Depth image-based rendering from multiple cameras with 3D propagation algorithm

Quang H. Nguyen
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
qnguyen2@illinois.edu

Minh N. Do
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
minhdo@illinois.edu

Sanjay J. Patel
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
sjp@illinois.edu

ABSTRACT

This paper proposes a new 3D propagation algorithm for the depth image-based rendering problem with multiple color and range cameras at arbitrary positions. The proposed algorithm efficiently renders novel images at arbitrary virtual views by propagating all available depth information from range cameras to color cameras, and then all available depth and color information from color cameras to the virtual views. Furthermore, the algorithm significantly enhances the propagated depth images by applying a new occlusion removal method and a new depth-color bilateral filtering. The paper also describes the parallelism structure of our algorithm and outlines a mapping onto massively parallel architectures such as general-purpose graphics processing units (GPGPUs). Experimental results show that the proposed algorithm provides good rendering quality while staying within computational bounds for real-time applications.

Keywords

Depth image-based rendering, massively parallel architecture, bilateral filtering

1. INTRODUCTION

Image-based rendering (IBR) is the process of synthesizing new "virtual" views from a set of "real" views. Because IBR provides more photorealism with fewer computations than conventional model-based rendering, it is recognized as one of the most attractive and high-potential research topics in the image processing community. It also has many interesting applications such as video conferencing, 3DTV, and telepresence. [10, 5, 2]

Thanks to the recent developments of new range sensors [1, 8] which measure time delay between transmission of a light pulse and detection of the reflected signal on an entire frame at once, per-pixel depth information can be obtained in real time from range cameras. This makes the depth IBR (DIBR) problem less computationally intense and more robust than other techniques. Furthermore, it helps significantly reduced the number of necessary cameras.

Some previous works have been proposed in DIBR literature. McMillan [6] with his warping method maps a point in an image to a corresponding point in another image at a different view as long as its depth value is known. However the work considered only single views and did not take advantage of multiple views. Besides, warping is only the first step of the synthesis work. The most difficult problem is how to deal with newly exposed areas (holes) appearing in the warped image. Some approaches to handle this problem were proposed in [3, 4]. However, these approaches considered only the 1D case where the virtual camera is forced to be on the same line with real cameras and assumed that depth images are given in the same views with color images. This assumption is not general because not all range cameras provide color information. Furthermore, standard color cameras are much cheaper and provide much higher color resolution than range cameras. So the combination of a few range cameras and many color cameras is more feasible.

Another approach which focuses on signal processing techniques is the Propagation Algorithm [7]. Using depth information, surface points that correspond to pixels in the real images are reconstructed and reprojected onto virtual views. Therefore, the real pixels are said to be *propagated* to the virtual image plane. Again, it is implied in [7] that color cameras and range cameras must have the same location and only the 1D case is considered.

Inspired from this work, we consider the generalized 3D case with separate locations for the color and depth cameras. A new occlusion removal method, depth-color bilateral filter, and disocclusion filling are applied to improve rendering quality. The proposed 3D Propagation Algorithm is a general solution for any camera configuration. In practice, the depth cameras usually provide images with lower resolution than that of the color cameras. We shows that the algorithm works well for this case by combining color information from the high resolution color images and available depth information to increase depth image resolution. Then we consider the mapping of these algorithms onto massively parallel architectures such as GPGPUs in order to facilitate real-time operations.

The remainder of the paper is organized as follows. Section 2 sets up the problem. In Section 3, the proposed algorithm is explained in detail. Experimental results are shown

Immerscom 2009, May 27-29, 2009, Berkeley, USA

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6234
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6234

in Section 4. Section 5 gives some discussions about parallelism and results. Finally, the conclusion is given in Section 6.

2. PROBLEM STATEMENT

The main goal of our proposed 3D propagation algorithm is to render a new image at an arbitrary virtual view based on images collected from multiple color cameras and range cameras. Assume that there are N color cameras and M range cameras capturing a scene in 3D space. The inputs for our algorithm are a set of color images $\{I_i(x)\}_{i=1}^{N}$, depth images $\{d_j(x)\}_{j=1}^{M}$, and parameters of range and color cameras $\{C_i(x), f_i, \vec{w}_i\}_{i=1}^{N+M}$ where C_i is camera position of the i^{th} camera, f_i is its focal length, and $\vec{w}_i$ is its normalized viewing direction vector which points from C_i to the image plane center. The output is a rendered image at virtual view $I_v(x)$. The proposed algorithm are based on two following assumptions: (i) *Calibrated cameras*: the positions, focal lengths, and viewing direction of all cameras are known; (ii) *Lambertian surfaces*: the color of a point on a surface is constant regardless the angle from which it is viewed.

The proposed algorithm is divided into two main steps:

1. *Depth propagation*: Depth information from each range camera is propagated to every color camera's image plane. Then, processing techniques including occlusion removal, depth interpolation, and disocclusion filling are applied to obtain a complete depth image at each color camera (see Figure 1).

2. *Rendering*: Depth and color information from each color camera are propagated to the virtual view, then merged and filtered to produce the output image (see Figure 2).

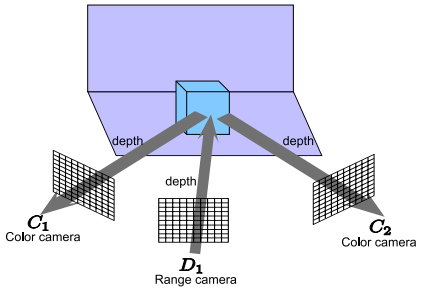


Figure 1: Depth image propagation step. Depth information is propagated from range cameras to color cameras

3. 3D PROPAGATION ALGORITHM

3.1 Depth propagation

In this section, we describe how depth information from a range camera can be propagated to a color camera. The range camera is considered as the reference view and the color camera is considered as the desired view. The block diagram of the algorithm is shown in Figure 3.

3.1.1 3D warping

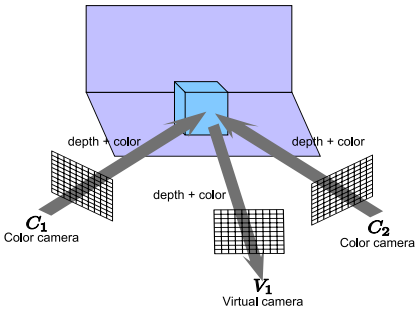


Figure 2: Rendering step. Depth and color information are propagated from color cameras to the virtual view

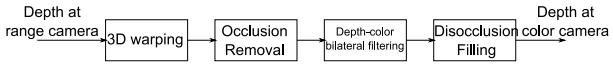


Figure 3: Depth image propagation diagram

The warping technique proposed in [6] allows us to map a point in a reference image to a corresponding point in a desired image at a different view as long as we know the depth value of that point. Consider a reference camera $\{C_r, f_r, \vec{w}_r\}$ and a desired camera $\{C_d, f_d, \vec{w}_d\}$ in a 3D Euclidian space with basis vectors $(\vec{i}, \vec{j}, \vec{k})$.

It is known that each point of a image in 2D space can be mapped one-to-one with a ray in 3D space that goes through the camera position. Given a 2D image plane with basis vectors $(\vec{s}, \vec{t})$ and a 3D space $(\vec{i}, \vec{j}, \vec{k})$, the 2D point to 3D ray mapping relation is:

$$\vec{r} = \begin{bmatrix} r_i \\ r_j \\ r_k \end{bmatrix} = \begin{bmatrix} \vec{s}_{ijk} & \vec{t}_{ijk} & f * \vec{w}_{ijk} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (1)$$

where (u, v) is the 2D coordinate of the point in the image plane; $\vec{r}$ represents the corresponding ray's direction; $\vec{s}_{ijk}$, $\vec{t}_{ijk}$ and $\vec{w}_{ijk}$ are representations of $\vec{s}$, $\vec{t}$, and viewing direction $\vec{w}$ in $\{\vec{i}, \vec{j}, \vec{k}\}$. Matrix P is called the *mapping matrix*.

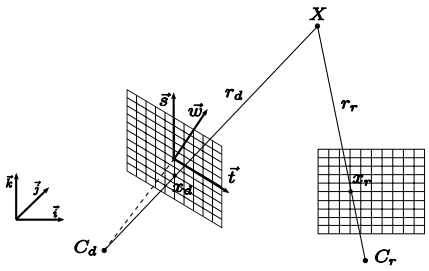


Figure 4: A point X can be warped from the reference image plane to the desired image plane

Consider a point X in 3D space $\{\vec{i}, \vec{j}, \vec{k}\}$. Let $\vec{x}_r$ and $\vec{x}_d$ be homogeneous coordinates of X in the reference image plane and the desired image plane (see Figure 4). Let P_r and P_d be mapping matrices of the reference camera and the desired camera. It was proven in [6] that the warping

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6234
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6234

equation between $\vec{x}_r$ and $\vec{x}_d$ is:

$$\vec{x}_d = P_d^{-1}\left(\frac{|P_r\vec{x}_r|}{d(\vec{x}_r)}(\vec{C}_r - \vec{C}_d) + P_r\vec{x}_r\right) \quad (2)$$

where $d(\vec{x}_r)$ is the depth value of point $\vec{x}_r$. In many-to-one mapping cases where more than one point in the reference image are mapped to the same point in the desired image, depth values of these points are compared and the one with smallest depth is kept.

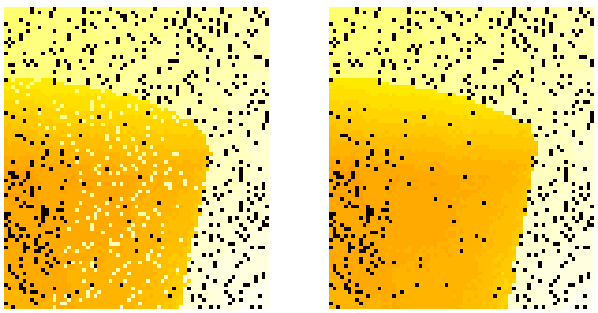
Figure 7 (a) shows the incomplete propagated depth images D_{warped} after warping. Unknown depth patches or holes, represented by black color in Figure 7 (a), are caused by two reasons. First, uniform sampling in the reference image becomes non-uniform sampling in the desired image. This means that in some certain patches, there are fewer sample points than in some other patches. Those holes can be filled by the interpolation step using depth-color bilateral filtering (section 3.1.3). Second, holes are also created when occluded areas in the reference image are revealed in the desired image, which is a disocclusion problem. It seems to be impossible to fill these holes. However, for our problem setting, we have full color information at the desired view. Therefore, these holes can be interpolated based on color image at the desired view (section 3.1.4).

In addition, as shown in Figure 5, some background sample points (in brighter color) visible in the reference depth image should be occluded by the foreground (in darker color) in the desired depth image but are still visible. That significantly degrades the interpolation quality. Therefore, an occlusion removal step is necessary to correct those points.

3.1.2 Occlusion Removal

The key idea of this occlusion removal method is based on the smoothness of surfaces. If a point A in D_{warped} is locally surrounded by neighboring points whose depth values are σ smaller than the depth of A, then A is selected to be occluded by the surface composed of those neighbors.

In our implementation, for every point A, a $(2w+1)\times(2w+1)$ window whose center is A is divided into four partitions as shown in Figure 6. If there are at least three of those partitions each of which has at least one point B such that $depth(A) - depth(B) > \sigma$, then point A is decided to be occluded and its depth is replaced by a new interpolated value. Figure 5 shows a patch of D_{prop} before and after occlusion removal step respectively. In our experiment, we choose $w = 3$.



(a) Before occlusion removal

(b) After occlusion removal

Figure 5: A patch in the propagated depth image before and after the occlusion removal step.

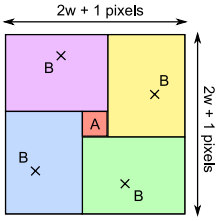


Figure 6: Partitions in the occlusion removal window.

3.1.3 Depth-color bilateral filtering (DCBF)

Bilateral filtering [9] is a simple, non-iterative scheme for edge-preserving smoothing. It is a combination of a spatial filter, whose weights depend on Euclidian distance between samples, and a range filter, whose weights depend on differences between values of samples. In this paper, by integrating known depth and color information, the proposed DCBF effectively interpolates unknown depth pixels in D_{prop} caused by non-uniform resampling while keeping sharp depth edges. The DCBF is defined as following:

$$d_A = \frac{1}{W_A}\sum_{B\in S_A} G_{\sigma_s^2}(|\vec{x}_A - \vec{x}_B|)\cdot G_{\sigma_r^2}(|I_A - I_B|)\cdot d_B \quad (3)$$

d_A : depth value of point A.
I_A : color value of point A
$\vec{x}_A = [u_A, v_A]$: 2D coordinate of point A
S_A : set of A neighboring points.
$G_\sigma(|\vec{x}|) = exp\left(-|\vec{x}|^2/2\sigma^2\right)$: Gaussian kernel with variance σ^2.

The idea of using color differences as a range filter to interpolate depth value is based on the observation that whenever a depth edge appears, there is almost always a corresponding color edge due to color differences between objects or between foreground and background. The DCBF also works well with textured surfaces since it counts only pixels on that surface which have similar color to the interpolated pixel. If surfaces have the same color, then color does not give any new information and the DCBF works as a simple interpolation scheme such as bilinear or bicubic. Figure 7 (b) shows the obtained depth image after the DCBF step.

3.1.4 Disocclusion filling

In order to fill holes caused by disocclusion, the DCBF can also be used, but it needs to follow a specific direction. Otherwise, filtering is performed from all directions, and incorrect depth values may be obtained. As described in Figure 8, filling from left-to-right causes incorrect interpolated depths because the holes should be filled with the background depth whereas all neighboring known depth values belongs to the foreground. We observed that, in the 2D image plane, the disocclusion holes always appear toward the projection of the reference camera position onto the desired image plane (epipole point). This fact is reasonable since the disocclusion holes are caused by the change of the camera position moving from one view to another view. So there must be a relationship between the disocclusion holes and the camera position. The filling direction can be decided correctly based on this relation. The epipole $\bar{e}$ can be computed as following:

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6234
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6234

(a)

(b)

(c)

Figure 7: (a) Incomplete propagated depth images at color view 1; (b) Propagated depth image after applying depth-color bilateral filtering (DCBF); (c) Complete propagated depth image after disocclusion filling

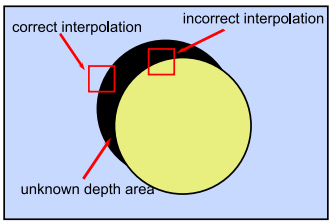


Figure 8: Disocclusion filling.

$$\begin{bmatrix} e_x & e_y & e_z \end{bmatrix}^T = P_d^{-1}(\vec{C}_r - \vec{C}_d) \quad (4)$$

$$\vec{e} = (e_x/e_z, e_y/e_z) \quad (5)$$

where $\vec{C}_r$ and $\vec{C}_d$ are positions of the reference and desired views, P_d is the mapping matrix of the desired view. Then the filling direction is a vector pointing from the epipole to the center of the desired depth image. For example, if the epipole lies in the top left quadrant of the image, the filling should start from the top left corner. Figure 7 (c) shows the complete propagated depth image after filling disocclusion areas.

3.1.5 *Case of low resolution depth images*

In practice, most of depth cameras (i.e. Canesta, PrimeSense,...) provide the depth images with lower resolution than that of the color images. For this case, the available depth information is first propagated to the color camera. Then the DCBF step calculates unknown depth pixels at the color image plane based on color information from the high resolution color image. As discussed in section 3.1.3, an important advantage of the DCBF filter is its ability to combine between color and depth information for edge preserving interpolation. Therefore, the proposed algorithm can work well with different resolution settings of the cameras.

3.2 Rendering

Now each color camera has both depth and color information. The last step is to propagate this information to the virtual camera. The color cameras become the reference views and the virtual camera becomes the desired view. This process is quite similar to the first part of the algorithm. Using the warping equation (1), color pixels from each reference view are propagated to the desired view and are merged together. Then a bilateral filter is applied to fill unknown color pixels and smooth surfaces. Note that most of the unknown color pixels in this step are caused by non-uniform resampling since the color cameras are intentionally installed in a way to capture the whole scene from different views and, therefore, reduce as much as possible holes caused by disocclusion. The complete rendered image is shown in Figure 9 (d).

4. EXPERIMENTAL RESULTS

For the experiments, we adopted a synthesis configuration with two color cameras and one range cameras. The input images are shown in Figures 9 (a), (b) and (c). The resolution is 800x600 pixels. The window size for occlusion removal step in Section 3.1.1 is 7x7. Some parameters fro the DCBF in Section 3.1.3 are set as following: $\sigma_s^2 = 3$, $\sigma_r^2 = 0.01$, kernel window size = 11x11. The result images

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6234
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6234

Table 1: Comparison in the running time of the 3D warping, occlusion removal, and DCBF steps

Mode	Frame rate (in fps)	Time (in msec)
Sequential	0.09	11 389.00
Parallel	24.37	41.03

for two different virtual views are shown in Figure 9 (d). The black area around the boundary of these images is due to the fact that there is completely no information (both depth and color) from the input images corresponding to those areas.

5. MAPPING TO MASSIVELY PARALLEL ARCHITECTURES

A major advantage of the proposed algorithm is that it can be easily mapped onto massively parallel architectures such as GPGPUs. In this section, we briefly describe the parallelism of each processing step of our algorithm.

The 3D warping, occlusion removal, and DCBF steps are purely parallel as each pixel in the desired view can be computed independently. Copying the depth values in the reference view to appropriate pixels in the desired view seems to be not trivial for the parallelism since, at some pixels, this is not a one-to-one mapping. However, either atomic memory updates or Z-buffering hardware on modern GPGPUs can resolve this problem. The disocclusion filling step in Section 3.1.4 is sequential because calculating unknown depth information is dependent on previously interpolated values. However, this dependence exists only on 1D lines emanating from the epipole point, and thus the problem can be expressed as a parallel set of 1D filters. In other words, the filling can be implemented parallel by DCBF on a set of 1D lines. Finally, the rendering step is quite similar to the first part of the algorithm and, hence, suits well with the parallelism.

In order to check the efficiency of the parallelism, we implemented the 3D warping, occlusion removal, and DCBF steps in two modes: sequential mode and parallel mode. The experiment was run on Intel Core2 Duo E8400 3.0GHz for the CPU and NVIDIA GeForce 9800GT for the GPU. The result in Table 1 shows that the parallel mode is about 277 times faster than the sequential mode. Note that the running time of the algorithm mainly depends on the image resolution, not on the complexity of the scene. Therefore, the obtained result can be approximated for other examples with the same size. Furthermore, we believe that more speedup rate can be obtained with more careful optimization of parallel kernels.

6. CONCLUSION

In this paper, we present a new 3D propagation algorithm for the depth IBR problem with multiple color and range cameras at arbitrary positions to render an arbitrary virtual view. The depth information from range cameras is first propagated to the color views, then both color and depth information are propagated to the virtual view. The algorithm provides an excellent improvement of rendering quality while staying within computational bounds for real-time applications. Furthermore, we also show that the algorithm can be implemented in parallel and efficiently run on GPGPUs. Future studies will focus on parallel implementation and optimization for real time purpose.

7. ACKNOWLEDGMENTS

This work was funded by the Vietnam Education Foundation (www.vef.gov) and the Universal Parallel Computing Research Center at the University of Illinois at Urbana-Champaign. The Center is sponsored by Intel Corporation and Microsoft Corporation.

8. REFERENCES

[1] Canesta. *http://www.canesta.com/*.

[2] S. C. Chan, H. Y. Shum, and K. T. Ng. Image-based rendering and synthesis. *IEEE Signal Processing Magazines*, pages 22–33, Nov 2007.

[3] W.-Y. Chen, Y.-L. Chang, S.-F. Lin, L.-F. Ding, and L.-G. Chen. Efficient depth image based rendering with edge dependent depth filter and interpolation. *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1314–1317, Aug 2005.

[4] I. Daribo, C. Tillier, and B. Pesquet-Popescu. Distance dependent depth filtering in 3D warping for 3DTV. *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 312–315, 2007.

[5] C. Fehn, R. Barré, and S. Pastoor. Interactive 3DTV: Concepts and key technologies. *Proc. of the IEEE. Special Issue on 3D Technologies for Imaging and Display*, pages 524–538, Mar 2006.

[6] L. McMillan. An image-based approach to three dimensional computer graphics. *PhD dissertation, University of North Carolina at Chapel Hill*, 1997.

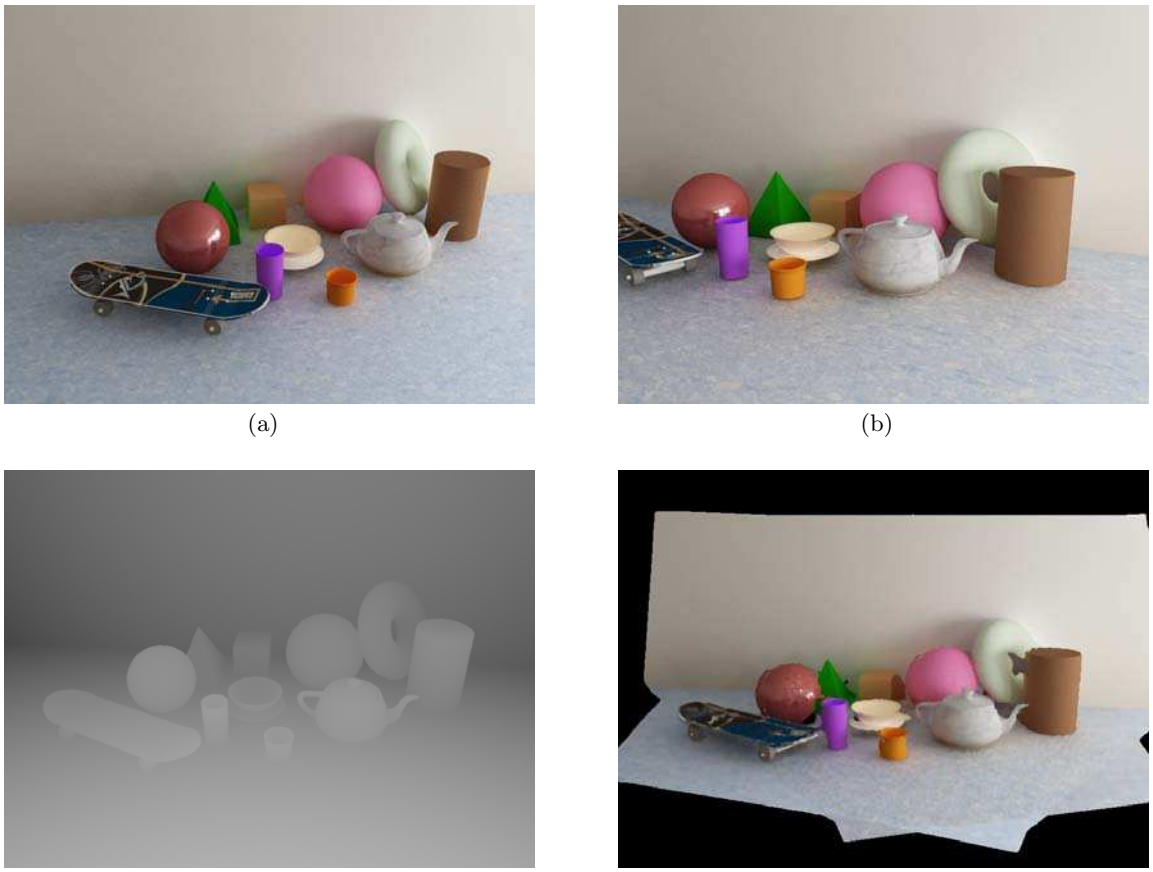
[7] H. T. Nguyen and M. N. Do. Image-based rendering with depth information using the propagation algorithm. *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Mar 2005.

[8] PrimeSense. *http://www.primesense.com/*.

[9] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proc. of the IEEE International Conference on Computer Vision*, Jan 1998.

[10] C. Zhang and T. Chen. A survey on image based rendering: representation, sampling, and compression. *EURASIP Signal Processing:Image Communication*, pages 1–28, Jan 2004.

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6234
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6234



(a) (b) (c) (d)

Figure 9: (a)&(b) Input images at color view 1 & 2; (c) Input depth image at the depth view; (d) Rendered image at a virtual view.

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6234
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6234