Point-Sampled 3D Video of Real-World Scenes ⋆

Michael Waschbüsch Stephan Würmlin Daniel Cotting
Markus Gross

Computer Graphics Laboratory, ETH Zurich, Switzerland

Abstract

This paper presents a point-sampled approach for capturing three-dimensional video footage and subsequent re-rendering of real-world scenes. The acquisition system is composed of multiple sparsely placed 3D video bricks. The bricks contain a low-cost projector, two gray-scale cameras and a high-resolution color camera. To improve on depth calculation we rely on structured light patterns. Texture images and pattern-augmented views of the scene are acquired simultaneously by time multiplexed projections of complementary patterns and synchronized camera exposures. High-resolution depth maps are extracted using depth-from-stereo algorithms performed on the acquired pattern images. The surface samples corresponding to the depth values are merged into a view-independent, point-based 3D data structure. This representation allows for efficient post-processing algorithms and leads to a high resulting rendering quality using enhanced probabilistic EWA volume splatting. In this paper, we focus on the 3D video acquisition system and necessary image and video processing techniques.

Key words: 3D video, free-viewpoint video, scene acquisition, point-based graphics

⋆ This paper is a revised and extended version of the manuscript 'M. Waschbüsch, S. Würmlin, D. Cotting, F. Sadlo, M. Gross; Scalable 3D Video of Dynamic Scenes' published in *The Visual Computer* 21(8–10):629–638, 2005, available at www.springerlink.com, DOI 10.1007/s00371-005-0346-7,

Email addresses: waschbuesch@inf.ethz.ch (Michael Waschbüsch), wuermlin@inf.ethz.ch (Stephan Würmlin), dcotting@inf.ethz.ch (Daniel Cotting), grossm@inf.ethz.ch (Markus Gross).

1 Introduction

3D video is a novel technology for capturing the dynamics and motion of a real-world scene during recording while providing the user with the possibility to change the viewpoint at will during playback. It is seen as one of the many promising emerging media technologies for next generation home entertainment and spatio-temporal visual effects. Free navigation regarding time and space in streams of visual media directly enhances the viewing experience, degree of immersion and interactivity. However, in most existing systems such virtual viewpoint effects have to be planned precisely and changes are no more feasible after the scene has been shot. As an example, freeze-and-rotate effects have been demonstrated in numerous feature films like *The Matrix*. It can only be realized by using a huge number of digital cameras which have to be placed very accurately. As another example, Digital Air's *Movia*® systems applies high-speed, HD cameras which can be controlled precisely such that no software view interpolation is needed for the desired effect. But as a consequence for both approaches, changes to the view trajectory are not possible during post-production.

Recently, several multi-view video systems have been presented which allow for realistic re-renderings of 3D video from arbitrary novel viewpoints. However, for producing high-quality imagery, the capturing systems are restricted to configurations where cameras are placed very close together. As an example, the system by Zitnick et al. [1] uses 8 cameras covering a horizontal field-of-view of 30°, where only linear arrangements are possible. To allow for more flexibility, i.e. configurations which cover an entire hemisphere with a small number of cameras, either model-based approaches need to be employed (e.g. Carranza et al. [2] with 8 cameras) or degradation in visual quality has to be accepted (e.g. Gross et al. [3] with 16 cameras). The latter two systems are also limited by the employed reconstruction algorithms to the capture of foreground objects or even pre-defined objects only.

In Waschbüsch et al. [4] we introduced a scalable framework for 3D video of dynamic scenes. Our work is motivated by the drawbacks of the aforementioned systems and by the vision of bringing 3D video to a new level where capturing, editing and subsequent high-quality re-rendering is cost-effective and convenient as with the 2D counterpart. For this purpose, special hardware solutions for real-time depth estimation, such as 3DV Systems' *ZCam*™ (http://www.3dvsystems.com), and Tyzx's *DeepSea* High-speed Stereo Vision System (http://www.tyzx.com) have recently become available. A central part of our research is the view-independent representation of the captured 3D geometry streams, with the goal to allow for similar authoring and editing techniques as carried out in common 3D content creation and modeling tools. Thereby, creation of spatio-temporal effects becomes straight-forward and one

has no longer to cope with the common limitations of image-based representations. In this paper we focus on the capturing and geometry processing part of our point-sampled 3D video framework.

2 Related work

This paper extends or integrates previous work in areas like point-based computer graphics, depth-from-stereo, and 3D video. For the sake of conciseness, we refer the reader to the ACM SIGGRAPH 2004 course on point-based computer graphics [5] and to relevant depth from stereo publications [6]. In the following, we will confine ourselves to related work in the area of 3D video.

In 3D video, multi-view video streams are used to re-render a time-varying scene from arbitrary viewpoints. There is a continuum of representations and algorithms suited for different acquisition setups and applications. Purely image-based representations [7] need many densely spaced cameras for applications like 3D TV [8]. Dynamic light field cameras [9,10] which have camera baselines of a couple of centimeters do not need any geometry at all. Camera configuration constraints can be relaxed by adding more and more geometry to image-based systems, as demonstrated by Lumigraphs [11]. Voxel-based representations [12] can easily integrate information from multiple cameras but are limited in resolution. Depth image-based representations [13,14] use depth maps which are computed predominantly by stereo algorithms [1]. Stereo systems still require reasonably small baselines and, hence, scalability and flexibility in terms of camera configurations is still not achieved. Redert et al. [15] use depth images acquired by Zcams [16] for 3D video broadcast applications. Appropriate representations for coding 3D audio/visual data are currently investigated by the MPEG-4 committee [17]. On the other end of the continuum, there are model-based representations which describe the objects or the scene by time-varying 3D geometry, possibly with additional video textures [2,18]. Almost arbitrary camera configurations become feasible, but most existing systems are restricted to foreground objects only.

Besides data representations, one has to distinguish between online and offline applications. Matusik et al. [19,20] focused on real-time applications, e.g. 3D video conferencing or instant 3D replays. However, they are restricted to capturing foreground objects only due to the nature of their silhouette-based depth reconstruction algorithms. Gross et al. [3] used a 3D video system based on a point sample representation [21] for their telecollaboration system *blue-c* and share the same limitation of only being able to reconstruct foreground objects. Mulligan et al. [22] also target telepresence. They compute geometric models with multi-camera stereo and transmit texture and depth over a network. Carranza et al. [2] presents an offline 3D video system which employs

an a-priori shape model which is adapted to the observed outline of a human. However, this system is only able to capture pre-defined shapes, i.e. humans. The 3D video recorder [23] handles point-sampled 3D video data captured by silhouette-based reconstruction algorithms and discusses data storage issues. No full scene acquisition is possible with the last two systems but almost arbitrary camera configurations are possible. Zitnick et al. [1] proposed a layered depth image representation for high-quality video view interpolation. Reconstruction errors at depth discontinuities are smoothed out by Bayesian matting. However, this approach again needs a quite dense camera setup to generate high-quality renderings in a limited viewing range. Scalability to larger setups is not addressed by the authors.

Cockshott et al. [24] also propose a 3D video studio based on modular acquisition units and pattern-assisted stereo. For concurrent texture acquisition, the patterns are projected using strobe lights requiring custom-built hardware. Only foreground objects are modeled using implicit surfaces.

3 Overview

Our 3D video acquisition system consists of several so-called 3D video bricks that are capturing high-quality depth maps from their respective viewpoints using calibrated pairs of stereo cameras (cf. figure 1). The matching algorithm used for depth extraction is assisted by projectors illuminating the scene with binary structured light patterns. Alternating projection of a pattern and its inverse allows for concurrent acquisition of the scene texture using appropriately synchronized color cameras.

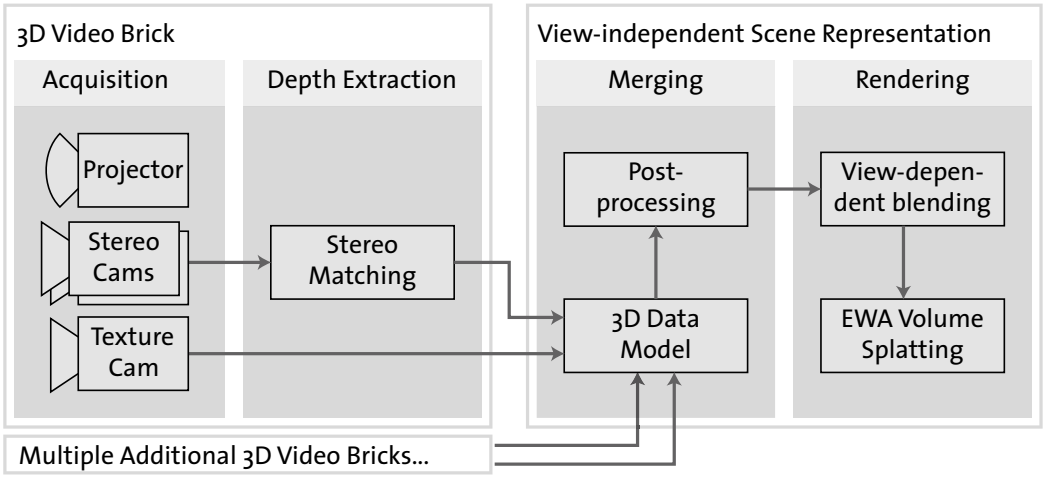

Fig. 1. Overview of the 3D video framework.

The results from different viewpoints are unified into a view-independent, point-based scene representation consisting of Gaussian ellipsoids. The data

model is post-processed to remove remaining outliers. Novel viewpoints of the dynamic scene are rendered using a probabilistic approach including view-dependent blending and EWA volume splatting.

4 Scalable acquisition system

In this section we present the concept of our low-cost z-cameras realized by 3D video bricks allowing simultaneous acquisition of textures and depth maps.

4.1 3D Video Bricks

The basic building blocks of the 3D video setup are movable bricks containing three cameras and a projector illuminating the scene with alternating patterns. Two grayscale cameras are responsible for depth extraction, while a color camera acquires the texture information of the scene. Figure 2 shows a single brick prototype with its components. In our current implementation, we operate with three bricks, each consisting of a standard PC with a genlock graphics board (NVIDIA Quadro FX3000G), a projector synchronizing to the input signal (NEC LT240K), and hardware-triggered cameras having XGA resolution and a capturing frame rate of 12 Hz (Point Grey Dragonfly). The components are mounted on a portable aluminum rig as shown in figure 2. The system is complemented by a synchronization microcontroller (MCU) connected to the cameras and the genlock-capable graphics boards.

Fig. 2. 3D video brick with cameras and projector (left), simultaneously acquiring textures (middle) and structured light patterns (right).

At a certain point in time, each brick can only capture depth information from a particular fixed position. In order to span a wider range of viewpoints and reduce occlusion effects, multiple movable bricks can be combined and individually oriented to cover the desired working space as illustrated in figure 3). Scalability of multiple bricks is guaranteed, because overlapping projections

are explicitly allowed by our depth reconstruction and because the computation load of each brick does not increase during real-time recording. Each brick performs the grabbing completely independently of the other bricks with the exception of the frames being timestamped consistently by using a common synchronization device.

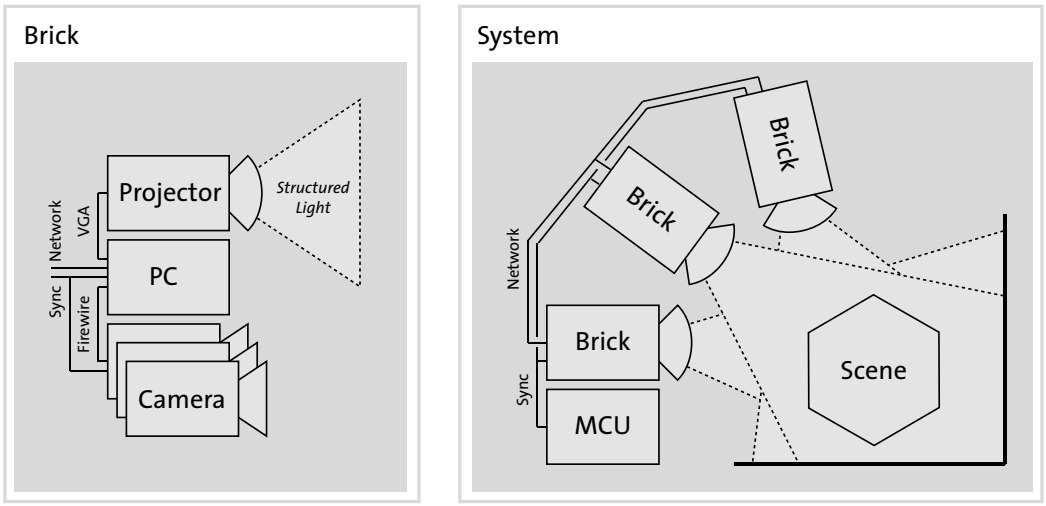

Fig. 3. Configuration of our 3D video prototype system.

In order to compute valid depth maps and merge the information gained from several bricks, all cameras in the 3D video system must be calibrated intrinsically and extrinsically. We determine imaging properties of all cameras using the MATLAB camera calibration toolbox [25]. Using a large $85 \times 120\text{cm}^2$ checkerboard calibration target at the center of the scene for extrinsic calibration, we are able to achieve a re-projection error of less than a quarter of a pixel. The projectors do not need to be calibrated.

4.2 Simultaneous texture and structured light acquisition

Each brick concurrently acquires texture information with the color camera and depth information using the stereo pair of grayscale cameras. Because texture and stereo cameras do not share exactly the same view, the depth images are warped into the view of the color camera.

Stereo vision (cf. section 5) generally requires a highly texturized scene to find good correlations between different views. It generally fails in reconstructing simple geometry of uniformly colored objects, e.g. white walls. Additionally, the textures should be non-periodic to guarantee unique matches. As a consequence, we add artificial textures to the scene by projecting structured light patterns, as originally proposed by Kang et al. [26]. We use a binary vertical stripe pattern with randomly varying stripe widths. It yields strong and unique correlations in the horizontal direction of the stereo baseline and is at the same

time insensitive to vertical deviations which may occur from inaccuracies in the camera calibration. To avoid untexturized shadows, the scene is illuminated by patterns from all bricks at the same time. Unlike pure structured light approaches or stereo matching between a single camera and a projector, our approach has the advantage of also working with multiple overlapping structured light projections, and it does not need a projector calibration.

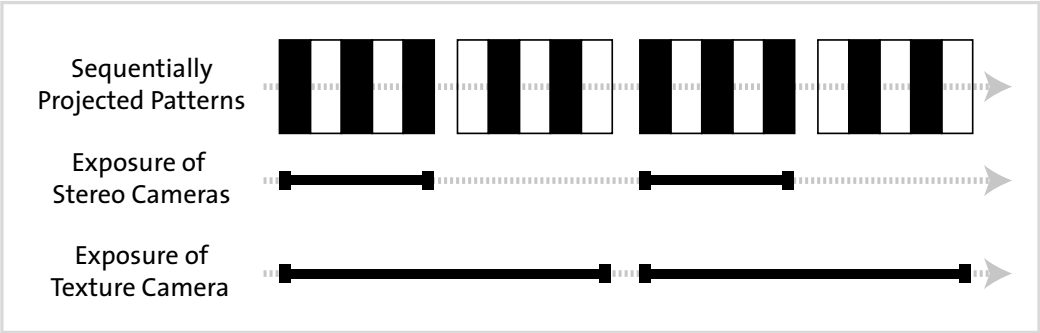

Fig. 4. Camera exposure with inverse pattern projection.

Alternating projections of structured light patterns and the corresponding inverses allow for simultaneous acquisition of the scene textures using an appropriately synchronized texture camera as illustrated in figure 4. Note that this camera does not see the patterns emanating from the projector, but only a constant white light, which preserves the original scene texture (cf. figure 2).

Since the patterns are changing at a limited rate of 60 Hz (projector input frequency), flickering is slightly visible to the human eye. Alternative solutions using imperceptible structured light [27] do not show any flickering, but require faster, more sensitive and therefore more expensive cameras for reliable stereo depth extraction.

5 Depth from stereo

Over the last decades, a large variety of stereo algorithms has been developed. A survey of different methods and their implementations can be found in Scharstein et al. [6]. Traditional algorithms are based on matching of small pixel neighborhoods within a small correlation window. However, due to their local view on the image, they have difficulties at ambiguities occurring in regions with uniform colors or repetitive textures. This can be easily solved by generating artificial textures using random structured light projections [26]. Moreover, such a high-frequency, high-contrast texture improves the precision of matching in the sub-pixel domain yielding very accurate depth values. A second issue are depth discontinuities which cannot be modeled if they occur inside the correlation window, leading to artifacts especially at silhouettes of

objects. Previously, we therefore used space-time stereo [28,29] which extends the correlation window into the time domain and thus allows to keep it small in image space. Because this assumes a high time coherence, it was only applied to static parts of the scene and thus there was still no solution for moving objects [4]. In the following sections we present a novel discontinuity-preserving stereo algorithm allowing for robust matching using large correlation windows while preserving depth discontinuities. Because it does not consider time, it can be applied uniformly on a whole video frame.

Global stereo matching algorithms [30,31] may be an alternative to our approach. By minimizing an energy composed of a correlation term and an edge-preserving smoothness term, they are able to handle both ambiguities and depth discontinuities. However, obtaining accurate sub-pixel estimates remains difficult. In texture-less regions, they cannot reach the precision of the structured-light approaches. In our experiments with structured light, the global methods often got confused by the many additional texture edges, yielding incorrect discontinuities in the computed depth image. Finding a globally correct tradeoff between data and smoothness energy appeared to be difficult.

5.1 Stereo matching

Stereo reconstruction is basically a search for corresponding pixels in two images. Those are finally used to compute the pixel depths by triangulation. The search can be restricted to the epipolar lines such that for two rectified images I_L and I_R one has to find for every pixel (u, v) its disparity d such that $I_L(u, v) = I_R(u + d, v)$. Following the notation of [28], this can be expressed as a minimization of an energy

$$E(d) = e(I_L(u, v), I_R(u + d, v)), \tag{1}$$

where e is an error function describing the difference between two pixels, for example the squared difference metric $e(a, b) = (a-b)^2$. In our implementation we use the error metric of Birchfield and Tomasi [32].

Because all real images are augmented by acquisition noise, the colors of two corresponding pixels will never be the same and the above pixel-wise minimization will not succeed. This problem is solved by minimizing over a small rectangular window $W(u_0, v_0)$ around one pixel of interest (u_0, v_0):

$$E(d) = \sum_{(u,v)\in W(u_0,v_0)} e(I_L(u, v), I_R(u + d, v)). \tag{2}$$

Our minimization procedure for equation (2) is divided into five steps:

S1. Matching cost computation: Construction of a disparity space image (DSI) by evaluating $e(I_L(u,v), I_R(u+d,v))$ for every pixel (u,v) and a set of discrete disparities d, i.e. all integer disparities within a specific range.
S2. Cost aggregation: Evaluation of the sum in equation (2) by applying a box filter to every DSI layer of constant disparity d.
S3. Discontinuity preservation: Application of a minimum filter to every DSI layer of constant disparity d.
S4. Minimization: Selection of the disparity with minimal cost for every pixel (u,v).
S5. Disparity refinement: Non-linear optimization of the disparities with sub-pixel accuracy.

S1, S2, S4 and S5 are common for many stereo matching algorithms, according to the taxonomy by Scharstein et al. [6]. We introduce an additional step S3 to improve the accuracy at depth discontinuities, which is explained in section 5.2. Furthermore, we describe our approach for disparity refinement in section 5.3.

5.2 Discontinuity preservation

Local stereo matching algorithms generally have difficulties in properly handling depth discontinuities. Due to occlusions, pixels in the first image may not have corresponding partners in the second image. Symmetric algorithms [33,34] are able to correctly detect those cases and mask out all affected pixels. One of the simplest symmetric approaches is cross checking. The stereo matching is performed twice: search in the right image for pixels corresponding to left image, and vice versa. Corresponding pixels in both disparity maps should have the same values, otherwise they belong to occlusions.

Due to the spatial extend of the correlation window, a second issue arises at discontinuities which is illustrated in figure 5. Because depth discontinuities generally come along with sharp color boundaries they cause a strong correlation. This yields a low matching cost as soon as the pixel window overlaps the color edge, no matter if the center pixel itself is part of that edge. Thus, the matching algorithm tends to find similar disparities in neighborhoods on both sides of color boundaries, which cannot be correct at depth discontinuities.

To solve that, we extend our matching algorithm to a multi-window approach. At each pixel we consider all matching windows of equal size that still contain the pixel of interest and choose the one with the best correlation. The chosen window usually has minimal overlaps with possible depth discontinuities, yielding a more reliable disparity value. A similar algorithm has already been proposed by [35] who considered 5 different dilated windows around each pixel

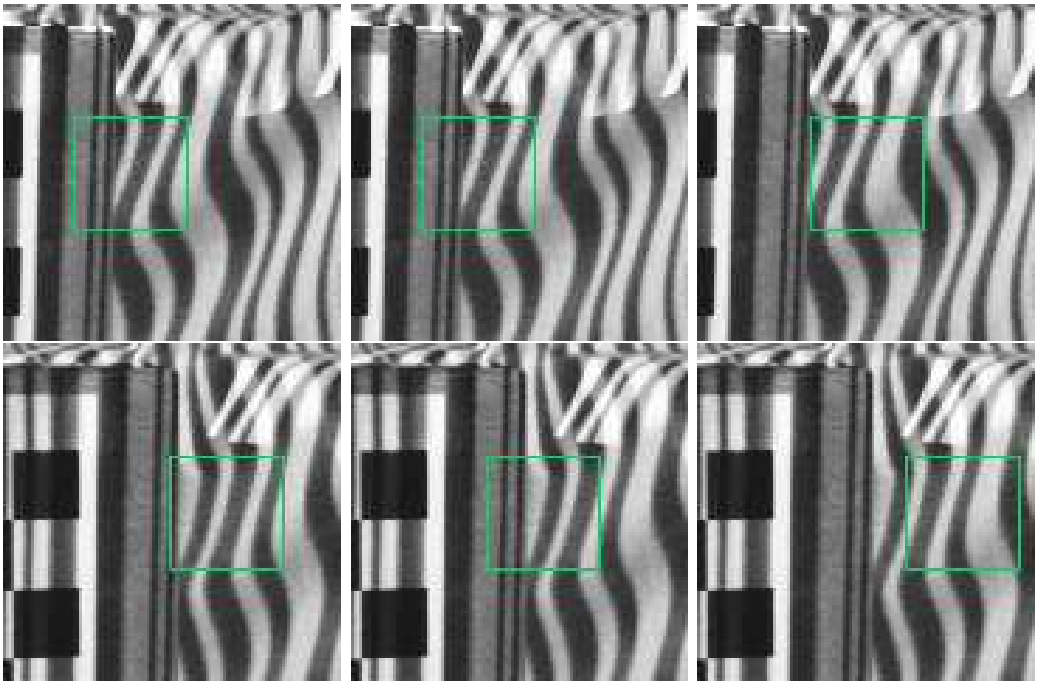

Fig. 5. Corresponding left (top row) and right (bottom row) pixel windows at depth discontinuities: desired correct correspondence (left), wrong correspondence detected by single-window matching (middle), correct correspondence detected by multi-window matching.

at the cost of a 5 times higher computation time. By sharing computations among neighboring pixels, our algorithm is able to consider all possible dilated windows containing the pixel of interest with low computational costs (see table 1). It is implemented as a minimum filter on the DSI layers of equal disparity, using the shape of the matching window as structuring element. The filter is applied after the cost aggregation in step S3 of the stereo pipeline:

As can be seen in figure 6, our extension produces less artifacts at discontinuities. It allows us to consider quite large correlation windows, e.g. 25×25 pixels, increasing the overall robustness.

5.3 Disparity refinement

The disparities computed so far are only approximations on a coarse level. Besides from being integer valued they may deviate from the ground truth by a few pixels because equation (2) assumes a constant disparity over the whole correlation window. This is, however, only correct for planar surfaces parallel to the image plane. To obtain accurate results at subpixel level, we extend the minimization by also searching for the gradient (d_u, d_v) of the disparity,

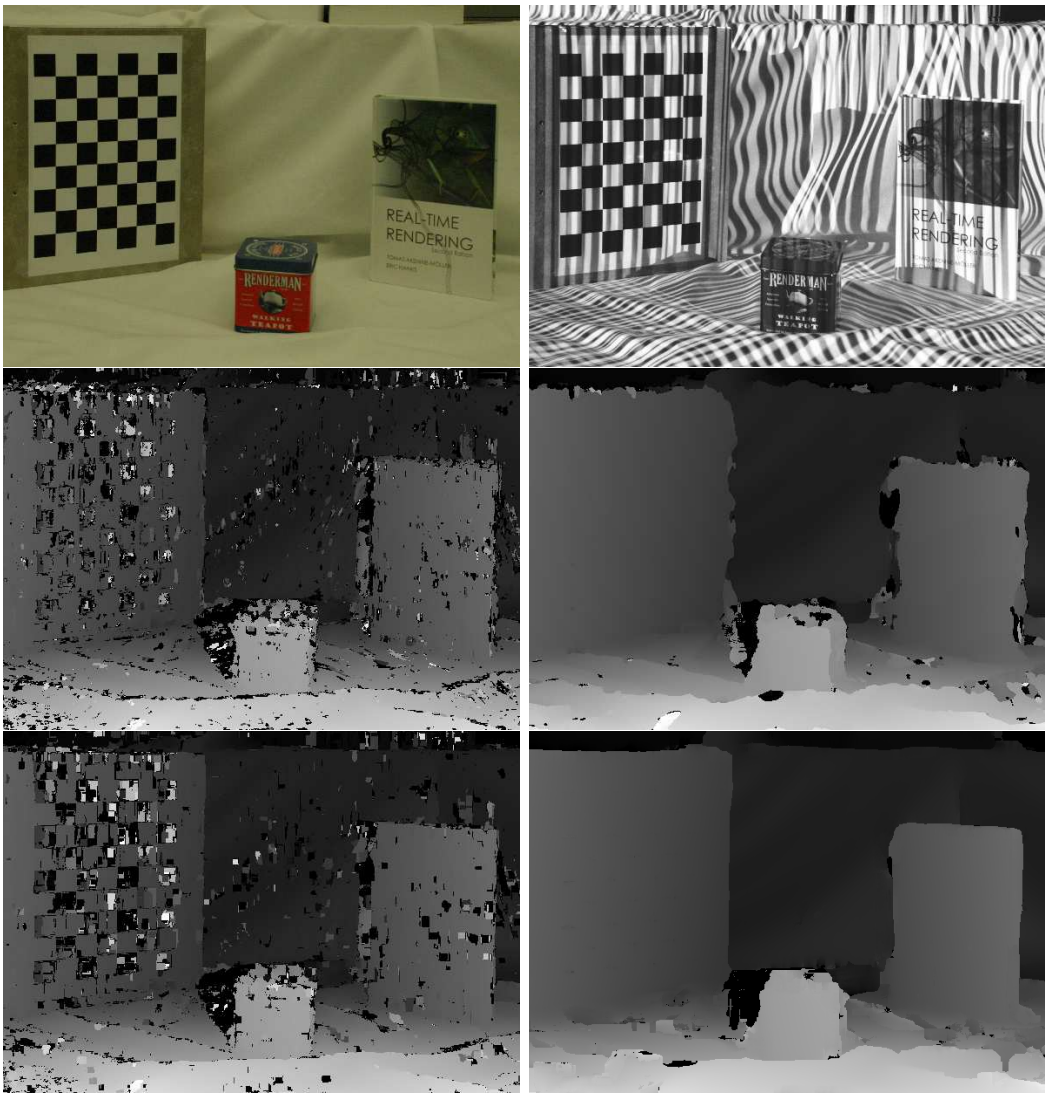

Fig. 6. Discontinuity preservation. Upper row: scene without and with structured light illumination. Disparity maps using a 7×7 (left) and 25×25 (right) pixel matching window, without (middle) and with (lower row) discontinuity preservation.

similar to [28]:

$$E(d, d_u, d_v) = \sum_{(u,v) \in W(u_0, v_0)} e(I_L(u, v), I_R(u + \tilde{d}, v)), \tag{3}$$

where

$$\tilde{d} = d + d_u(u - u_0) + d_v(v - v_0). \tag{4}$$

This corresponds to linearly changing disparities in the matching window which is exact for all planar surfaces and a good local approximation of the scene geometry.

By using the previously computed coarse disparities as starting values, equation (3) can be minimized using the Levenberg-Marquardt algorithm or sim-

ilar non-linear optimization methods. We handle discontinuities by masking those pixels in the correlation window whose coarse disparities differ too much from the one of the center pixel, yielding a non-rectangular window at depth boundaries. The additionally computed disparity gradients are also used later to compute the 3D scene representation (section 6.1).

The computational effort of our C implementation of the whole stereo matching pipeline is summarized in table 1. Most of the time is used by the disparity refinement because we are still using a non-optimized standard implementation for that step.

Table 1
Stereo matching performance for an image pair of 791×524 pixels, using a 25×25 pixels correlation window. Timings were measured on a 3GHz Pentium-4 PC. Note that steps S1 to S4 denote the overall time for two matching processes which are necessary to perform cross checking.

S1 matching cost computation	9.0s
S2 cost aggregation	4.7s
S3 discontinuity preservation	9.3s
S4 minimization	1.2s
S5 disparity refinement	348s

6 View-independent scene representation

To model the resulting three-dimensional scene, we propose a view-independent, point-based data representation. The distinct reconstructed views are merged by back-projecting their image pixels to a common three-dimensional world reference frame. With this approach, we achieve a convenient, scalable representation which allows for easy adding of additional views. Our model is in principle capable of providing a full 360° view if the scene has been acquired from enough viewpoints. Unlike image-based structures it is possible to keep the amount of data low by removing redundant points from the geometry [36]. Compared to mesh-based methods, points provide advantages in terms of scene complexity because they reduce the representation to the absolutely necessary data and do not carry any topological information, which is often difficult to acquire and maintain. As each point in our model has its own assigned color, we also do not have to deal with texturing issues. Moreover, our view-independent representation is very suitable for 3D video editing applications since tasks like object selection or re-lighting can be achieved easily with standard point processing methods [5].

Our point-based model consists of an irregular set of samples, where each sample corresponds to a point on a surface and describes its properties such as location and color. The samples can be considered as a generalization of conventional 2D image pixels towards 3D video. If required, the samples can be easily extended with additional attributes like surface normals for re-lighting.

To avoid artifacts in re-rendering, we have to ensure full surface coverage of the samples. Thus, our samples cannot be represented by infinitesimal points, but need to be considered as small surface or volume elements. One obvious representation are surfels [37], which are small elliptical disks aligned tangentially to the surface. However, surfels do not handle noise due to inaccurate 3D reconstruction or camera calibration very well, and require accurate geometries and therefore stable surface normals.

Therefore, we have chosen a different approach, similar to Hofsetz et al. [38]. Every point is modeled by a three-dimensional Gaussian ellipsoid spanned by the vectors $\mathbf{t}_1$, $\mathbf{t}_2$ and $\mathbf{t}_3$ around its center $\mathbf{p}$. This corresponds to a probabilistic model describing the positional uncertainty of each point by a trivariate normal distribution

$$\begin{aligned} p_X(\mathbf{x}) &= N(\mathbf{x}; \mathbf{p}, \mathbf{V}) \\ &= \frac{1}{\sqrt{(2\pi)^3 |\mathbf{V}|}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p})^{\mathrm{T}} \mathbf{V}^{-1} (\mathbf{x}-\mathbf{p})} \end{aligned} \tag{5}$$

with expectation value $\mathbf{p}$ and covariance matrix

$$\mathbf{V} = \Sigma^{\mathrm{T}} \cdot \Sigma = \begin{pmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{t}_3 \end{pmatrix}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{t}_3 \end{pmatrix} \tag{6}$$

composed of 3×1 column vectors $\mathbf{t}_i$.

To estimate $\mathbf{V}$, Hofsetz et al. [38] have chosen an approach based on the quality of the pixel correlation of the stereo matching. It turns out that these resulting heuristic uncertainties are quite large compared to the high-quality disparities we are able to obtain from our structured light assisted approach. Consequently, we propose a different approach which constrains the uncertainties to cover only small but well-defined acquisition errors. We assume that most disparities are correctly estimated up to small errors caused by deviations in the camera calibration and compute point sizes which just provide full surface coverage.

Assuming a Gaussian model for each image pixel uncertainty, we first compute the back-projection of the pixel into three-space which is a 2D Gaussian parallel to the image plane spanned by two vectors $\mathbf{t}_u$ and $\mathbf{t}_v$. Extrusion into the

third domain by adding a vector $\mathbf{t}_z$ guarantees a full surface coverage under all possible views. This is illustrated in figure 7.

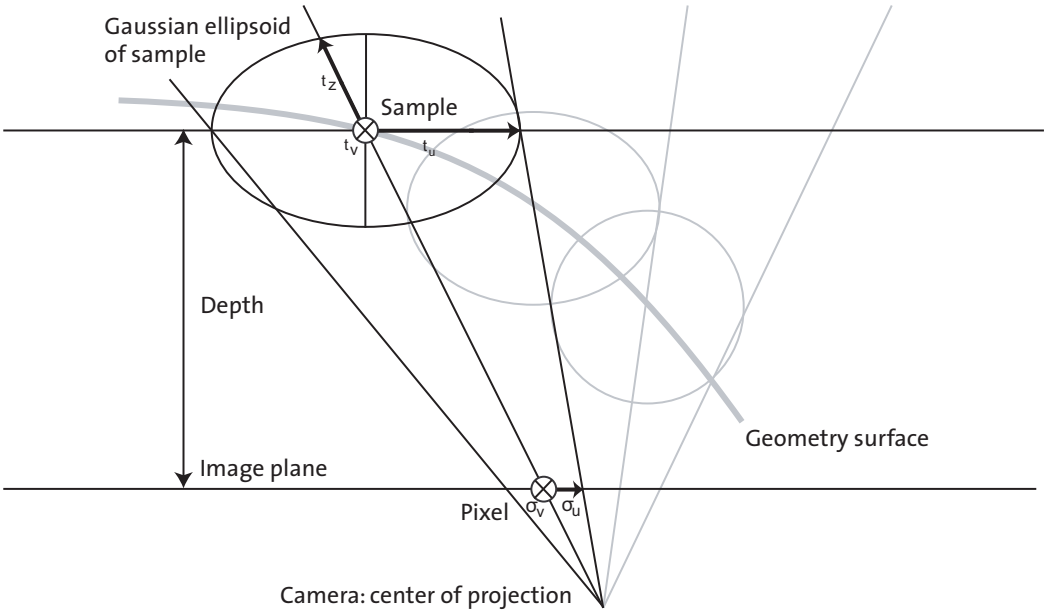

Fig. 7. Construction of a 3D Gaussian ellipsoid.

Each pixel (u, v) is spanned by orthogonal vectors $\sigma_u(1, 0)^{\mathrm{T}}$ and $\sigma_v(0, 1)^{\mathrm{T}}$ in the image plane. Assuming a positional deviation σ_c, the pixel width and height under uncertainty are $\sigma_u = \sigma_v = 1 + \sigma_c$. σ_c is estimated to be the average re-projection error of our calibration routine.

The depth z of each pixel is inversely proportional to its disparity d as defined by the equation

$$z = -\frac{f_L \cdot \|\mathbf{c}_L - \mathbf{c}_R\|}{d + p_L - p_R}, \tag{7}$$

where f_L is the focal length of the left rectified camera, $\mathbf{c}_L$ and $\mathbf{c}_R$ are the centers of projection, and p_L and p_R the u-coordinates of the principal points. The depth uncertainty σ_z is obtained by differentiating equation (7) and augmenting the gradient Δd of the disparity with its uncertainty σ_c:

$$\sigma_z = \frac{f_L \cdot \|\mathbf{c}_L - \mathbf{c}_R\|}{(d + p_L - p_R)^2} \cdot (\Delta d + \sigma_c)\,. \tag{8}$$

Now, we can construct for each pixel its Gaussian in ray space with

$$\Sigma_R = \begin{pmatrix} \sigma_u \cdot z & 0 & \sigma_z \cdot u \\ 0 & \sigma_v \cdot z & \sigma_z \cdot v \\ 0 & 0 & \sigma_z \end{pmatrix}. \tag{9}$$

It is transformed into the world coordinate system by

$$\Sigma = \mathbf{P}^{-1} \cdot \Sigma_R \tag{10}$$

using the camera projection matrix $\mathbf{P}$. The centers $\mathbf{p}$ of the ellipsoids are constructed by back-projection as

$$\mathbf{p} = \mathbf{P}^{-1} \cdot (u, v, 1)^{\mathrm{T}} \cdot z + \mathbf{c}, \tag{11}$$

where $\mathbf{c}$ is the center of projection of the camera.

Finally, the resulting point-cloud is post-processed to remove remaining artifacts. Apart from standard outlier removal techniques [39], we clean the data by ensuring photo-consistence with the input images [4]. Redundant points are removed using a clustering algorithm [36].

7 Rendering

We render novel viewpoints of the scene using the GPU and CPU cooperatively. Smooth images are generated using the uncertainties of the Gaussian ellipsoids. Our method combines the advantages of two probabilistic image generation approaches described in Broadhurst et al. [40]. Additionally we perform a view-dependent blending similar to Hofsetz et al. [38].

7.1 Probabilistic rendering

Broadhurst et al. [40] use probabilistic volume ray casting to generate smooth images. Each ray is intersected with the Gaussians of the scene model. At a specific intersection point $\mathbf{x}$ with the sample i, the evaluation $N(\mathbf{x}; \mathbf{p}_i; \mathbf{V}_i)$ of the Gaussian describes the probability that a ray hits the corresponding surface point. To compute the final pixel color, two different approaches are described. The maximum likelihood method associates a color with the ray using only the sample which has the most probable intersection. The second approach employs the Bayes rule: It integrates all colors along each ray weighted by the probabilities without considering occlusions. Thus, the color of a ray R is computed as

$$\mathbf{c}_R = \frac{\int_{\mathbf{x} \in R} \sum_i \mathbf{c}_i N(\mathbf{x}; \mathbf{p}_i, \mathbf{V}_i)}{\int_{\mathbf{x} \in R} \sum_i N(\mathbf{x}; \mathbf{p}_i, \mathbf{V}_i)}. \tag{12}$$

The maximum likelihood method generates crisp images, but it also sharply renders noise in the geometry. The Bayesian approach produces very smooth

images with fewer noise, but is incapable of handling occlusions and rendering solid surfaces in an opaque way.

Fig. 8. Comparison of maximum likelihood (left) and Bayesian rendering (center) with our approach (right).

We propose a rendering method which combines both approaches in order to benefit from their respective advantages. Our idea is to accumulate the colors along each ray like in the Bayesian setting, but to stop as soon as a maximum accumulated probability has been reached. Reasonably, a Gaussian sample should be completely opaque if the ray passes its center. The line integral through the center of a three-dimensional Gaussian has a value of $\frac{1}{2\pi}$ and for any ray R it holds that

$$\int_{\mathbf{x}\in R} N(\mathbf{x};\mathbf{p},\mathbf{V}) \leq \frac{1}{2\pi}. \tag{13}$$

Thus, we accumulate the solution of the integrals of equation (12) by traversing along the ray from the camera into the scene and stop as soon as the denominator of equation (12) reaches $\frac{1}{2\pi}$. Assuming that solid surfaces are densely sampled, the probabilities within the surface boundaries will be high enough so that the rays will stop within the front surface.

We compare the maximum likelihood and Bayesian rendering with our approach on noisy data in figure 8. Notice the large distortions in the maximum likelihood image that get smoothed out by the other two methods. However, the Bayesian renderer blends all the points including those from occluded surfaces, while our method renders opaque surfaces and maintains the blending. Thus, our renderer provides the advantages of both previous methods.

In our implementation, we replace the ray caster by a volume splatter [41] running on graphics hardware. After pre-sorting the Gaussians according to their depths by the CPU, the GPU splats them from front to back. The pixel colors are blended according to the Gaussian alpha masks until the accumulated alphas reach a level of saturation. This is directly supported by the OpenGL blending function GL_SRC_ALPHA_SATURATE.

Fig. 9. Rendering without (left) and with (right) view-dependent uncertainty blending (right).

One specific sample usually looks most accurate from the view it has been acquired from. As the angle between the acquisition and the virtual view becomes larger, the quality decreases depending on the depth uncertainty of the Gaussian. Projections of samples with high uncertainty become more and more stretched, introducing visible artifacts, while samples with low uncertainties look good from all views. We treat this issue by applying the view-dependent blending of Hofsetz et al. [38]. The authors compute an alpha value representing the maximum opacity of each Gaussian in its center using the view-dependent criteria of Buehler et al. [42], weighted by the individual depth uncertainty σ_z. This method largely improves the resulting image quality, as can be seen in figure 9.

8 Results and discussion

For the results presented in this section, we have recorded a dynamic scene with our setup consisting of three sparsely placed bricks covering an overall viewing angle of 70° horizontally and 30° vertically. The left side of figure 10 shows novel views of the acquired scene in figure 2, rendered from our reconstructed 3D model. Moreover, our point-based, view-independent data model conveniently provides possibilities 3D video editing and special effects (figure 10, right).

Our re-renderings have a decent look with a high-quality texture. Acquisition noise is smoothed out by our blending method. We are even able to reconstruct highly detailed geometry like the folds in the tablecloth shown in figure 11.

Fig. 10. Re-renderings of the 3D video from novel viewpoints (left) and actor cloning as a special effect (right).

Fig. 11. Geometric detail in the tablecloth. For illustration we recomputed smooth surface normals and rendered the scene with Phong lighting under two different illumination conditions.

However, there are still some artifacts at silhouettes which we would like to eliminate in the future. This is possible by using matting approaches as done by Zitnick et al. [1]. Some remaining outliers are also visible in the images. They could be reduced by enforcing time coherence in the whole reconstruction pipeline. Time coherence may further help in filling remaining holes caused by occlusions.

With our system we are able to acquire a large viewing range with a relatively low amount of cameras. To support increasingly large ranges, our system is

scalable up to full spherical views. To fully cover 360° in all dimensions about 8 to 10 3D video bricks are needed. Note that this is not constrained to convex views. Although overlaps in the geometry can help to improve the overall quality, they are not required as each brick reconstructs its own scene part independently.

The use of projectors still imposes some practical constraints because of the visible light spots and shadows that are created in the scene. We accept this limitation for the sake of a maximum 3D reconstruction quality. Using calibrated projectors it would be possible to compute the incident light at each surface point and compensate for the artifacts.

9 Conclusions and future work

We presented a system for recording and re-rendering 3D video of dynamic scenes. The brick concept combined with a point-based, view-independent data model allows for scalable capturing of a large viewing range with sparsely placed components. We are able to obtain high-quality depth maps using discontinuity-sensitive stereo on structured light while concurrently acquiring textures of the scene. Decent-quality images of novel views are achieved using Gaussian ellipsoid rendering with view-dependent blending methods. Our data representation can directly benefit from a large variety of available point processing algorithms, e.g. normal estimation for re-lighting effects.

In the future, we would like to improve the resulting image quality by exploiting inter-brick correlation and eliminating remaining artifacts at silhouettes using matting approaches. It would also be desirable to achieve a comparable rendering quality using passive reconstruction algorithms only, which would make our system suitable for large outdoor scenes. In addition, we would like to address compression of time-varying point-sampled 3D video streams and investigate novel possibilities of 3D video editing for special effect creation.

Acknowledgements

We would like to thank Filip Sadlo for helping with the camera calibration and Doo Young Kwon for acting in the 3D video. This work is carried out in the context of the blue-c-II project, funded by ETH grant No. 0-21020-04 as an internal poly-project.

References

[1] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, in: Proc. SIGGRAPH '04, 2004, pp. 600–608.

[2] J. Carranza, C. Theobalt, M. Magnor, H.-P. Seidel, Free-viewpoint video of human actors, in: Proc. SIGGRAPH '03, 2003, pp. 569–577.

[3] M. Gross, S. Würmlin, M. Näf, E. Lamboray, C. Spagno, A. Kunz, A. V. Moere, K. Strehlke, S. Lang, T. Svoboda, E. Koller-Meier, L. V. Gool, O. Staadt, blue-c: A spatially immersive display and 3D video portal for telepresence, in: Proc. SIGGRAPH '03, 2003, pp. 819–827.

[4] M. Waschbüsch, S. Würmlin, D. Cotting, F. Sadlo, M. Gross, Scalable 3D video of dynamic scenes, The Visual Computer (Proc. Pacific Graphics '05) 21 (8–10) (2005) 629–638.

[5] M. Alexa, M. Gross, M. Pauly, H. Pfister, M. Stamminger, M. Zwicker, Point-Based Computer Graphics, SIGGRAPH '04 Course Notes, 2004.

[6] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, International Journal of Computer Vision 47 (1-3) (2002) 7–42.

[7] M. Levoy, P. Hanrahan, Light field rendering, in: Proc. SIGGRAPH '96, 1996, pp. 31–42.

[8] W. Matusik, H. Pfister, 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes, in: Proc. SIGGRAPH '04, 2004, pp. 814–824.

[9] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, M. Levoy, High performance imaging using large camera arrays, in: Proc. SIGGRAPH '05, 2005, pp. 765–776.

[10] J. C. Yang, M. Everett, C. Buehler, L. McMillan, A real-time distributed light field camera, in: Proc. Eurographics Workshop on Rendering '02, 2002, pp. 77–86.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen, The lumigraph, in: Proc. SIGGRAPH '96, 1996, pp. 43–54.

[12] S. Vedula, S. Baker, T. Kanade, Spatio-temporal view interpolation, in: Proc. Eurographics Workshop on Rendering '02, 2002, pp. 65–76.

[13] Y. Bayakovski, L. Levkovich-Maslyuk, A. Ignatenko, A. Konushin, D. Timasov, A. Zhirkov, M. Han, I. K. Park, Depth image-based representations for static and animated 3d objects, in: Proc. ICIP '02, Vol. 3, 2002, pp. 25–28.

[14] J. Shade, S. Gortler, L.-W. He, R. Szeliski, Layered depth images, in: Proc. SIGGRAPH '98, 1998, pp. 231–242.

[15] A. Redert, M. O. de Beeck, C. Fehn, W. Ijsselsteijn, M. Pollefeys, L. V. Gool, E. Ofek, I. Sexton, P. Surman, ATTEST: Advanced three-dimensional television system technologies, in: Proc. 3DPVT '02, 2002, pp. 313–319.

[16] G. J. Iddan, G. Yahav, 3D imaging in the studio (and elsewhere...), in: Proc. SPIE '01, Vol. 4298, 2001, pp. 48–55.

[17] A. Smolic, H. Kimata, Description of exploration experiments in 3DAV, in: JTC1/SC29/WG11 N6194, ISO/IEC, 2003.

[18] T. Kanade, P. Rander, P. J. Narayanan, Virtualized reality: Construction of virtual worlds from real scenes, Proc. IEEE Multimedia 4 (1) (1997) 34–47.

[19] W. Matusik, C. Buehler, L. McMillan, Polyhedral visual hulls for real-time rendering, in: Proc. Eurographics Workshop on Rendering '01, 2001, pp. 115–125.

[20] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, L. McMillan, Image-based visual hulls, in: Proc. SIGGRAPH '00, 2000, pp. 369–374.

[21] S. Würmlin, E. Lamboray, M. Gross, 3D video fragments: Dynamic point samples for real-time free-viewpoint video, Computers & Graphics '04 28 (1) (2004) 3–14.

[22] J. Mulligan, K. Daniilidis, View-independent scene acquisition for tele-presence, in: Proc. International Symposium on Augmented Reality '00, 2000, pp. 105–110.

[23] S. Würmlin, E. Lamboray, O. G. Staadt, M. H. Gross, 3D video recorder, in: Proc. Pacific Graphics 02, 2002, pp. 325–334.

[24] W. P. Cockshott, S. Hoff, J.-C. Nebel, An experimental 3D digital TV studio, in: Proc. Vision, Image & Signal Processing 03, 2003, pp. 28–33.

[25] J.-Y. Bouguet, Camera calibration toolbox for matlab, http://www.vision.caltech.edu/bouguetj/calib_doc.

[26] S. Kang, J. Webb, L. Zitnick, T. Kanade, A multi-baseline stereo system with active illumination and real-time image acquisition, in: Proc. ICCV '95, 1995, pp. 88–93.

[27] D. Cotting, M. Naef, M. Gross, H. Fuchs, Embedding imperceptible patterns into projected images for simultaneous acquisition and display, in: ISMAR '04, 2004, pp. 100–109.

[28] L. Zhang, B. Curless, S. M. Seitz, Spacetime stereo: Shape recovery for dynamic scenes, in: Proc. CVPR '03, 2003, pp. 367–374.

[29] L. Zhang, N. Snavely, B. Curless, S. M. Seitz, Spacetime faces: high resolution capture for modeling and animation, in: Proc. SIGGRAPH '04, 2004, pp. 548–558.

[30] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, in: Proc. ICCV '99, 1999, pp. 377–384.

[31] J. Sun, N.-N. Zheng, H.-Y. Shum, Stereo matching using belief propagation, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (7) (2003) 787–800.

[32] S. Birchfield, C. Tomasi, A pixel dissimilarity measure that is insensitive to image sampling, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (4) (1998) 401–406.

[33] C. Zitnick, T. Kanade, A cooperative algorithm for stereo matching and occlusion detection, Tech. Rep. CMU-RI-TR-99-35, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (October 1999).

[34] J. Sun, Y. Li, S.-B. Kang, H.-Y. Shum, Symmetric stereo matching for occlusion handling., in: Proc. CVPR '05, 2005, pp. 399–406.

[35] A. Fusiello, V. Roberto, E. Trucco, Efficient stereo with multiple windowing, in: Proc. CVPR 97, 1997, p. 858.

[36] M. Pauly, M. Gross, L. Kobbelt, Efficient simplification of point-sampled surfaces, in: Proc. Visualization '02, 2002, pp. 163–170.

[37] H. Pfister, M. Zwicker, J. van Baar, M. Gross, Surfels: Surface elements as rendering primitives, in: Proc. SIGGRAPH '00, 2000, pp. 335–342.

[38] C. Hofsetz, K. Ng, N. Max, G. Chen, Y. Liu, P. McGuinness, Image-based rendering of range data with estimated depth uncertainty, IEEE Computer Graphics & Applications 24 (4) (2005) 34–42.

[39] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, M. Gross, Post-processing of scanned 3D surface data, in: Eurographics Symposium on Point-Based Graphics '04, 2004, pp. 85–94.

[40] A. Broadhurst, T. Drummond, R. Cipolla, A probabilistic framework for the space carving algorithm, in: Proc. ICCV '01, 2001, pp. 388–393.

[41] M. Zwicker, H. Pfister, J. van Baar, M. Gross, EWA splatting, IEEE Transactions on Visualization and Computer Graphics 8 (3) (2002) 223–238.

[42] C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, Unstructured lumigraph rendering, in: Proc. SIGGRAPH '01, 2001, pp. 425–432.