Depth Camera to Generate On-line Content for Auto-Stereoscopic Displays

François de Sorbier* Yuko Uematsu† Hideo Saito‡

Graduate School of Science and Technology
Keio University
Japan

Abstract

Currently, content for stereoscopic and auto-stereoscopic displays is mainly generated from videos or from synthetic data. Very few examples of applications for 3D displays mixing these both approaches are available because processing time and multi-view rendering are important constraints. In this paper, we present a capture system based on a depth camera and a color camera that is suitable for augmented reality on 3D display. The key-point of our approach is to be able to generate in real time an exact 3D model of a scene thanks to a time of flight camera. This mesh is then used to compute interactions between the real scene and virtual objects. Moreover, several images from new viewpoints are produced from that result and are displayed on a auto-stereoscopic display in real time.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—; I.3.3 [Computer Graphics]: Picture/Image Generation—; I.4.8 [Image processing and computer vision]: Scene Analysis—

1 Introduction

Integrating virtual elements in a real environment can be a complex problem especially when the result have to be displayed on a stereoscopic display. Mainly, real time is difficult to maintain since the computational time to generate the two input images required by the 3D display is too high. This becomes more problematic with an auto-stereoscopic screen that needs from 5 to 65 input images. Moreover, this latest family of displays can also require the use of many capture devices, which can be expensive and difficult to manage. In that sense, some image based rendering or depth image based rendering solutions have been proposed [5, 14, 18, 23], but produce results with an average quality or can't reach real-time.

Knowing the depth of a given environment in real-time is a very useful information especially for the purpose of multi-view rendering. A standard depth camera is then suitable in such situation, but doesn't provides the corresponding color image. In a previous work, Bartczack *et al.*[1] proposed a system based on a time-of-flight camera, a color camera and a fish-eye camera. They generate a mesh from the depth values transmitted by the depth camera and use the result to resolve the mismatches between the viewpoints of the color camera and depth camera. The role of the fish-eye camera is to initially capture and generate a static mesh of the background in order to reduce the occlusions. Authors use the final result for a mixed reality application by adding static virtual objects in the scene.

Our capture system is also based on such kind of depth camera. The color information is captured thanks to another high resolution

*e-mail: fdesorbi@hvrl.ics.keio.ac.jp
†e-mail: yu-ko@hvrl.ics.keio.ac.jp
‡e-mail: saito@hvrl.ics.keio.ac.jp

20th International Conference on Artificial Reality and Telexistence (ICAT2010), 1-3 December 2010, Adelaide, Australia
ISBN: 978-4-904490-03-7 C3450 ©2010 VRSJ

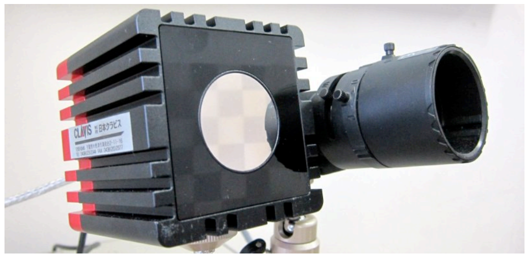
Figure 1: A color camera is added in the system since our depth camera cannot capture the color information.

camera located next to the depth camera. Then, the system generates a 3D model of a given scene in real time that helps to resolve the mismatches between the both cameras' viewpoints. Moreover, the mesh is also used to compute interactions between the real scene and virtual objects and to generate the multiple views required by an auto-stereoscopic display.

This paper is structured as follows. We start by giving an overview of the depth and color cameras based system that takes advantage of a 3D-mesh to correct the mismatches between the two viewpoints. Then, we present our method to generate the multiple views of the captured scene in order to be displayed on a 3D display. In the next section, we describe our approach to compute interactions between the real scene and virtual objects by using our capture system.

2 The Depth Camera

2.1 Description

The depth map corresponding to a given environment can be obtained by using different techniques. Several previous works introduced depth map generated from one or several color cameras. Depth estimation based on a single camera is often considered as non-accurate and requiring higher computation costs. Those algorithms take advantage of depth cues such as shading, linear perspective, defocus, texture [10, 6, 8]. Such approaches are suitable for pre-processing a video in order to convert it into a stereoscopic one. In that case, the result is often enhanced thanks to a manual operation.

With two or more cameras, the accuracy of the result and computational time depends on the algorithm used. Preferred approaches are based on the computation of a disparity map between two views [16] or the study of the motion thanks to an optical flow method [11] for example.

The recent depth cameras are a compromise between the number of required devices and the accuracy of the result. The main technology used for depth cameras is named Time-Of-Flight or TOF [9]. It is made of an illumination unit (LEDs) and a capture unit (CCD/CMOS sensor). A light pulse (IR) is emitted, reflected by the objects located in the scene and come back toward the sensor of the camera. The time corresponding to the travel of the light is precisely computed and used to evaluate the depth value of each pixel. The Swiss Ranger SR4000 [15] time-of-flight camera can generate a depth map in real-time, and can also provide a gray-scale amplitude image, a confidence map and the spatial coordinate associated with a depth value for each pixel of the sensor.

However, the color information is not generated by most of the depth camera systems based on the TOF technology.

2.2 Adding color information

The color information is obtained by adding a color camera besides the depth camera as depicted in Fig. 1. The viewpoint of both cameras is different, so the depth image has to be transformed to match the color camera's viewpoint. This is composed of two steps : a calibration stage and a mapping stage.

The Swiss Ranger depth camera provides a spatial coordinate for each depth value. In our approach, we take advantage of these data to generate a mesh of the captured scene. If those data are not generated by the depth camera, they can be obtained if each depth values is multiplied with intrinsic parameters of the depth camera. Then, we project the color image onto that mesh and render it from color camera's viewpoint to obtain the corresponding depth information. We chose to achieve the rendering from the color camera's viewpoint because the resolution of the depth camera is low and we also want to maintain the quality of the color image that has an important role in scene perception [21]. Our mesh based approach can easily take advantage of the graphic card (GPU) and rendering process of multiple views can then be achieved in real-time.

2.2.1 Calibration

The goal of the calibration stage is to get the position estimation of the color camera (extrinsic parameters) according to the depth camera's parameters. We assume that depth and color cameras are fixed together, so the calibration stage should have to be evaluated just once.

The pose estimation of depth and color cameras has to be evaluated in the same coordinate system in order to be able to project the color image onto the mesh generated from the depth map. As explain in the previous sub-section, the depth camera provides the corresponding 3D coordinates for each pixel of the depth image. This set of points is defined in a coordinate system wherein the depth camera's position is at the origin. Following this statement, the depth camera is also set as the origin of our capture system. Then, the calibration stage only requires to estimate the extrinsic parameters of the color camera.

Using a set of 2D/3D correspondences is a common way to estimate the pose of a camera with OpenCV for example (Fig. 2). First, 2D correspondences are found between the depth (using the gray-scale amplitude image) and the color image thanks to a chessboard pattern or by clicking several pixels. Second, since a 3D coordinate is defined for each pixel of the images generated by the depth camera, a list of 2D/3D correspondences between the color image and the 3D space is created. The confidence map provided by the depth camera is used to remove the correspondences that may have inaccurate 3D coordinate.

2.2.2 Mapping

In order to compute the depth map corresponding to the viewpoint of the color camera, our objective is to take advantage of a mesh generated from the 3D coordinates provided by the depth camera. The mesh can also be generated from the depth by using intrinsic parameters of the depth camera. In that case, a unit vector is computed for each pixel of the depth image and scaled according to the corresponding metric depth value to get the 3D coordinate .

The relation among the depth camera, the color camera and the mesh is depicted in Fig. 3. A computer graphics based approach

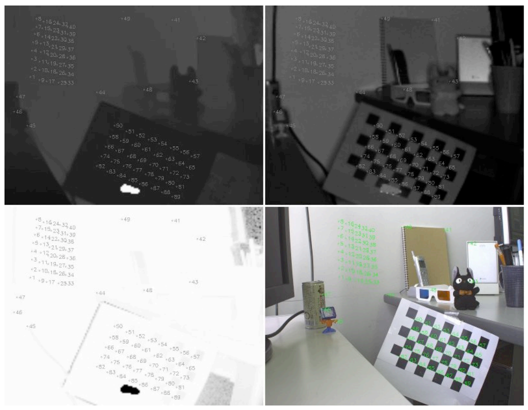

Figure 2: Four images are used for the calibration. (A) the depth map, (B) the corresponding gray scale image, (C) the confidentiality map and (D) the color Image.

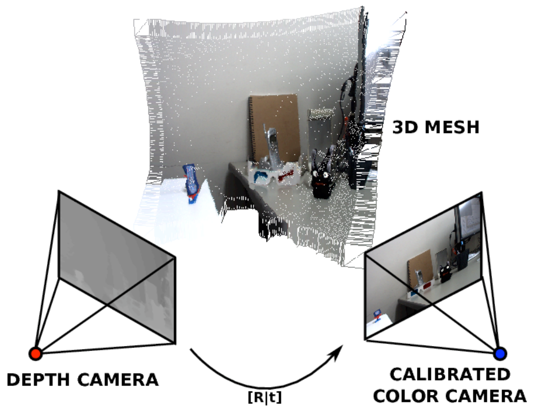

Figure 3: Our approach uses a 3D mesh to generate the depth map corresponding to the color cameras viewpoint.

significantly reduces the processing time in comparison to a standard Depth Image Based Rendering (DIBR) approach.

The intrinsic and extrinsic parameters, computed during the calibration stage, are used to set up the position of the viewpoint for the rendering stage. The mesh, made of triangles and defined in the coordinate system of the depth camera, is then observed from the viewpoint of the color camera. The mesh is finally rendered with that setup.

The rendering process will also automatically generate a depth map but the computed z-values are not linear due to the OpenGL non-linear transformation. A GPU-based program is then used to convert this depth map into a metric depth information. This operation requires to get the position of the vertices in the camera coordinate system (MODELVIEW transformation) and to quantify it between 0. and 1.0 according to a given maximal depth value.

A solution to map the color image on the mesh is to apply a projective texture technique [4] that generates automatically a texture coordinate for each vertex of the mesh. This method requires to define a projective texture matrix as follows:

$$\mathbf{M}_{texture} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & 0.5 & 0 & 1 \end{bmatrix} \times \mathbf{M}_{projection} \times \mathbf{M}_{modelview} \quad (1)$$

Where $\mathbf{M}_{projection}$ and $\mathbf{M}_{modelview}$ are matrices defined according to the converted intrinsic and extrinsic parameters of the color

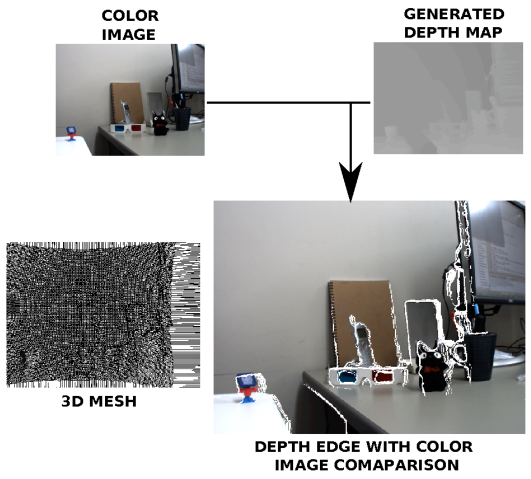

Figure 4: Comparison between the edges color image and its corresponding depth-map. Errors are mainly located in occlusions and close areas. Especially, the screen on the left part is not visible from the depth camera, so depth information doesnt match in this area.

camera. The texture coordinates are obtained by multiplying each vertex of the mesh by this matrix using the GPU for example. Then, the color information will be correctly mapped on the mesh, even if the viewpoint is modified.

2.2.3 Results

In our approach, occlusions are reduced because we are using only one mesh in which all the points are connected with triangles. So, occlusions are replaced by an automatic interpolation between two different depths. However, occlusions also exist in different parts of the border of the image, but contrary to the previous case, the mesh is not defined for those areas. Our solution is to extrude the borders of the mesh based on the depth of the points located on the borders of the mesh. However, the extrusion, presented in the bottom left of Fig. 4 (right part of the mesh), will generate unstable flat areas in the depth map because depth information located on the border of the image are less accurate.

A matching between the color image and the edges extracted from the depth map is presented in Fig. 4 and shows that errors are mainly located in the previously described flat areas, in areas close from the depth camera, on specular objects and on occlusion areas. With our approach, the depth map corresponding to the color camera's viewpoint is generated in real time. The rendering is real-time (35 frames per second) and only limited by the frame-rate of the cameras.

3 Input format for Auto-Stereoscopic displays

Auto-stereoscopic displays are all based on the same principle: they receive several input images captured from slightly different viewpoints and display them thanks to a special filter that will spread images over the field of view. So, if the user is well located in front of the display, each eye can see a single specific image.

Each auto-stereoscopic screen has its own format for input images, but we can categorize two main families: the 2D plus depth format and the sub-pixel alignment Format.

3.1 2D-plus-Depth Format

The 2D plus Depth Format, introduced by Fehn [5], consists of a color image associated with its corresponding depth map. Both information are combined to generate new images from slightly different viewpoints. Occlusions are resolved by interpolating existing values or by using methods like in-painting. The main advantage of this format is to reduce the bandwidth because only two images are transmitted.

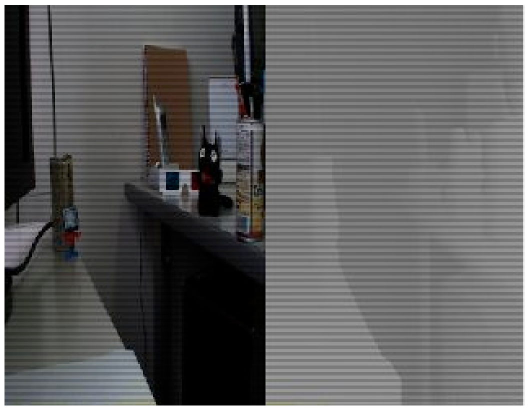

Figure 5: Input image based on the 2D plus depth format used by the Philips auto-stereoscopic screen.

The 2D-plus-Depth input format was selected by *Philips* [17] to integrate the auto-stereoscopic displays. Users have to provide a color image with its corresponding depth map, but can also add an image of the background and a mask of the foreground in order to reduce occlusion problems. Hardware integrated in the screen generates the new nine required views.

Our approach can easily generate the color image with its corresponding depth map in real time. Depth information is easily and quickly mixed with the color image map by using the GPU to fit the requirement of the 2D plus depth format. An example of input image is presented in Fig. 5.

3.2 Sub-pixel Alignment Format

Another kind of input requires to provide directly all the views to the screen. All these images have to be regrouped into a single big image to be displayed on the screen. However, the organization of the images cannot be reduced to a simple interleaving or juxtaposition because it will increase the horizontal resolution of the final image while the vertical one will remain the same. This limitation is commonly resolved by proposing a particular sub-pixel alignment of the different images into a single one [20].

So, we first need to generate the new views by using our mesh based approach. a 3D mesh reduces significantly the complexity of the multi-view rendering because we can use the principle of stereoscopic rendering algorithm in the same way than computer graphics. It consists of a translation of the viewpoint along a specific axis according to the eye separation distance and the view direction. This axis is extracted from the first column vector of the OpenGL MODELVIEW matrix, which corresponds to the right vector of the color camera's system coordinate. An example of result obtained with our approach and using a sub-pixel alignment is presented in Figure 6.

However, multi-view rendering is often considered as a slow process since each view requires a specific rendering pass. Computational time can be very high, especially when the geometry is made of thousands of triangles (25000 in our case). A method introduced by de Sorbier *et al.* [3] can be used to speed-up the process thanks to a GPU-based multi-view rendering algorithm.

4 Interactions with Virtual Objects

In previous sections, we explained that the result was based on of a mesh generated from the depth information. The benefit of this approach is to speed-up and simplify the multi-view rendering for

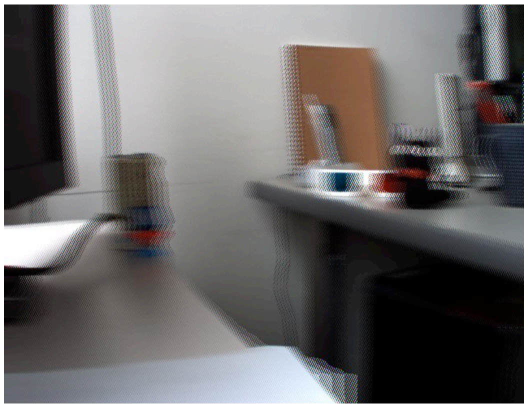
Figure 6: Input image based on a sub-pixel alignment used by the Tridelity auto-stereoscopic screen. This image is the result of a combination of five different views.

auto-stereoscopic displays. In this section, we present an application of our capture system for augmenting a real scene with virtual objects that also takes advantage of the mesh. In particular, it becomes possible to easily interact with the virtual objects.

4.1 Interactions

Generic approaches to augment a real scene with virtual elements [7] use the geometric context extracted form captured images. Specific objects are tracked over the time making possible to evaluate their position and orientation. The knowledge of such characteristics are very important in order to correctly associate virtual elements with those objects. However, it often increases the computation cost and is not always suitable for on-line applications.

A survey introduced by Kolb *et al.* [12] describes several applications using time-of-flight cameras dedicated to user interaction and user tacking. The depth map is analyzed to estimate position of different parts of the user's body. This approach is used with interactive screens for which no specific device are required to manipulate virtual content.

Bartczak *et al.* [1] presented an application of time-of-flight camera for mixed-reality. They integrate virtual objects in real scene by taking also advantage of a mesh generated from depth information but did not propose any kind of interaction. In that approach, the position of virtual objects is manually set, so there are not taking advantage of the geometry to automatically setup the position of the virtual objects.

The mesh generated from the depth camera has two benefits. First, occlusions between the real scene and virtual objects are easily resolved thanks to the depth test applied during the OpenGL rendering stage. Second, physical interactions between virtual and real objects are computed in real-time thanks to the mesh we are using to simulate the real environment. So, our approach is then completely similar to a standard computer graphics application.

In computer graphics, a common approach to generate interactions is to integrate a physics engine. We decided to use such technique into our system. Our mesh is dynamic, meaning it is updated for every new frame, but most of existing libraries require a static mesh to speed up the computation of collisions or insure the stability of the computation. *Bullet* [2] is the only open-source library suitable for that purpose. It computes a bounding box for each independent object and evaluates collisions between each boxes. If a collision is detected, the physical engine computes more precisely a possible collision among the triangles of the meshes. The physics engine also provides tools to compute the reactions of objects after a collision and then to produce a realistic behave.

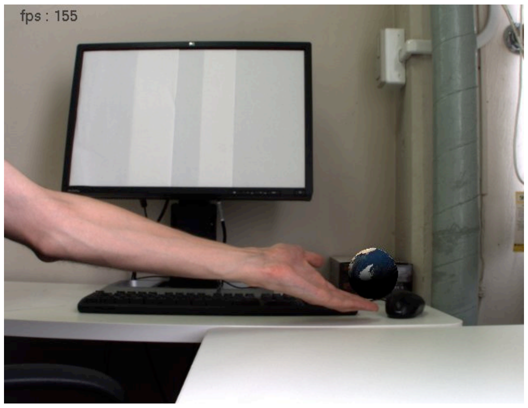

Figure 7: Our application proposes an interaction with a virtual ball in real-time.

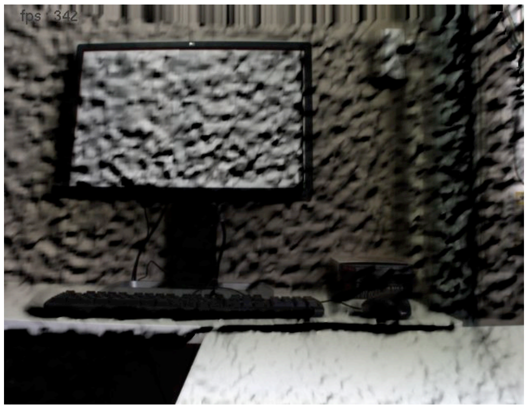

Figure 8: The virtual lighting of the mesh generated by the depth camera gives a poor quality result because a the instability of the mesh.

However, the mesh created by the depth camera is updated every frame and looks slightly unstable because of the accuracy of the capture device (few centimeters). This means that a surface that should be flat will appear with wavelets evolving in the time. So, an object that should lay on a surface will have shake because of the instability of the mesh. A solution could be to detect and replace flat surfaces with a new mesh.

4.2 Virtual lighting

An important point for augmenting a real scene with virtual objects is lighting. The goal is to illuminate the virtual objects according to the position of the real light source or to the position of a new virtual light source.

This approach requires to provide a normal for each point of the mesh in order to obtain a smoother lighting. We first define the normal of each vertex to zero. Then, we compute a normal for each triangle and add it to the three vertices belonging to the triangle. Finally, after computing the normals for each vertex, they are normalized. Lighting is computed with standard OpenGL methods but the instability of the mesh described in the previous sub-section reduces significantly the quality of the result as depicted in Fig. 8.

Another effect of lighting is shadow casting. A previous work [19] already introduced the use of shadows in mixed reality by using a depth camera, but uses a static light map that does not evolve in time and concerns only virtual objects. In our approach, we are using dynamic objects so we need to compute shadows for

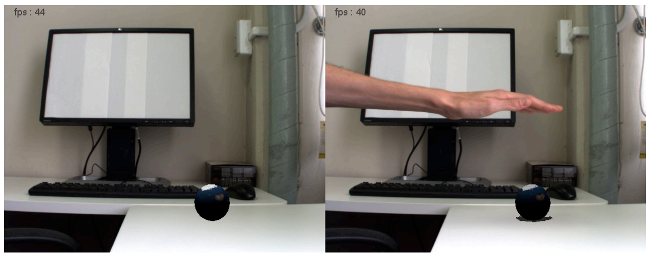

Figure 9: On the left image: the augmented scene without shadows. On the right image: real objects cast shadows on virtual objects and *vice-versa*.

every frame. Moreover, we use the mesh generated from the real scene to cast shadows on virtual objects.

We use the algorithm of shadow mapping [22], which doesn't rely on the complexity of the geometry complexity and is suitable for real-time applications. The basic concept of the shadow mapping method is to compare depth of a same point from the viewpoint and from the light source position. If the depth value is different, then it means that an occlusions exists and that the color of the point corresponding to that depth value should be altered to simulate the shadow. An example of result is presented in Figure 9.

4.3 Results

We experimented our algorithm on a bi-Xeon 2,5Ghz running Linux. The graphic card is a nVIDIA GeForce GTX 285 with 1Go memory. We used a resolution of 800×600 for the rendering stage.

With physical interactions and shadow casting, we obtain an average frame-rate of 45 frames per second. The frame-rate decreases to 35 frames per second when we add the multiple view rendering (5 views) for auto-stereoscopic displays.

5 Conclusion

We have presented a depth-camera based capture system that is used to create an augmented reality application. A 3D mesh is generated in real-time and is transformed to match the viewpoint of the color camera. Then, the mesh is used to create interactions between virtual objects with the real environment in real time. Finally, several images are rendered from different novel viewpoints in real time and used on auto-stereoscopic displays. Depth cameras still remain expensive, but coming devices like [13] could change this problem.

We explained that interactions between virtual elements and the real scene suffer of the instability of the mesh. So, for example, a virtual object that should lay on a flat surface will have small unexpected movements, or virtual relighting of the real scene will be of poor quality. In future works, we expect to improve the result with flat surfaces detection that can be replaced in the mesh.

The result of multiple view rendering and interactions can also be improved by reducing the occlusion problems of our approach. For example, we plan to add one more depth camera and/or another color camera or to segment the mesh into several layers.

Acknowledgements

This work is partially supported in part by a Grant-in-Aid for the GCOE for high-Level Global Cooperation for Leading-Edge Platform on Access Spaces from MEXT, Japan. This research is partially supported by National Institute of Information and Communications Technology, Japan.

References

[1] B. Bartczak, I. Schiller, C. Beder, and R. Koch. Integration of a time-of-flight camera into a mixed reality system for handling dynamic scenes, moving viewpoints and occlusions in real-time. In *Proceedings of the 3DPVT Workshop, Atlanta, GA, USA (June 2008)*. Citeseer.

[2] Bullet Physics Library. http://bulletphysics.org/. website, 2010.

[3] F. de Sorbier, V. Nozick, and H. Saito. Gpu-based multi-view rendering. In *Computer Games, Multimedia and Allied Technology (CGAT 2010)*, April 2010.

[4] C. Everitt. Projective texture mapping. *White paper, NVidia Corporation*, 2001.

[5] C. Fehn. A 3D-TV approach using depth-image-based rendering (DIBR). In *Proc. of VIIP*, volume 3, 2003.

[6] P. Grossmann. Depth from focus. *Pattern Recognition Letters*, 5(1):63–69, 1987.

[7] M. Haller, M. Billinghurst, and B. Thomas. *Emerging technologies of augmented reality: interfaces and design*. Igi Publishing, 2007.

[8] R. Haralick. Using perspective transformations in scene analysis. *Computer Graphics and Image Processing*, 13(3):191–221, 1980.

[9] G. Iddan and G. Yahav. 3D Imaging in the studio (and elsewhere). In *Proc. SPIE*, volume 4298, pages 48–55, 2001.

[10] B. Kim and P. Burger. Depth and shape from shading using the photometric stereo method. *CVGIP: Image Understanding*, 54(3):416–427, 1991.

[11] J. Kim, J. Kim, D. Lee, and K. Cho. 3D depth estimation for target region using optical flow and mean-shift algorithm. In *International Conference on Control, Automation and Systems, 2008. ICCAS 2008*, pages 34–39, 2008.

[12] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-Flight Cameras in Computer Graphics. In *Computer Graphics Forum*, volume 29, pages 141–159. John Wiley & Sons, 2010.

[13] Microsoft Kinect. http://www.xbox.com/en-us/kinect/. website, 2010.

[14] V. Nozick and H. Saito. Multiple view computation for multi-stereoscopic display. In *IEEE Pacific-Rim Symposium on image amd video Technology (PSIVT 2007*, volume 4872, pages 399–412, 2007.

[15] T. Oggier, B. Buttgen, F. Lustenberger, G. Becker, B. Ruegg, and A. Hodac. Swissranger sr3000 and first experiences based on miniaturized 3d-tof cameras. *Proc. of the First Range Imaging Research Day at ETH Zurich*, 2005.

[16] T. Ozanian. Approaches for stereo matching: A review. *Modeling, identification and control*, 16(2):65–94, 1995.

[17] Philips. Wowvx for amazing viewing experiences. *Philips 3D solutions*, 2006.

[18] A. Redert, M. de Beeck, C. Fehn, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, I. Sexton, and P. Surman. ATTEST: Advanced three-dimensional television system technologies. 2002.

[19] I. Schiller, B. Bartczak, F. Kellner, J. Kollmann, and R. Koch. Increasing realism and supporting content planning for dynamic scenes in a mixed reality system incorporating a time-of-flight camera. In *5th European Conference on Visual Media Production (CVMP 2008)*, pages 1–10, 2008.

[20] Tridelity. http://www.tridelity.de/. website, 2010.

[21] T. Troscianko, R. Montagnon, J. L. Clerc, E. Malbert, and P.-L. Chanteau. The role of colour as a monocular depth cue. *Vision Research*, 31(11):1923 – 1929, 1991.

[22] L. Williams. Casting curved shadows on curved surfaces. *ACM SIGGRAPH Computer Graphics*, 12(3):270–274, 1978.

[23] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *ACM SIGGRAPH 2004 Papers*, pages 600–608. ACM, 2004.