz.

Function	Time	Frame size	Total time
X-transfer	45 ms	1012×811	90 ms
Y-transfer	45 ms	YUV 4 : 2 : 2	
X-transfer	12 ms	516×484	24 ms
Y-transfer	12 ms	YUV 4 : 2 : 2	

Results for these two tests are both shown in Figure 18.

6.1.3 Speed

To investigate the speed of the view reconstruction part, our algorithm has been evaluated on a Pentium III 550 MHz. The average execution time for the X-interpolation and Y-extrapolation steps is shown in Table 1. The average execution time for the X-transfer and Y-transfer steps is shown in Table 2.

In the final VIRTUE system the CCIR601 standard (720×576) YUV422 will be used. Since the processing time is roughly linear to the number of pixels to be processed, we estimated from Table 1 that for this format our implementation based on the "one optimal uniform scale factor" idea is feasible for real-time processing (25 fps). However, since other processing have to be done as well (e.g., disparity estimation, head tracking, network handling), based on the system analysis, we are exploring the parallel processing ability, which is inherent in the realization for the X-interpolation and the Y-extrapolation, on the final dedicated hardware Trimedia.

6.2. Comparison

Within the VIRTUE consortium, we also tried the trifocal tensor approach [51]. Using the same example data and disparity maps as in Figure 11, the reconstructed novel views both from our multi-step algorithm and from the trifocal tensor approach are shown in Figure 17. Except from the fact that the reconstructed view from our algorithm is a bit sharper, they are comparable with each other. However, under the same condition, on the Pentium III 550 MHz PC, the trifocal tensor implementation costs in average more than 140 ms for one frame. This is more than twice the average time shown in Table 1 that our algorithm costs. Further, since

FIGURE 17: The reconstructed novel view from our multi-step algorithm (left) compared with that from the trifocal tensor approach (right) using the same input data and system configuration as in Figure 11. Except that the reconstructed view from our algorithm is a bit sharper, they are comparable with each other.

the final dedicated hardware Trimedia is not good at floating point calculation, with Trimedia 133 (133 MHz), using the same data as that for the above PC evaluation, the trifocal tensor implementation costs more than 1000 ms for one frame. However, in our algorithm several small lookup tables can be built in advance. So in the multi-step view reconstruction process only integer addition and memory access are performed, reducing the total reconstruction time to 150 ms per frame. The parallel processing ability for the X-interpolation and Y-extrapolation is now explored on four Trimedias to achieve the real-time requirement.

It would be interesting to compare the visual results and time figure presented here to those reported in [29] and [30]. In [29], it is declared that their implementation is already real-time, but therein advanced hardware (four 600 MHz PCs and one dual 933 MHz Pentium III PC) are needed to guarantee the high speed, and the image dimension is much smaller than what we use. On the other hand, it is already found that the visual hull of an object does not match the object's exact geometry and in particular it cannot represent concave surface regions [32]. Because of this it may encounter difficulty for reconstructing the complex facial expressions and human gestures. Strong geometry distortions may be perceived from the novel view of a person with certain gesture as shown in [30].

7. CONCLUSIONS

The fixed-viewpoint experiments with both the original setup (Figures 11 and 18) and the rotated setup (Figure 14) show that the reconstructed novel views are comparable with the real views perceived at the virtual viewpoint. Our algorithm reproduces the viewpoint in good approximation, minimizing the visual projective distortion. The dynamic-viewpoint experiment shown in Figure 18 illustrates the continuity of the spatial change of the viewpoint, which is crucial for providing the motion parallax cue. The overall visual quality is good. However, there are still some artifacts along the object border and fingers. These are mainly caused by inaccuracies of the estimated disparity fields and of occlusion areas for which we filled in by linear interpolation (Figure 16, compared with Figure 15). The quality of the final displayed view can be further improved in the compositor but is not shown here. Since only still images can be shown in this paper, more extensive video results can be seen at http://www-ict.its.tudelft.nl/~bangjun/publications.html#ViewSynthesis.

The view reconstruction can be done in real time on a powerful Pentium processor or multiple dedicated hardware like Trimedias in favor of the inherent parallel processing ability.

Our contribution is two-fold (1) for the first time a theoretical analysis of the interpolation based view reconstruction approach is given and extended to the general case; (2) optimization and improvement of the quality of the proposed multi-step algorithm are considered in a practical application (in this case VIRTUE). The proposed algorithm together with its implementation can be integrated tightly with the whole VIRTUE processing chain to speed up the system. Further, since our algorithm only requires simple hardware such as Trimedia for real-time processing, the computation power of, for example, the main processor can be used for other tasks.

For our algorithm proposed in this paper, both the "multiple scale factors" extension and the "two 1D transfer steps" scheme would be interesting topics for the future VIRTUE++ system. Further, currently our algorithm processes video sequences frame per frame independently. It would be interesting if we could use the temporal information. In this way, only updated content would be reconstructed and thus most computations could be omitted.

APPENDICES

A. WORKING PRINCIPAL OF VIRTUE

VIRTUE project aims at constructing a virtual cooperation environment in which a three-party teleconference system becomes reality. In VIRTUE, you are at a meeting table with people spaced around in front of you. You are able to communicate with them effectively as if they are sitting next to you in the same room, in fact you are led to believe they are present in the same room, but they are actually located at several remote locations.

To achieve the above goal, seven key components should mainly be investigated and developed:

(1) high accurate camera calibration;
(2) dynamic eye tracking;
(3) realistic wide view synthesis for dynamic scenes;
(4) virtual and real scene fusion;
(5) tele- audio/video transmission;
(6) real-time processing platform;
(7) human factors investigation.

Supported by those techniques, the final VIRTUE system will feature:

- semi-immersive display with life-size head and torso images;
- camera views for multiple participants;
- integrated visual environment for multiple participants;
- compression and multiplex layer.

(1a) Left view of frame 5.

(1b) Right view of frame 5.

(1c) Fixed view of frame 5.

(1d) Dynamic view of frame 5.

(2a) Left view of frame 10.

(2b) Right view of frame 10.

(2c) Fixed view of frame 10.

(2d) Dynamic view of frame 10.

(3a) Left view of frame 15.

(3b) Right view of frame 15.

(3c) Fixed view of frame 15.

(3d) Dynamic view of frame 15.

(4a) Left view of frame 20.

(4b) Right view of frame 20.

(4c) Fixed view of frame 20.

(4d) Dynamic view of frame 20.

(5a) Left view of frame 25.

(5b) Right view of frame 25.

(5c) Fixed view of frame 25.

(5d) Dynamic view of frame 25.

(6a) Left view of frame 30.

(6b) Right view of frame 30.

(6c) Fixed view of frame 30.

(6d) Dynamic view of frame 30.

Figure 18: The results of our multi-step algorithm obtained from selected frames (5th, 10th, 15th, 20th, 25th, and 30th) of a pair of stereo sequences (columns 1 and 2) containing 30 frames with viewpoint fixed (column 3) or changed from frame to frame (column 4).

To fulfill the semi-immersive requirement, an integrated environment should be built for all participating conferees. For this purpose, techniques from computer graphics [2] can be employed to construct a uniform virtual environment. Then this virtual background can be combined with the "tele-presences" of the remote conferees to produce at each site a local conference atmosphere for the local participant.

The "tele-presences" of the conferees should be perceived differently for each participating conferee, in correspondence with his (her) viewpoint. This means a motion parallax cue need to be provided for all the participants [52]. To provide the motion parallax cue, an adaptive viewpoint vision system is developed. This system enables all conferees to experience a "look-around" feeling, and at the same time provide realistic eye to eye contact.

Specifically, in VIRTUE we have four cameras at each site to support a 3-party teleconference. In the virtual meeting space (Figure 2), if, for instance, at the local site we have to reconstruct a novel view of conferee C, we reconstruct it by only using the broadcasted stereo views coming from cameras 1 and 2 at remote site 2. The images from these two cameras are most similar to what conferee A sees of conferee C. In the same way conferee B is reconstructed at the local site from cameras 3 and 4. This approach has three advantages: (1) ease the bandwidth requirements; (2) facilitate the stereo correspondence estimation (smaller baseline); (3) support straightforward extension to 4-party or even more-party teleconference session.

B. FORWARD MAPPING VERSUS BACKWARD MAPPING

Assume that, for a parallel setup, the stereo images can be described by two functions $I_L(x, y)$ and $I_R(x, y)$, respectively. The objective is to construct a virtual middle view $I_M(x, y)$ situated just at the middle of the stereo pair.

Forward Mapping

Assumption

We have the disparity map $d_L(x, y)$ based on the left image to the right image

Interpolating Middle View Procedure

For every pixel in the left image

$I_M(x - d_L(x, y)/2, y)$

$= (I_L(x, y) + I_R(x - d_L(x, y), y))/2$

End For

description

Algorithm B.1

This process is called *Forward Mapping*, since it is going *from known to unknown* [12, 47]. During this process, the *visibility* problem due to over-mapping should be solved separately. Further, some post-processing step must be adopted to fill in the *holes* at which position no value was assigned. An alternative is *Backward Mapping—from unknown to search in known* [47].

Backward Mapping

Assumption

We have the disparity map $d_M(x, y)$ based on middle image to the right image.

Interpolating Middle View Procedure

For every pixel in the middle image

$I_M(x, y)$

$= (I_L(x + d_M(x, y), y) + I_R(x - d_M(x, y), y))/2$

End For

description

Algorithm B.2

This backward mapping process is very straightforward. And, unless subpixels are taken into account, no extra computation is needed. It has been widely used in image transformations such as rotation, warping, and distortion correction. However, many authors have postulated that it is very difficult to apply this technology to view reconstruction [12, 34] due to the non-invertibility of the geometric transformation equations.

ACKNOWLEDGMENTS

The authors are thankful to partners HHI, BT, Sony UK, and HWU in the VIRTUE consortium for valuable discussions and relevant contributions.

REFERENCES

[1] J. Lengyel, "The convergence of graphics and vision," *IEEE Computer*, vol. 31, no. 7, pp. 46–53, 1998.

[2] A. Watt, *3D Computer Graphics*, Addison-Wesley, New York, NY, USA, 3rd edition, 2000.

[3] Z. Zhang, "Image-based geometrically-correct photorealistic scene/object modeling (IBPhM): a review," in *Proc. 3rd Asian Conference on Computer Vision (ACCV '98)*, pp. 340–349, Hong Kong, January 1998.

[4] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric View*, MIT Press, Cambridge, Mass, USA, 2nd edition, 1993.

[5] S. F. El-Hakim, "Three-dimensional modeling of complex environments," in *Videometrics and Optical Methods for 3D Shape Measurement*, vol. 4309 of *SPIE Proceedings*, San Jose, Calif, USA, January 2001.

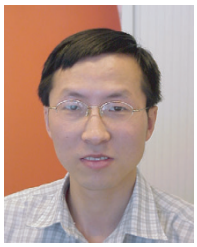
[6] Z. Zhang, "Image-based modeling of objects and human faces," in *Videometrics and Optical Methods for 3D Shape Measurement*, vol. 4309 of *SPIE Proceedings*, San Jose, Calif, USA, January 2001.

[7] H.-Y. Shum and S. B. Kang, "A review of image-based rendering techniques," in *IEEE/SPIE Visual Communications and Image Processing (VCIP 2000)*, pp. 2–13, Perth, Australia, June 2000.

[8] P. E. Debevec, "Pursuing reality with image-based modeling, rendering, and lighting," in *Keynote Paper for the 2nd Workshop on 3D Structure from Multiple Images of Large-Scale Environments and Applications to Virtual and Augmented Reality (SMILE2)*, Dublin, Ireland, June 2000.

[9] VIRTUE, "European IST project IST-1999-10044," 2000–2003, http://193.113.58.109/index.html.
[10] B. J. Lei and E. A. Hendriks, "Middle view stereo representation—an efficient architecture for teleconference with handling occlusions," in *Proc. IEEE International Conference on Image Processing*, November–December 2001.
[11] U. R. Dhond and J. K. Aggarwal, "Structure from stereo—a review," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1489–1510, 1989.
[12] D. Scharstein, *View Synthesis Using Stereo Vision*, vol. 1583 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, New York, NY, USA, 1999.
[13] L-Q. Xu, A. Loffler, P. J. Sheppard, and D. Machin, "True-view videoconferencing system through 3-D impression of telepresence," *BT Technical Journal*, vol. 17, no. 1, 1999.
[14] N. L. Chang and A. Zakhor, "Intermediate view reconstruction for three-dimensional scenes," in *Proc. International Conference on Digital Signal Processing*, vol. 2, pp. 636–641, Nicosia, Cyprus, July 1993.
[15] S. M. Seitz, *Image-based transformation of viewpoint and scene appearance*, Ph.D. thesis, University of Wisconsin-Madison, Computer Sciences Department, October 1997.
[16] H. Sawhney and S. Ayer, "Compact representations of videos through dominant and multiple motion estimation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 814–830, 1996.
[17] J. Shade, S. Gortler, L.-W. He, and R. Szeliski, "Layered depth images," in *SIGGRAPH '98*, pp. 231–242, Orlando, Fla, USA, July 1998.
[18] N. L. Chang and A. Zakhor, "A multivalued representation for view synthesis," in *Proc. IEEE International Conference on Image Processing*, Kobe, Japan, October 1999.
[19] C.-F. Chang, G. Bishop, and A. Lastra, "LDI tree: A hierarchical representation for image-based rendering," in *SIGGRAPH '99*, pp. 291–298, Los Angeles, Calif, USA, August 1999.
[20] S. Baker, R. Szeliski, and P. Anandan, "A layered approach to stereo reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 434–441, June 1998.
[21] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, M. Landy and J. A. Movshon, Eds., pp. 3–20, MIT Press, Cambridge, Mass, USA, 1991.
[22] K. Yamada, T. Ichikawa, T. Naemura, Aizawa K., and T. Saito, "High-quality stereo panorama generation using a three-camera system," in *SPIE Visual Communications and Image Processing (VCIP 2000)*, vol. 4067, pp. 419–428, Perth, Australia, June 2000.
[23] S. E. Chen, "Quicktime-VR—an image-based approach to virtual environment navigation," in *SIGGRAPH '95*, pp. 29–38, Los Angeles, Calif, USA, August 1995.
[24] M. Aggarwal and N. Ahuja, "On generating seamless mosaics with large depth of field," in *Proc. 15th International Conference on Pattern Recognition*, vol. 1, pp. 588–591, Barcelona, Spain, September 2000.
[25] S. Peleg, M. Ben-Ezra, and Y. Pritch, "OmniStereo: Panoramic stereo imaging," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 279–290, 2001.
[26] H. Youichi, K. I. Anjyo, and K. Arai, "Tour into the picture: Using a spidery mesh interface to make animation from a single image," in *Computer Graphics (SIGGRAPH '97 Proceedings)*, pp. 225–232, Los Angeles, Calif, USA, August 1997.
[27] D. Svedberg and S. Carlasson, "Calibration, pose and novel views from single images of constrained scenes," in *Proc. 11th Scandinavian Conference on Image Analysis*, pp. 111–118, Kangerlussuaq, Greenland, USA, June 1999.
[28] H. W. Kang, S. Y. Pyo, K. Anjyo, and S. Y. Shin, "Tour into the picture using a vanishing line and its extension to panoramic images," *Computer Graphics Forum*, vol. 20, no. 3, pp. 132–141, 2001.
[29] W. Matusik, C. Buehler, and L. McMillan, "Polyhedral visual hulls for real-time rendering," in *Proc. the 12th Eurographics Workshop on Rendering*, pp. 115–125, London, UK, June 2001.
[30] Y. Wexler and R. Chellappa, "View synthesis using convex and visual hulls," in *BMVC 2001*, Manchester, UK, 2001.
[31] R. Szeliski, "Rapid octree construction from image sequences," *CVGIP: Image Understanding*, vol. 58, no. 1, pp. 23–32, 1993.
[32] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," in *Proc. ACM SIGGRAPH '2000*, pp. 369–374, New Orleans, La, USA, July 2000.
[33] S. Laveau and O. D. Faugeras, "3-D scene representation as a collection of images," in *Proc. International Conference on Pattern Recognition*, vol. 1, pp. 689–691, Jerusalem, Israel, 1994.
[34] S. Avidan and A. Shashua, "Novel view synthesis by cascading trilinear tensors," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 4, pp. 293–306, October–December 1998.
[35] A. Shashua and L. Wolf, "On the structure and properties of the quadrifocal tensor," in *Proc. the European Conference on Computer Vision (ECCV)*, Dublin, Ireland, June 2000.
[36] S. E. Chen and L. Williams, "View interpolation for image synthesis," in *SIGGRAPH '93*, pp. 279–288, 1993.
[37] S. M. Seitz and C. R. Dyer, "View morphing," in *SIGGRAPH '96*, New Orleans, La, USA, August 1996.
[38] B. J. Lei and E. A. Hendriks, "Reviewing camera calibration techniques," Tech. Rep. ict-00-02, Information and Communication Theory Group, ITS, TUDelft, April 2000, http://www-ict.its.tudelft.nl/~bangjun/doc/Camera_Calibration_ps.zip.
[39] S. Rauthenberg, A. Graffunder, U. Kowalik, and P. Kauff, "Virtual shop and virtual meeting point—two prototype applications of interactive services using the new multimedia coding standard MPEG-4," in *Conference on Computer Communication*, Tokyo, Japan, September 1999.
[40] A. Redert, E. Hendriks, and J. Biemond, "Correspondence estimation in image pairs," *IEEE Signal Processing*, vol. 16, no. 3, pp. 29–46, 1999.
[41] P. Kauff, N. Brandenburg, M. Karl, and O. Schreer, "Fast hybrid block—and pixel—recursive disparity analysis for real-time applications in immersive tele-conference scenarios," in *Proc. WSCG 2001, 9th Int. Conference on Computer Graphics, Visualization and Computer Vision*, Plzen, Czech Republic, February 2001.
[42] O. Schreer, N. Brandenburg, S. Askar, and P. Kauff, "Hybrid recursive matching and segmentation-based postprocessing in real-time immersive video conferencing," in *Proc. VMV 2001, Vision, Modeling and Visualization 2001*, Stuttgart, Germany, November 2001.
[43] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.
[44] S. M. Seitz and C. R. Dyer, "Physically-valid view synthesis by image interpolation," in *Proc. Workshop on Representation of Visual Scenes*, pp. 18–25, Cambridge, Mass, USA, 1995.
[45] B. J. Lei and E. A. Hendriks, "View synthesis for VIRTUE," Tech. Rep. ict-00-03, Information and Communication Theory Group, TUDelft, August 2000, http://www-ict.its.tudelft.nl/~bangjun/publications.html#Technical.
[46] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *International Journal of Computer Vision*, vol. 33, no. 3, pp. 181–200, 1998.

[47] G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, Calif, USA, 1990.
[48] M. Leonard, "Computing visibility without depth," UNC Technical Report TR95-047, University of North Carolina, 1995.
[49] M. M. Oliveira and G. Bishop, "Relief textures," UNC Computer Science Technical Report TR99-015, University of North Carolina, March 1999.
[50] L.-Y. Wei and M. Levoy, "Texture synthesis over arbitrary manifold surfaces," in *SIGGRAPH '2001*, pp. 355–360, Los Angeles, Calif, USA, August 2001.
[51] F. Isgrò, E. Trucco, and L. Q. Xu, "Towards teleconferencing by view synthesis and large-baseline stereo," in *Proc. the IAPR/IEEE International Conference on Image Analysis and Processing*, Palermo, Italy, 2001.
[52] O. Schreer and P. Sheppard, "VIRTUE—the step towards immersive tele-presence in virtual video-conference systems," in *eWork and eBusiness 2000*, Madrid, Spain, October 2000.



B. J. Lei received his B.S. in computer software and M.S. in parallel network computing from Xi'an Jiaotong University, China in 1995 and 1998, respectively. He is now working towards a Ph.D. in the Information and Communication Theory Group at the Technical University of Delft, The Netherlands. His main research interests are in feature extraction, camera calibration, correspondence estimation, and novel view reconstruction. Currently he is involved in the EU framework V VIRTUE project.

E. A. Hendriks received his M.S. and Ph.D. degrees from the University of Utrecht in 1983 and 1987, respectively, both in physics. In 1987, he joined the Electrical Engineering Faculty of Delft University of Technology as an Assistant Professor. In 1994 he became a member of the Information and Communication Theory of this faculty and since 1997 he is heading the computer vision section of this group as an associate professor. His interest is in low-level image processing, image segmentation, stereoscopic and 3D imaging, motion and disparity estimation, and real time applications.