Real Time Rendering for Multiview Autostereoscopic Displays

R-P. M. Berretty[a], F.J. Peters[a] and G.T.G. Volleberg[b],

[a]Philips Research, Prof. Holstlaan 4, 5656AA Eindhoven, Netherlands;
[b]Philips Applied Technologies, Glaslaan 2, 5616 LW Eindhoven, Netherlands

ABSTRACT

In video systems, the introduction of 3D video might be the next revolution after the introduction of color. Nowadays multiview autostereoscopic displays are in development. Such displays offer various views at the same time and the image content observed by the viewer depends upon his position with respect to the screen. His left eye receives a signal that is different from what his right eye gets; this gives –provided the signals have been properly processed– the impression of depth. The various views produced on the display differ with respect to their associated camera positions.

A possible video format that is suited for rendering from different camera positions is the usual 2D format enriched with a depth related channel, e.g. for each pixel in the video not only its color is given, but also e.g. its distance to a camera.

In this paper we provide a theoretical framework for the parallactic transformations which relates captured and observed depths to screen and image disparities. Moreover we present an efficient real time rendering algorithm that uses forward mapping to reduce aliasing artefacts and that deals properly with occlusions. For improved perceived resolution, we take the relative position of the color subpixels and the optics of the lenticular screen into account. Sophisticated filtering techniques results in high quality images.

Keywords: display algorithms, rendering, depth, 3D, real-time, video format

1. INTRODUCTION

When a human sees the real world, his two eyes see a slightly different image. In the brain, the two images are combined into one model. The difference between the left and the right image contributes to the experience of depth.

Nowadays, *autostereoscopic* displays exist that offer various views at the same time. The view observed depends upon the viewing position, allowing for distinct views to be observed by the two eyes of an observer. Examples of autostereoscopic displays are Philips lenticular multiview 3D displays,[1] Sharp two view displays,[2] and Opticality barrier displays.[3]

A *multiview display* is an autostereoscopic display, capable of showing multiple images, e.g,. eight, or nine, for different (horizontal) viewing directions. Note that if the screen accommodates more than just two views the user may see parallax by moving his head sideways. Moreover, multiview displays can be watched by more than one viewer simultaneously.

Various multiview technologies exist. One such technology is lenticular based. The principle of such a lenticular multiview display is depicted in Figure 1(a). In front of a conventional LCD, cylindrical lenses are attached. The lenses accommodate that for a specific viewing angle, the viewer only sees a subset of the subpixels of the underlying LCD. The lenses are slanted to optimize the usage of horizontal and vertical resolution of the underlying display panel. In Figure 1(b), for a given viewing direction, the subpixels which are visible are highlighted.

In this paper, we focus on rendering for multiview displays that support *horizontal disparities*, i.e. horizontal parallax only; in other words we consider transformations in which objects are displaced only horizontally, dependent on their depths.

There are two main video formats to represent 3D video: encoding based on two streams (one for the left eye, and one for the right eye), and encoding based on image plus depth. The latter format, also called the 2.5D video format,[4] has several advantages. It it is device independent (can be used for n-view systems,without first specifying the number of views) and it is less bandwidth demanding. In this paper, we demonstrate that high image quality can be achieved when using the image plus depth format for 3D (multiview) video systems.

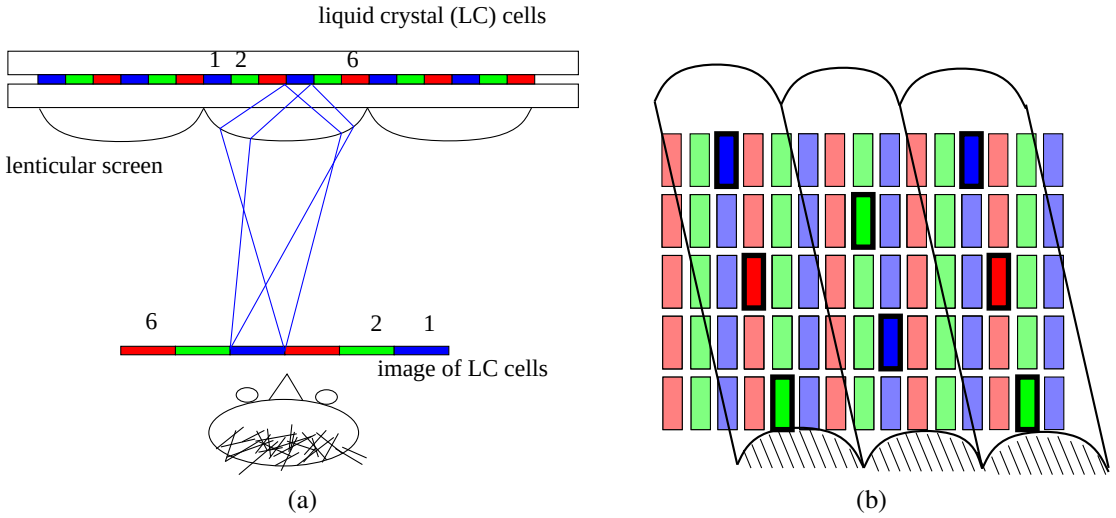


Figure 1. (a) The principle of the lenticular screen. (b) Visible subpixels for one viewing direction.

To be more specific, the *2.5D video format* is an extension of the conventional video format. It specifies for each sample point in the input images not only information about the color, but depth related data as well. This format is suited as input for rendering on 3D screens. Depth related data may be either *depth* or *disparity*, where depth may be loosely defined as 'distance (of an object) to a camera or a screen' and disparity as 'amount of shift (of an object) on the screen'. Depth and disparity are strongly related as will be detailed in Section 2.

In this paper we provide a theoretical framework for the parallactic transformations which relates screen and image disparities to captured and observed depths. Moreover we present an efficient real time rendering algorithm that uses forward mapping to reduce aliasing artefacts and that deals properly with occlusions. For improved perceived resolution, we propose to take the relative position of the color subpixels and the optic properties of the lenses into account.

This paper is organized as follows. In Section 2, we introduce the framework that relates depth values to disparity values and derive the parallactic disparity mapping formula that gives the relation between depth values and screen coordinates of transformed pixels. Moreover we present a simple algorithm to deal properly with occlusions. Next, in Section 3, we show how to integrate the parallactic transformations with sophisticated video filters in order to get high quality, i.e. free from aliasing artefacts, video output. Next, in Section 4, we give the implementation details and specifications of an FPGA realisation of the overall rendering algorithm. Finally, in Section 5, we provide some concluding remarks.

2. PARALLACTIC TRANSFORMATION

In this section, we study the parallactic transformation. We first introduce a framework for parallel camera set-ups. Next, we show how to use this framework to perform the required parallactic transformation for rendering on multiview displays.

2.1. Disparity from a parallel camera pair

We adopt the notation and terminology of Faugeras.[5] Figure 2 shows a simple geometrical model for a two camera system. Both cameras are aligned so that their optical axes are parallel, their retinas are in the same plane and their optical centers C_0, C_1 are distance b_1 apart. The optical center C_0 is in the origin of the coordinate system. The focal length of the cameras is f, whereas the world coordinates of point P are as follows: its x-coordinate is p_x, its z-coordinate is p_z. Note that p_z is the distance between P and the plane parallel to the screens and through the optical centers of the cameras. Slightly abusing notation, we denote the camera associated with optical center C_0, by C_0.

From observing similar triangles in Figure 2 the following relation is derived for camera C_0:

$$\frac{v_0}{f} = \frac{-p_x}{p_z}$$

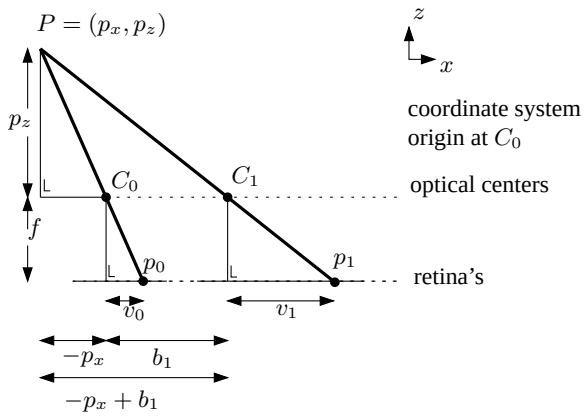


Figure 2. Perspective projection of point P onto the retinas of two parallel cameras.

and for the right camera we have:

$$\frac{v_1}{f} = \frac{-p_x + b_1}{p_z}$$

Subtracting the above equations from each other, we obtain:

$$\frac{v_1 - v_0}{f} = \frac{b_1}{p_z}$$

Let us now define *disparity* d_1 of point P at camera C_1, with respect to camera C_0 as:

$$d_1 = v_1 - v_0$$

Then, we find the following relation:

$$d_1 = \frac{f b_1}{p_z}$$

It follows that the disparity is inversely proportional to depth p_z and proportional to focal distance f and to the distance between the cameras b_1.

2.2. Disparities for multiple parallel cameras

Now that we know how a point P is mapped to the retinas of two parallel cameras, we can derive how to map the same point P onto the retina of a third parallel camera with optical center C_2 (see Figure 3). Let us assume that the third camera is a distance b_2 from the original camera. The associated disparity d_2 with the projection of P onto the retina of the third camera is:

$$\begin{aligned} d_2 &= v_2 - v_0 \\ &= \frac{f b_2}{p_z} \end{aligned}$$

Clearly, there is a constant c_2 (i.e., independent of the point P), such that $b_2 = c_2 \cdot b_1$. Subsequently, we can derive for d_2 the following:

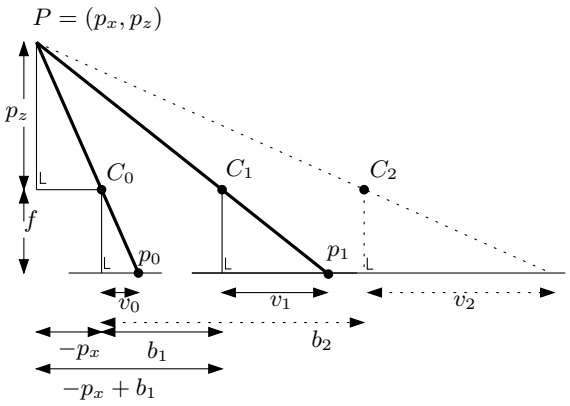


Figure 3. Perspective projection of point P onto the retinas of three parallel cameras.

$$\begin{aligned} d_2 &= \frac{f c_2 b_1}{p_z} \\ &= c_2 d_1 \end{aligned}$$

Moreover, from d_2, we can compute v_2, the coordinate in the retina associated with C_2.

$$\begin{aligned} v_2 &= d_2 + v_0 \\ &= c_2 d_1 + v_0 \end{aligned}$$

So, if we store the disparities d_1 with the pixels p_0 in the image associated with C_0, we can compute the image for any parallel camera, given the constant c_2 with respect to the baseline used to compute the disparities. The computation of desired pixel locations for the desired viewpoint requires an addition and a multiplication.

2.3. Disparities for 3D rendering

Multiview screens comprise the capability to display of distinct images for the two eyes of the viewer. We recall from the introduction, that this offers the ability of rendering a stereoscopic depth cue. If the screen is capable of displaying more than two images, the user can move his head and experience horizontal parallax. In this section, we discuss how to perform the 3D coordinate mapping tailored to such 3D screens.

In Figure 4(a), we schematically draw what is desired. The multiview screen is depicted as a line segment in the upper part of the image. In the depicted example, the viewer "sees" a point P that is behind the screen.

In order to be able to reason in terms of previously introduced disparities, we associate the viewers eye positions, with virtual camera centers. Also, we identify the origin of the multiview screen with its intersection with the optical axis of C_0. For the viewpoint C_0, corresponding to his left eye, the point P needs to be rendered at position s_0, for C_1, his right eye, the point P needs to be rendered at position s_1. For viewpoint C_{-1} which can be seen if the user moves his head to the left, the point P needs to be rendered at s_{-1}. Note that for points P on the screen, $s_0 = s_i$, for all $i \in$ viewpoints.

Clearly, the positions of p_i, $i \in$ viewpoints, can be derived by perspectively mapping the point P onto the screen. Therefore, let us consider these positions using the parallel camera paradigm of Section 2.2. In Figure 4(b), we see a point P, that is mapped onto the retinas of three parallel cameras. The centers of projection of the cameras are at a distance αf to the 3D screen. The focal lengths of the cameras are f. We identify the origin of the virtual camera retina with its intersection with the optical axis of C_0. The mapped projections of P are denoted by p_i. The distance of p_i to center of the associated camera retina is denoted by v_i.

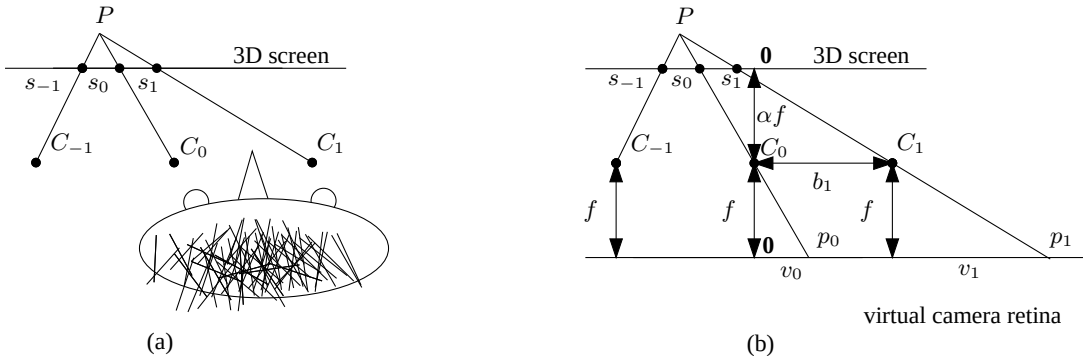


Figure 4. A viewer in front of a lenticular screen

We want to relate the positions on the retinas of the virtual cameras to positions on the 3D screen. We note that in Section 2.2, all coordinates are specified with respect to the optical axis of the camera. In Figure 4(b), three virtual cameras (C_{-1}, C_0, C_1) share a retina, but have different optical axes. In order to get the x coordinate of the screen points s_i, we need to project the points p_i onto the screen through the corresponding camera axes.

From similar triangles, we derive:

$$\frac{v_0}{f} = \frac{-s_0}{\alpha f}$$

Hence,

$$s_0 = -\alpha v_0$$

More general, it holds that

$$\frac{v_i}{f} = \frac{-s_i + b_i}{\alpha f} \text{ and}$$
$$s_i = -\alpha v_i + b_i$$

If we know the disparities for the camera pair C_0, C_1, we can express $s_1 - s_0$ using our parallel camera paradigm as follows:

$$\begin{aligned} s_1 - s_0 &= -\alpha v_1 + b_1 + \alpha v_0 \\ &= -\alpha d_1 + b_1 \end{aligned}$$

Moreover, using $b_i = c_i \cdot b_1$ and deriving $d_i = c_i \cdot d_1$ (see Section 2.2), we can express $s_i - s_0$, as

$$\begin{aligned} s_i - s_0 &= -\alpha v_i + b_i + \alpha v_0 \\ &= -\alpha d_i + b_i \\ &= c_i(-\alpha d_1 + b_1) \end{aligned}$$

Using the equation above, we can compute the screen position of point P for virtual viewpoints with an arbitrary baseline, using disparities associated with a given baseline. Let us, withous loss of generality, assume that the given baseline equals the baseline of an actual camera pair that recorded point P. Given the associated disparities, we can compute the screen position of point P for any desired virtual viewpoint. We conclude that we can use image plus camera disparities as input for our mapping to 3D screen coordinates, using the simple formula

$$s_i = s_0 + c_i(-\alpha d_1 + b_1)$$

where c_i is a gain factor which controls the baseline for the virtual cameras with respect to the baseline of the original recording camera pair. c_i also incorporates the view number (e.g., $\in -4, \ldots, 4$ for a nine view display). Let us denote the view number by k. For equidistant virtual viewpoints, it holds $c_i = kc_1$. Hence, the equation may rewritten as follows:

$$\begin{aligned} s_i &= s_0 + kc_1(-\alpha d_1 + b_1) \\ &= s_0 + kc_1\alpha(-d_1 + \frac{b_1}{\alpha}) \end{aligned}$$

We observe that α is a constant. We can simplify the formula above by taking $\gamma = \alpha \cdot c_1$, and taking $\delta = \frac{b_1}{\alpha}$. Now, the general *parallactic disparity mapping* formula is:

$$s_i = s_0 + \gamma k(-d_1 + \delta)$$

where γ, and δ, are control parameters. We can control which part of the scene will appear on the screen by adjusting δ. Points for which $d_1 = \delta$ will appear on the screen, whereas points for which $d_1 > \delta$ or $d_1 < \delta$ will be observed in front or behind the screen respectively. We can adjust the range of observed depths by adjusting γ. A higher value of γ will increase the disparities, allowing objects to appear closer to the viewer.

The parallactic disparity mapping is a very powerful function. We can use it to generate the coordinates for any desired viewpoint, i.e., for any view for a multiview display, from a reference image and disparity information using only a multiplication and addition per pixel. Even stronger, the function allows us to transmit the disparities, and control the amount of depth, and the parts of the image that appears at the display in the rendering stage at the receiver side.

In our implementation in Section 4, we use the parallactic disparity mapping formula to generate the desired views for our multiview display. In our system, γ and δ are run-time control parameters.

2.4. Detecting occlusions

The coordinate mapping for 3D screens of Section 2.3 have the special properties that multiple input points can map to the same output coordinate. This raises the need for special processing to prevent blending of overlapping parts of the input image in a generated view from a desired viewpoint.

For sake of discussion, in the following, we consider the input image as a set of samples of an image projected onto a terrain, i.e., a texture with relief according to the corresponding depths associated with the image information. Hence, we can reconstruct the terrain from the set of samples. Without loss of generality, we place the screen plane behind the terrain, i.e., the control parameter from Section 2.3 $\delta = 0$. We will show that problem of generating new views can be solved by projecting our reconstructed terrain onto a flat surface from various viewpoints.

First, we give an overview of related work. From literature, we find that it is possible to traverse the input image in such a way that occluding parts are visited before occluded parts. Anderson[6] reports how to find a front to back order for rendering a landscape from an arbitrary viewpoint. He starts at the epi-pole of the transformation; the epi-pole is the image in the desired view of the optical center of the original camera. The front to back order is accomplished by starting at the epi-pole of the camera-transformation and by moving away from the epi-pole. Anderson maintains a representation of the 2D perimeter of the rendered area in order to decide whether during his traversal of the landscape, newly discovered parts should be rendered.

McMillan[7] introduces the result of Anderson in the computer graphics community. By reversing the order of rendering from back to front, he does no longer need to maintain the rendering perimeter, but sacrifices possible integration with higher order video filtering.

As put forward in the introduction, the desired transformation for our 3D display only encompasses horizontal disparity. Next, we shall see how, given the aforementioned property, the perimeter of Anderson can be composed of a single value per scan line. Subsequently, we will show how we can efficiently integrate the transformation with hardware video filters.

In the following few paragraphs, let us consider a single scan line of the input image, i.e., the '1.5D' problem of projecting a one dimensional piecewise linear terrain onto an image line from various viewpoints (see Figure 5 for a cross

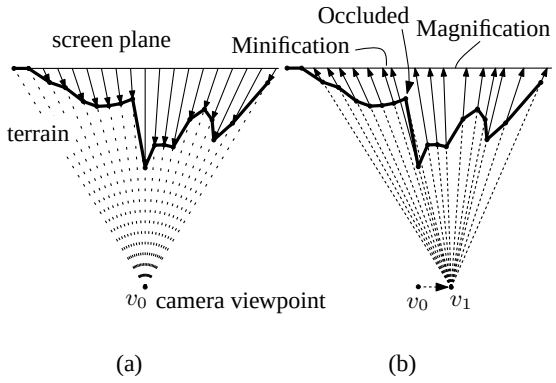


Figure 5. (a) Reconstruction of a cross section of the sampled terrain representing image plus depth (b) Original image samples remapped for a different view point.

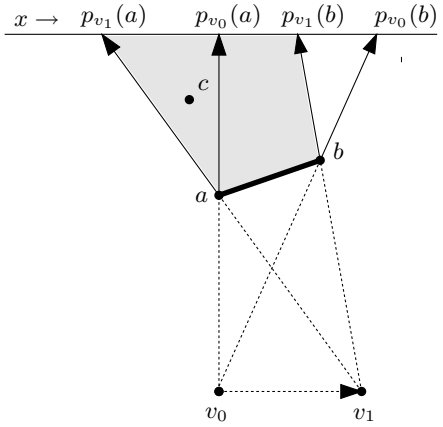


Figure 6. Illustration of Lemma 1.

section of the terrain with the y, z-plane). We shall show that occlusions can be determined on the fly during a traversal of this scan line.

We denote the projection of an input sample point a for viewpoint v onto the image line by $p_v(a)$. We identify the original camera viewpoint with v_0. The following lemma gives us a relation between scan line order and occlusions for other viewpoints.[8] Figure 6 illustrates the lemma.

LEMMA 1 ([8]). *Let a, b be subsequent samples on an input scan line of a depth terrain for original camera position v_0, such that $p_{v_0}(a) < p_{v_0}(b)$. Let $v_1 > v_0$ be the desired camera viewpoint. Let c be a sample point from the original image that is occluded by line segment (a, b) from viewpoint v_1. Then $p_{v_0}(c) < p_{v_0}(a)$.*

Proof: In order for line segment (a, b) to be visible from viewpoint v_1, v_1 needs to be on the same side of the line supported by (a, b) as v_0. Consequently, $p_{v_1}(a) < p_{v_1}(b)$. From the occlusion of c, it follows that $p_{v_1}(a) < p_{v_1}(c) < p_{v_1}(b)$.

By construction, c is visible from viewpoint v_0, so either $p_{v_0}(c) < p_{v_0}(a)$, or $p_{v_0}(c) > p_{v_0}(b)$. Since $v_1 > v_0$, $p_{v_1}(b) < p_{v_0}(b)$, which implies that $p_{v_0}(c) > p_{v_0}(b)$ cannot hold. We conclude that $p_{v_0}(c) < p_{v_0}(a)$. □

From Lemma 1, it follows that for a desired viewpoint $v_1 > v_0$, a traversal of the input scan line with *decreasing* x-coordinate, i.e., from right to left, will let us encounter occluding parts of the terrain before occluded parts. Therefore, we can solve occlusions as follows (see Figure 7): First, we introduce a variable extent that maintains the x-extent of the mapped pixel coordinates in the output domain. Then, we can conclude that if a pixel that is processed does not lengthen

the extent, it must be occluded by the previously processed pixels. For a viewpoint transformations $v_1 < v_0$, the argument is analogous: in that case we traverse the scan line with *increasing* x-coordinate, i.e., from left to right.

Now that we can detect occlusions, the road is open for an input driven rendering approach.

3. RESAMPLING FRAMEWORK

The depth information within the 2.5D video stream allows us to model the original image as a set of samples of an image projected onto a terrain. In Figure 5(a) we show a cross section of the terrain reconstucted from the depth values: the lengths of the arrows are related to the depth values of the samples.

The 2.5D video format represents a subset of the full 3D model of the world. Rendering from other viewpoints can be accomplished by projecting the terrain onto the image plane from the desired viewpoint. In Figure 5(b) we show that after viewpoint transformation, the density of the projected input pixels is not uniform in the output domain. Hence, a resampling procedure is required. In general, a resampling procedure can be seen as a four step procedure.[9]

1. Reconstruct a continuous signal from the sampled data.
2. Deform the continuous signal as desired.
3. Band limit the deformed signal.
4. Sample the band limited signal.

Let us first set out the resampling framework more formally. We formalize the four step resampling procedure for a one dimensionally signal.

We adopt the notation from Heckbert.[9] Let $f(\mathbf{u})$ ($\mathbf{u} \in \mathbb{N}$) denote the input signal, $\mathbf{m}(\mathbf{u})$ denote a bijective mapping function that maps input coordinates $\mathbf{u}$ onto output coordinates $\mathbf{x}$, $r(\mathbf{u})$ denote a reconstruction filter and $h(\mathbf{x})$ ($\mathbf{x} \in \mathbb{R}$) denote a prefilter. Then, the general resampling framework can be formulated as follows.

1. The reconstructed signal from $f(\mathbf{u})$ is

$$f_c(\mathbf{u}) = f(\mathbf{u}) \otimes r(\mathbf{u}) = \sum_{\mathbf{k} \in \mathbb{N}} f(\mathbf{k}) r(\mathbf{u} - \mathbf{k})$$

   where $\otimes$ denotes convolution.

2. The deformed input signal is

$$g_c(\mathbf{x}) = f_c(\mathbf{m}^{-1}(\mathbf{x}))$$

3. The band limited deformed input signal is

$$g'_c(\mathbf{x}) = g_c(\mathbf{x}) \otimes h(\mathbf{x}) = \int_{\mathbb{R}} g_c(\mathbf{t}) h(\mathbf{x} - \mathbf{t}) d\mathbf{t}$$

4. The discrete output signal is

$$g(\mathbf{x}) = g'_c(\mathbf{x}) i(\mathbf{x})$$

   where i is an impulse train.

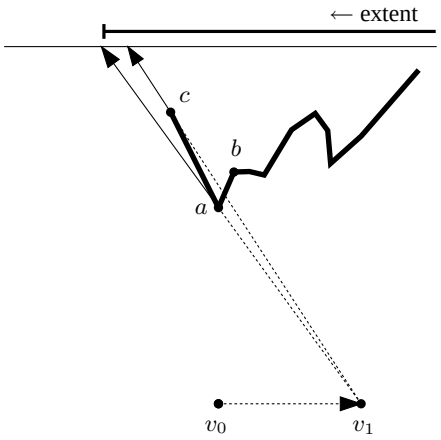


Figure 7. Detecting occlusions by maintaining the x-extent in the output domain. The camera translation imposes a right to left traversal. Sample c is occluded, since it does not lengthen the extent.

3.1. Forward Mapping

An explicit expression for $g'_c(\mathbf{x})$ is derived by expanding the above steps in reverse order:

$$\begin{aligned} g'_c(\mathbf{x}) &= \int_{\mathbb{R}} h(\mathbf{x}-\mathbf{t}) \sum_{\mathbf{k}\in\mathbb{N}} f(\mathbf{k}) r(\mathbf{m}^{-1}(\mathbf{t})-\mathbf{k}) d\mathbf{t} \\ &= \sum_{\mathbf{k}\in\mathbb{N}} f(\mathbf{k}) \rho_{\mathbf{k}}(\mathbf{x}) \end{aligned}$$

where

$$\rho_{\mathbf{k}}(\mathbf{x}) = \int_{\mathbb{R}} h(\mathbf{x}-\mathbf{t}) r(\mathbf{m}^{-1}(\mathbf{t})-\mathbf{k}) d\mathbf{t}$$

The warped and filtered basis function $\rho_{\mathbf{k}}(\mathbf{x})$ is defined as a screen space integral, and is constructed by first warp and filter the reconstruction filter footprint to construct the resampling kernels $\rho_{\mathbf{k}}$ and then sum up the contributions of these kernels in screen space. This way, the inverse of the mapping function is not used explicitly. Therefore, this input driven resampling is known as *forward mapping*.[10]

We have now formalized the resampling framework. In the following section, we shall detail how to obtain a bijective mapping function, and subsequently apply forward mapping to implement the resampling.

3.2. Integrating mapping and filtering

In this subsection, we show how to put the parallactic disparity mapping into a resampling framework. We build upon the previous subsections.

We recall the four step procedure of Section 3. The first step is carried out by a reconstruction filter, the third step is carried out by a prefilter. The transformation used in Step 2 should reflect the parallactic disparity mapping, as explained in Section 2.3.

From Section 3, it follows that $\mathbf{m}$ needs to be invertible. Clearly, the self-occlusions of the input induced by horizontal camera translation cause the view point transformation to dissatisfy this constraint. Fortunately, Lemma 1 allows us to find occlusions during the traversal of the scan line. If we maintain the extent of the mapped input samples in the output domain, and discard occluded samples, we derive a mapping $\mathbf{m}$ that is invertible.

Besides the induced occlusions, there are also areas of possible magnification of the input signal, as well as areas of possible minification. These minifications, magnifications and occlusions follow from the fact that objects in the image have different distances to the camera. Therefore, we can try to interpret what gives rise to each of these cases. Let us

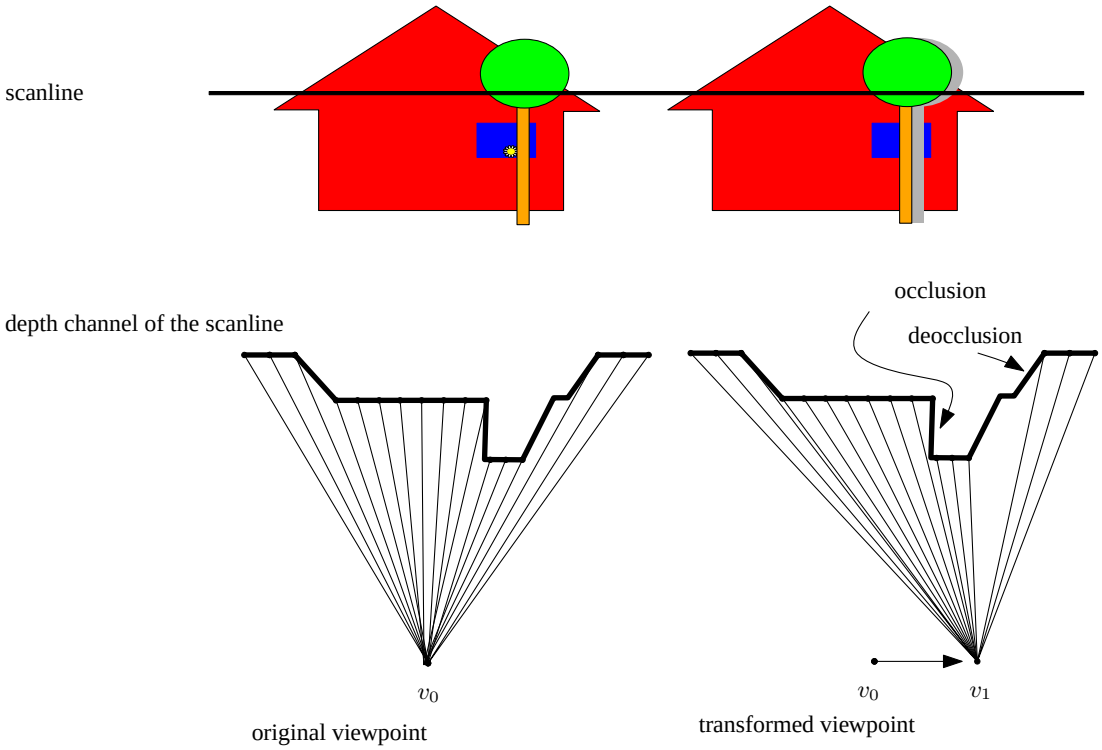


Figure 8. Rendering from image+depth: occlusions and disocclusions.

consider the example depicted in Figure 8: a tree in front of a house. We have a base image and a per-pixel depth channel. To render an image from a new point, image information can be taken from the base image. Actually, there are parts of the new image to which both the foreground (tree), and the background (house) are mapped. Clearly, the rendering stage has to deal with the introduced occlusion. Moreover, there is a part of the house in the new viewpoint that was not recorded by the original camera. In other words, when we consider 2.5D video, we lack information about that part of the house in the image, and we have to reconstruct the signal in the disoccluded area. Good results for have been achieved by extrapolating the background color. An other option is to transmit extra information about the background, and insert the reconstructed extra information at the position of the disocclusions. This so called hidden layer extension is beyond the scope of this paper. Here we consider single frame rendering, only.

Now that we have an invertible mapping, we can apply an input driven splatting method. When splatting, we need to select the appropriate filter function for prefiltering, suitable for the grid of the various views of muliview displays (recall Figure **??**).

In the literature, we found the work of Konrad and Agniel who have addressed filtering for screens similar to the Philips 3D lenticular screens.[11, 12] Konrad and Agniel start with a set of pre rendered images and focus on the display specific filtering to avoid aliasing. Konrad and Agniel derive a two-dimensional filter that optimally prefilters for a multiview panel, and they discuss an orthogonal approximation of the ideal prefilter.[11] Their observation is that the added value of a non-separable 2D filter over a separable 2D filter is negligible. In our implemenation, we have chosen to use a separable filter. In the next section we will detail how we implemented the required filter passes.

4. INPUT DRIVEN IMPLEMENTATION

In the following, we discuss the input driven algorithm to perform rendering. An input driven algorithm computes for each input pixel which output pixels are affected by the input pixel and updates those output pixels accordingly. Such an input driven algorithm could be executed in a streaming architecture that only demands limited line memories. In the following, we detail the various components of the implementation.

Let us first describe possible prefilters. We recall that Konrad and Agniel studied prefiltering for 3D lenticular streams. Their observation was that an orthogonal filter performs as well as a prefilter for such screens. Therefore, to accommodate a streaming algorithm, we prefer a separable orthogonal filter. We can use such a filter in an integrated two-pass algorithm. When necessary, vertical filtering and scaling is performed before horizontal mapping and filtering.

The vertical pass is a straightforward scaling pass. The output of the vertical pass is fed to the horizontal pass. The horizontal pass only requires a single input line at the time. Hence, the vertical scaler can run in a streaming mode, using line memories, and does not need to compute an entire intermediate image.

For the horizontal processing, we apply the techniques presented in Sections 2 and 3. Depending on the desired viewpoint, we either scan the input line from left to right, or from right to left. During the scan, we use the parallactic disparity mapping to map the input pixels to their desired output locations and solve the occlusions by maintaining the extent of the mapping. As we have learned, the resulting mapped input pixels are non uniformly distributed in the output grid. Therefore, we prefer a filter that can handle variable scaling factors. A suitable filter is the step response scaler of Koen Meinds as described in.[13] This scaler, which uses a polyphase FIR filter, is an implementation of input driven, forward mapping.

In the preceding sections, we have discussed how to map and filter a one-channel signal. The 3D autostereoscopic display, however, has red, green and blue subpixels. Due to the slanted lenticulars, the pixels that are visible for a single view are arranged in a slanted pattern. The resolution per view is lower than the original panel resolution. In our implementation, we choose to use the slanted sampling grid of the lenticular display as output domain in the integrated mapping and filtering step. We compute only the necessary subpixels per view.

For improved perceived resolution, we take the relative position of the color subpixels into account. This is called *subpixel sampling*.[14] For each color component of a single view, we determine the impulse train i in Step 4 of the resampling framework (recall Section 3). The advantage of this approach is that we can display higher frequencies (resolutions) than the Nyquist frequency induced by a single view of the panel. Such, at the cost of color aliasing. To further refine the position of the impulse train, we have measured the spatial location of light emitted through the lenticular foil from these subpixels. The resulting refined sample point positions result in superior image quality compared to straightforward rectangular subsampling.

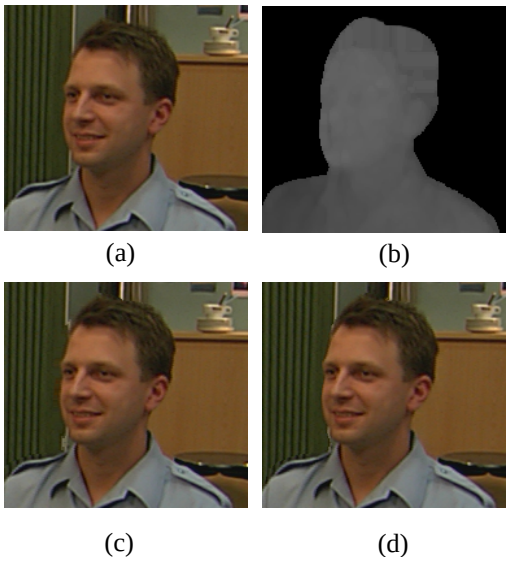
5. SUMMARY AND DISCUSSION

In this paper, we have presented an efficient real time rendering algorithm for multiview displays. The input format under consideration was image plus depth related information.

We have shown how to combine hidden surface removal and high quality filtering techniques into an efficient system suitable for driving high definition panels. Various alternative approaches to solving the rendering problem exist. There is a clear trade off between image quality and computation speed. However, what quality is needed depends upon the application (cf.Hollywood movies on a nine-view HD-TV screen versus simple 3D games on a five-view display on a mobile phone). There are many parameters that affect image quality and compute requirements, and there are trade-offs between image quality and compute requirements. The proof of the pudding is in the eating; therefore, render algorithms should be selected only after extensive testing on the application they are aimed for.

In Figure 9, we show the results of our algorithm computed for a rectangular grid suited for printing on paper. In (a) we show a detail of a test sequence. This detail comprises 188 x 188 pixels; an associated disparity map is shown in (b). A left view and a right view as generated by our algorithm is displayed in (c) and (d) respectively. The maximum left and maximum right shift applied to an input pixel in the example is 2.2 pixels wide. Thanks to the advanced filter employed, no blur has been introduced; note the crisp and bright nature of the generated images.

As it is impossible to convey the impression of 3D video on paper or on a 2D screen, the quality of our 3D images can only be demonstrated on real displays. Nevertheless, it is important to mention that our techniques alleviate many annoying aliasing artefacts, such as these typically induced by the slanted lenticular foil. Our techniques accomplish this without introducing unnecessary blur. Consequently, our processing delivers crisp and bright images which are a pleasure to watch.



(a) (b)

(c) (d)

Figure 9. Test results: (a) original view, (b) disparity map, (c) left-hand view, (d) right-hand view

Acknowledgements

We thank our 3D rendering colleagues Marc op de Beeck, Bart Barenbrug, Bram Riemens, Boris Djapic, and Ben Zwaans for their valuable remarks and discussions. This work was partially supported by the European Commission (EC) through their Information Society Technologies (IST) program under proposal No. IST-1-507926 - OLGA.

REFERENCES

1. "Philips lenticular autostereoscopic 3D display." `http://www.business-sites.philips.com/3dsolutions/technology/index.html`.
2. "Sharp two view barrier display." `http://www.sharp3d.com/`.
3. " Opticality multiview barrier display." `http://www.opticalitycorporation.com/technology/technology_core.html`.
4. C. Fehn, "Depth-image-based rendering (DIBR) compression and transmission for a new approach on 3D-TV ," in *Proc. of SPIE Stereoscopic Displays and Applications*, 2004.
5. O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*, MIT Press, 1994.
6. D. Anderson, "Hidden line elimination in projected grid surfaces," *ACM Transactions on Graphics* , pp. 274–288, 1982.
7. L. McMillan, *An Image-Based Approach to Three Dimensional Computer Graphics*. PhD thesis, UNC Computer Science, TR97-013, 1997.
8. R.-P. Berretty and F. Ernst, "High quality images from 2.5D video," in *Short Presentations, Eurographics*, pp. 255–262, 2003.
9. P. Heckbert, *Fundamentals of Texture Mapping and Image Warping*. PhD thesis, Department of EECS, U.C. Berkeley, 1989.
10. D. Ghazanfarpour and B. Peroche, "A high-quality filtering using forward texture mapping.," *Comp. & Graphics* **15**(4), pp. 569–577, 1991.
11. J. Konrad and P. Agniel, "Artifact reduction in lenticular multiscopic 3-d displays by means of anti-alias filtering," in *SPIE*, 2003.
12. J. Konrad and P. Agniel, "Non-orthogonal sub-sampling and anti-alias filtering for multiscopic 3-d displays," in *SPIE*, 2004.
13. K. Meinds and B. Barenbrug, "Resample hardware for 3D graphics," in *Proceedings of Graphics Hardware 2002*, 2002.
14. M. Klompenhouwer and G. de Haan, "Subpixel image scaling for color-matrix displays," *Journal of the Society for Information Display* **11**(1), pp. 99–108, 2003.