High quality images from 2.5D video

R-P. Berretty and F. Ernst

Philips Research, Eindhoven, The Netherlands

Abstract

Given a 2D video stream with an accompanying depth channel, we render high quality images from viewpoints close to the original one. This is for instance required to generate a 3D impression on stereoscopic or multiview screens. We propose a technique for video based rendering that supports higher order video filtering. We focus on screens that support horizontal parallax.

We can optionally incorporate rendering of a so called hidden layer *that contains data of parts of the scene that are hidden from the original viewpoint. We are able to render high quality images at only the added cost of the video filtering.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Display algorithms

1. Introduction

Stereovision is one of the strongest cues for a human to percieve the 3D nature of the world. The left eye receives a different image from the right eye, because foreground objects are in a different position relative to background objects for distinct viewing positions. The relative position of the same object in the two projected images is called the disparity. In order to minimize eyestrain, a stereo-pair of images must only have horizontal, and no vertical disparity.

A video format that is suited for rendering from different viewpoints is 2D video format enriched with a depth channel, i.e., for each point in the scene, we know the distance of the point to the camera. We call the resulting stream 2.5D video. This format is currently advocated to transmit 3D video [14].

A 2.5D video stream can be obtained by a camera that also records depth information, or by depth estimation from a 2D recorded video sequence[4, 7, 12, 15].

From 2.5D video, we want to render video streams that appear to be seen from another position close to the original camera. We focus on new views for screens that support horizontal parallax. The challenge is to efficiently render these views without introduction of unnecessary artefacts.

1.1. Previous Work

One way to solve the problem is to model the 2.5D image as a fine (pixel size) wire frame with the image as a texture, and subsequently rendering the image with a standard PC graphics pipeline. This solution, however, has two drawbacks. Firstly, there is a lot of overhead involved in the set-up of the triangles in the pipeline. Secondly, this approach suffers from the poor quality of the filters that are present in such pipelines. This is because PC graphics cards are optimized for performance and not for image quality. We found that standard computer graphics approaches introduce annoying aliasing artefacts in the resulting output video stream.

In the literature, there are solutions that partially address our problem. Contributions come from the area of image based rendering. The paper of Oliveira *et al.*[11] deals with photo realistic rendering based on images that have depth information; Popescu *et al.* [13] propose a hardware architecture to render from images with depth maps. However, these techniques do not, or only marginally support the integration of high order video filters.

An approach that does offer high quality pre-filtering is forward texture mapping, or texel splatting [10, 17]. We shall explain this approach in Section 3. The drawback of splatting, however, is the way occlusions are solved. The standard solution in forward texture mapping architectures needs a fragment buffer that stores contributions for each output pixel, together with the depth information. The buffer accumulates the contributions of the entire scene, and sorts them from front to back. After processing the whole scene, the buffer can be used to render the scene front to back.

1.2. Our Approach

In this paper, we describe how to integrate higher order video filters into the rendering stage for 2.5D video. These filters can be chosen to alleviate the aforementioned aliasing artefacts.

We will use texel splatting, but avoid the use of a fragment buffer, thereby enabling usage of higher order video filters at low cost. In our special case we only have to address horizontal camera translations (the transformation that supports the horizontal parallax for multiview displays). From the camera transformation under consideration, it follows that the deformation of the input image is constrained to be horizontal. This allows us to process in scanline order.

At depth discontinuities, the use of data which is not available in the original image may be advantageous. While most current processing techniques use slight distortions in an attempt to reconstruct information at depth discontinuities, we have adopted a representation that provides extra data, together with an efficient way to store this data. Hence, to reduce blur or other distortion artefacts, we use an extension to the 2.5D video stream that can be used in the rendering stage of the display. This extension consists of what is called a *hidden layer*. In this paper, we also show how to splat the hidden layer to the output video screen. Our rendering routine, however, does not depend on the availability of a hidden layer and has fallback options that will never leave holes in the resulting images.

The remainder of this document is organized as follows. In Section 2 we describe the 2.5D video format and discuss the difficulties that have to be dealt with in the rendering stage. In Section 3 we show how to integrate video filtering into the rendering stage. In Section 4 we elaborate on the hidden layer extension of the 2.5D stream that can be used to eliminate blur artefacts. In Section 6 we summarize and draw some conclusions. In Appendix 5, we give example output frames.

2. Preliminaries

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 1(a) we show a cross section of the sampled terrain. The lengths of the arrows indicate 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 1(b) we show that after viewpoint transformation, the density of the projected input

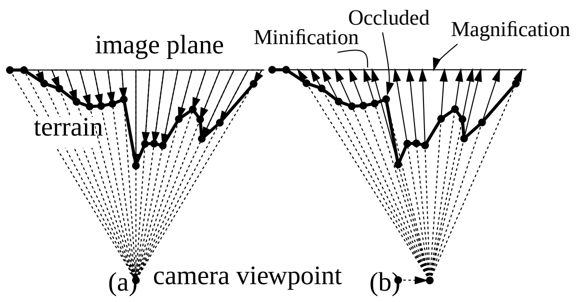

Figure 1: *(a) Reconstruction of a cross section of the sampled terrain representing image plus depth (b) Original image samples remapped for a different view point.*

texels 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 [6].

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.

The first and third step are filtering steps. The first step is carried out by a reconstruction filter, the third step is carried out by a pre-filter. From Figure 1(b) we can see that the view point transformation imposes special requirements onto Step 2 (the deformation step) that are not present in a conventional resampler for 2D video: viewpoint transformation can induce occlusions of parts of the original image.

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 consider the example depicted in Figure 2: 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 fall back to advanced reconstruction (magnification) filters that (try to) reconstruct the signal in the deoccluded area. An other option is to transmit extra information about the background, and insert the reconstructed extra information at the position of the deocclusions. This so called hidden layer extension will be treated in Section 4.

In the following section, we show how to render from a

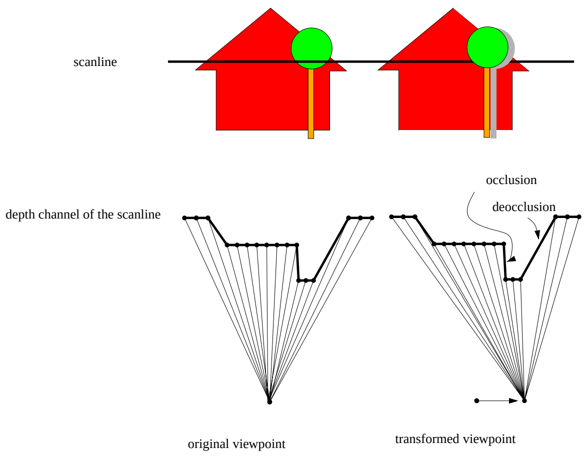

Figure 2: *Rendering from image+depth: occlusions and deocclusions.*

transformed viewpoint using only the 2.5D input channels. We also show how to find occlusions efficiently.

3. Rendering 2.5D video

In this section, we show how to properly render a 2.5D video stream. We treat the problem in the context of the resampling framework. 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 [6]. Let $f(\mathbf{u})$ $(\mathbf{u} \in \mathbb{N})$ denote the input signal, $\mathbf{m}(\mathbf{u})$ denote a 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.

Our implementation of the resampling procedure, which will be detailed in Section 3.2, is a splatting approach [10, 17], in which an explicit expression for $g'_c(\mathbf{x})$ is derived by expanding the above steps in reverse order:

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

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 is called splatting.

From the definition of $\rho_{\mathbf{k}}(\mathbf{x})$, 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.

In the introduction, we mentioned that a common solution to deal with occlusions is to collect contributions for each output-pixel in a so called fragment buffer. Such a fragment buffer, however, leads to an inefficient algorithm in terms of bandwidth and memory. Therefore, we shall show that in the case of the view point transformation, we can find a mapping $\mathbf{m}$ that is invertible, i.e., in the following subsection we show how occlusions can be found during the processing of a scanline.

3.1. Occlusion Handling

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 [1] 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 [9] 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 scanline. Subsequently, we will show how

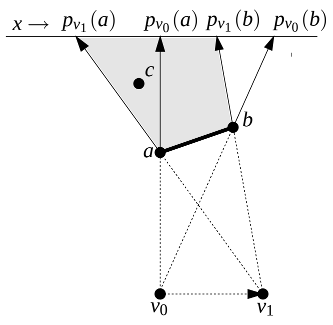

Figure 3: *Illustration of Lemma 1.*

we can efficiently integrate the transformation with hardware video filters.

In the following few paragraphs, let us consider a single scanline 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. We shall show that occlusions can be determined on the fly during a traversal of this scanline.

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 scanline order and occlusions for other viewpoints. Figure 3 illustrates the lemma.

Lemma 1 Let a, b be subsequent samples on an input scanline 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 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 scanline with *decreasing* x-coordinate, will let us encounter occluding parts of the terrain before occluded parts. Therefore, we can solve occlusions as follows (see Figure 4): First, we introduce a variable extent that maintains the x-extent of the projected texels in the output domain. Then, we can conclude that if a texel that is processed does not lengthen the extent, it must be occluded by the previously processed texels. For a viewpoint transformations $v_1 < v_0$, the argument is analogous: in that case we traverse the scanline with *increasing* x-coordinate.

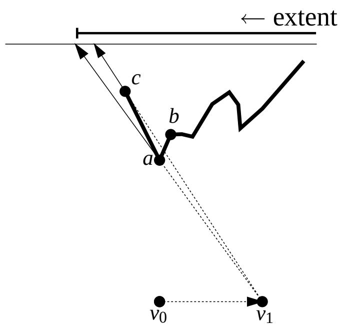

Figure 4: *Detecting occlusions by maintaining the x-extent in the output domain. The camera translation imposes a right to left traversal. Sample c is occluded, since it does not lengthen the extent.*

Now that we can detect occlusions, the road is open for a texel splatting procedure. In Section 3.2 we discuss the implementation of this procedure.

3.2. Filter Implementation

Now that we know how to detect occlusions in the output view. We shall show how to implement the solution in the resampling framework that was given in the beginning of this section.

Let us discretize the general resample procedure. A common 1D discretization of the resampling process is to use a zero-order reconstruction filter (Dirac function).

A drawback of the Dirac reconstruction filter is the suffering from an artefact called DC-ripple, or frequency ripple. DC ripple is visible as intensity fluctuations on the output signal for constant intensity input signal[5, 8]. Since we deal with small, varying minification and magnification, our application is especially susceptible to DC ripple, so we prefer not to use Dirac reconstruction.

DC ripple can be avoided by using a first order (box) reconstruction filter as shown by Meinds and Barenbrug [10]. Their resampling algorithm, that uses box reconstruction, is developed below. Let box be the piecewise constant box filter:

$$\begin{aligned} \mathrm{box}(\mathbf{x}) &= 1 : -\frac{1}{2} < \mathbf{x} < \frac{1}{2} \\ \mathrm{box}(\mathbf{x}) &= 0 : \text{otherwise} \end{aligned}$$

Then,

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

Substituting $\mathbf{t} = \mathbf{m}(\mathbf{u})$:

$$\rho_{\mathbf{k}}(\mathbf{x}) = \int_{\mathbb{R}} h(\mathbf{x}-\mathbf{m}(\mathbf{u}))\,\mathrm{box}(\mathbf{u}-\mathbf{k})d\mathbf{u}$$

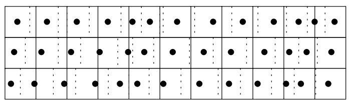

Figure 5: *Three scanlines of the box reconstructed signal projected onto the output domain. The solid lines are the output pixels. The dots are the projected texel coordinates. The dashed lines are the midpoints of the projected texels.*

We simplify again, $\mathrm{box}(\mathbf{u}-\mathbf{k}) = 1$ for $\mathbf{u} \in [\mathbf{k}-\frac{1}{2}, \mathbf{k}+\frac{1}{2}]$, so

$$\rho_{\mathbf{k}}(\mathbf{x}) = \int_{\mathbf{k}-\frac{1}{2}}^{\mathbf{k}+\frac{1}{2}} h(\mathbf{x}-\mathbf{m}(\mathbf{u}))\,d\mathbf{u}$$
$$= H(\mathbf{x}-\mathbf{m}(\mathbf{k}+\frac{1}{2})) - H(\mathbf{x}-\mathbf{m}(\mathbf{k}-\frac{1}{2}))$$

where H is the definite integral of h.

In our implemention, we compute the size of the boxes in the output domain. We take the midpoints of two successive warped input samples as the warped midpoint of intermediate values. Still, the density is dependent on the density of the projected texels onto the image line. In the current implementation, we have chosen to simply drop input texels that appear to be occluded during the terrain traversal; it is possible, however, to refine the reconstruction and to use, e.g., a contribution proportional to the non-occluded part of the first order reconstructed input sample of the image. Figure 5 shows the result after box reconstruction and occlusion handling. The size of the boxes is a measure for the contribution of the original input sample.

For the implementation of pre-filter, we use a splatting approach, which is input driven. We have implemented various pre-filter FIR (finite input response) functions with a varying number of taps. For each input sample, we have to implicitly compute $\rho_{\mathbf{k}}(\mathbf{x})$. Because we use a FIR filter, $\rho_{\mathbf{k}}(\mathbf{x})$ is defined only for a finite number of output samples. We define the range of output samples that receive contributions as $[\mathbf{f}_{\mathbf{k}}(\mathbf{x}), \mathbf{l}_{\mathbf{k}}(\mathbf{x})]$.

From the discussion in Section 3.1 it follows that $\mathbf{m}(\mathbf{u})$ is monotonous. Hence, $\mathbf{f}_{\mathbf{k}}(\mathbf{x})$, and $\mathbf{l}_{\mathbf{k}}(\mathbf{x})$ are monotonous as well. Hence, the contributions of input samples can be accumulated using a sliding window of output samples that traverses the output scanline as we traverse the input scanline. This method is known as a transposed mode FIR filter, and can easily be implemented in hardware.

We have implemented the above filter structure in software. We have started to only splat to pixels onto a single scanline. This eliminates aliasing at the boundary of foreground objects, and does not introduce artefacts in the interiors of object. Note that we could also perform vertical filtering to improve picture quality even more by eliminating vertical aliasing which may be caused by shear [16]. Our experiments indicate that plain vertical splatting gives satisfactory results. Figure 6 shows the output pixels that receive a contribution from one input texel for a pre-filter footprint that also has vertical extent. The latter approach can also be implemented in hardware, but requires a number of line memories equal to the height of the pre-filter footprint.

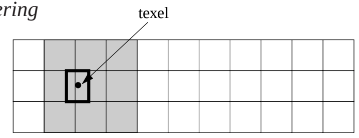

Figure 6: *The pixels in the shaded area of the output domain receive a contribution from the depicted input texel that is projected onto the output domain.*

Note that the implementation of the filter structure allows us to detect areas of magnification in the image. This is when two consecutive midpoints of warped input samples are more distant than one unit in the output grid. We can, in this situation put more effort in the reconstruction. In the next section we discuss how to insert samples from another input stream at deocclusions. Our implementation has a fallback option of feeding 'artificial samples' that are reconstructed using a bilinear interpolating filter to the higher order pre-filter resampling structure. We could, however, also fall back to higher order reconstruction filters.

In Appendix 5, we show example output of our renderer. Figure 8 shows a video frame after viewpoint transformation that was generated without the use of a higher order video filter. Figure 9 shows a frame that was generated with a four-tap horizontal FIR filter.

In the next section, we show how we incorporate the rendering of the hidden layer into the framework.

4. Rendering layered video

In this section, we discuss the hidden layer that is used to improve rendering of deoccluded areas in generated views. In Section 4.1, we elaborate on the definition and give a short note on our generation of the hidden layer. In Section 4.2 we discuss how the hidden layer can be incorporated in the rendering framework of the paper.

4.1. Hidden layers

The idea of the hidden layer that we use was introduced by Chang and Zakhor. They present the "Multivalued Representation" (MVR), which describes a scene in terms of multiple levels, based on a certain reference frame [2]. The base layer is the visible part in the reference frame, and consists of 2.5D video that we already discussed in the previous part of this paper; (hidden) layer k consists of those pixels which are occluded $k-1$ times for the reference image. The reader must note that the hidden layers are specified in the coordinate frame of the base image. Therefore, the features of the hidden layer are hidden by foreground objects of the base layer. Compared to a conventional layered approach, it is claimed that with this approach, 3 levels are usually sufficient. The

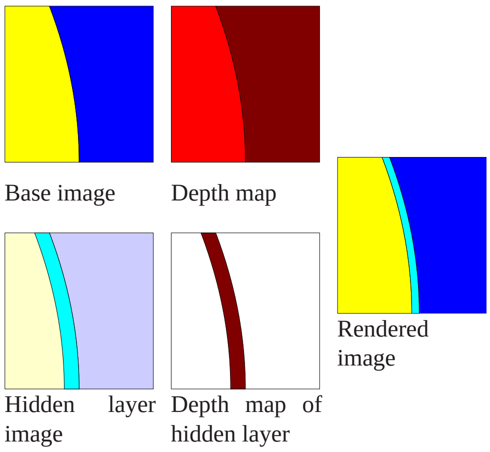

Figure 7: *Rendering from image, depth and a hidden layer. Darker colors are further away from the camera. At the deocclusion, the rendered image is filled with information from an additional 'hidden' layer, consisting of those parts of the scene which are visible in the rendered image but not in the original image.*

layers are constructed from independently computed three dimensional models corresponding to the multiple views.

The concept of the hidden layer can be explained by means of an example. In Figure 7, we zoom in on a potential deocclusion. In the upper half of the figure, we see the base image with the corresponding depth information, i.e. a 2.5D video frame. Darker colors are further away from the camera. In the bottom half of the figure, we see the hidden layer, with the corresponding depth map. The information in the hidden layer is not visible from the original view point, but can be disclosed by changing the viewpoint as shown on the right: the light part of the background is visible there. Note that the hidden layer only contains data for parts of the image that are occluded from the original viewpoint and visible from the viewpoint of another recording.

The original implementation of Chang and Zakhor's MVR has several drawbacks. One of the drawbacks is related to the merge step of multiple frames. Generation of depth is done for each image separately, which requires a step of warping to the reference frame and clustering to handle noise and outliers in the depth maps. The clustering step is error prone.

We are able to generate a hidden layer in a manner that is more consistent than the original paper of Chang and Zakhor. In the following, we sketch our algorithm for the computation of the hidden layer. The main focus of the paper is on rendering, which is explained in Section 4.2.

We generate the hidden layer as part of the 2D-to-3D conversion described by Ernst *et al.* [4]. The basics steps in the 2D-to-3D conversion of video data are segmentation and motion estimation.

First, we segment the base image, i.e., we group the pixels of the input image into regions that do not overlap object boundaries. Next, we use another frame, the *secondary image*, of the same scene that is recorded from a different viewpoint, and for each segment, the motion vector that maps the segment to the secondary image is estimated. The segment-based motion estimation algorithm is based on the 3DRS motion estimation algorithm [3], which is adapted for the use of segments. An extensive description of segment-based motion estimation can be found in the paper of Ernst *et al.* [4] .

Now, the depths of the segments have to be found. First, we motion-compensate the image and the motion field (or depth map) to the camera position of the secondary image. This amounts to shifting the segments over their computed motion vector. Parts of the image will not have a predicted segment assigned to them; those are the parts which are deoccluded. These parts will reside in the hidden layer since they are visible in the secondary image, but not in the original one.

The main issue that remains is to determine where the deoccluded pixels would have been located in the original image. Since they are only visible in the secondary image, they can not be matched, and no motion or depth information is available. We know, however, that they must contain background data, otherwise they would not have been occluded by other objects in the original image. We choose to assign the depth of the neighboring background segment in the base image.

As a final step, an inverse motion compensation of the deoccluded pixels is carried out to the viewpoint of the original image. All deoccluded pixels now get a position in the first image, together with a depth or motion vector.

The final result is then a hidden layer, together with the motion or depth for all the pixels in the hidden layer. This hidden layer can now be used in the rendering routine (see Section 4.2).

As a final remark, we mention that the secondary image in the 2D-to-3D conversion need not be an image that was recorded using a stereo camera. Another frame from the same 2D video sequence can also serve as a reference, even when there is motion in the video sequence. In that case, we also need to estimate the camera motion between the frames of the sequence as a prerequisite to compute the depth channel.

4.2. Rendering the hidden layer

In this section, we discuss how to use the hidden layer to improve image quality in rendered views. We recall from Section 3, how we put the problem of rendering from another viewpoint in the context of resampling the original image to the sampling grid imposed by the projection from the desired viewpoint.

In Section 3.2 we presented the implementation of the rendering stage, using a transposed mode FIR pre-filter. An output scanline is generated by traversing the input scanline and use a sliding window of output samples that receive contributions as we progress along the scanline.

When traversing the base layer, we monitor the distance between successive midpoints of warped reconstructed input samples. As long as the midpoints indicate minification we directly feed them to the pre-filter structure. In case of magnification, we now have three options.

The first case is when we have minor magnification. In that case, we insert artificial input samples to the pre-filter. In our implementation, we use linear interpolation (of the color and depth values) to generate these artificial samples. We have set the threshold to a magnification factor of 1.5 for this case.

The second case is when we have major magnification. From the discussion in Section 2, it follows that the regions that appear to require major magnification are associated with deocclusions. Hence, we now try to locate input samples from the hidden layer that could be inserted to fill in the deoccluded background data.

We recall that the hidden layer is specified in the coordinate frame of the base layer. In order to be able to efficiently fill in contributions from the hidden layer, we interleave processing of the scanline the base image layer, and the hidden layer. For both scans, we maintain the extent in the output scanline. This way, we only perform a single scan over the base image scanline interleaved with a single scan over the hidden image scanline.

The third case applies when we cannot find the input samples in the hidden layer. Then, we apply a higher order reconstruction filter r, to compute the values of the output pixels.

In Appendix 5, we show example output of our renderer. Figure 10 shows a video frame after viewpoint transformation that was generated without the use of a hidden layer. Figure 11 shows a frame that was generated using a hidden layer.

We note that while the quality is optimal if the viewpoint of the rendered image is between those of the original and matched image, the rendering stage can always either use input samples from the base layer, or from the hidden layer. If there is no hidden layer information available, then we need to fall back to the reconstruction filter for the magnified areas in the final image.

5. Results

In this section, we show example output from our rendering routines. Figures 8 and 9 show video frames after viewpoint transformation without and with higer order filtering respectively. Figure 10 shows a video frame after viewpoint transformation that was generated without the use of a hidden layer. Figure 11 shows a frame that was generated using a hidden layer. The ghost edge artefact is due to an error in the depth information in the input stream.

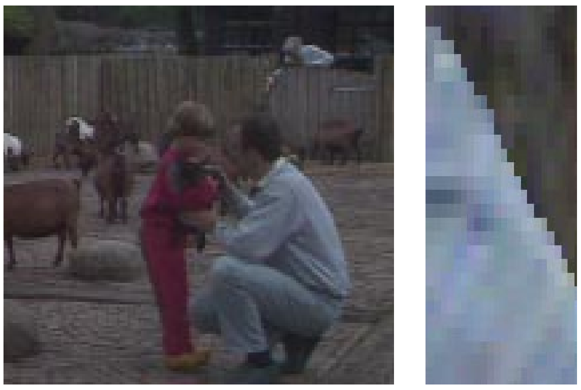

Figure 8: *Image generated without higher order filtering. The viewpoint is left to the viewpoint of the original camera. Aliasing is visible at the boundary of the man's back*

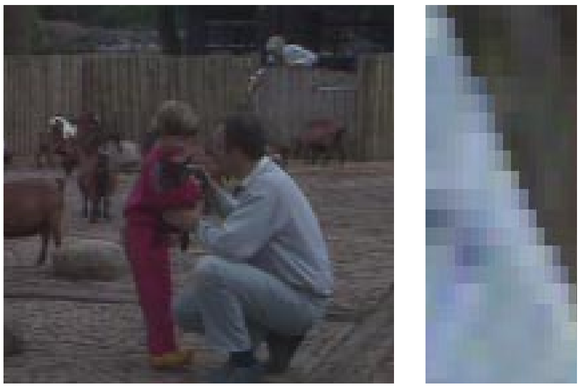

Figure 9: *Image generated with four-tap horizontal filtering. The viewpoint is left to the viewpoint of the original camera. No aliasing at the boundary of the man's back*

6. Conclusions

In this document, we have described a way to integrate higher order video filters in the rendering stage of rendering 2.5D video that supports horizontal parallax. This approach is suitable for, e.g., 3D televisions, or systems that enable viewers to change the viewpoint. We summarize the most important properties of the rendering stage.

The renderer can process streaming video, without the use of a full frame buffer. Each input scanline is traversed only once. The *extent* to which a output scanline is rendered increases monotonically, i.e., the renderer never has to rerender parts of the output scanline. This extent is maintained in a single variable during the processing of the scanline. During the traversal of a scanline, the renderer computes the

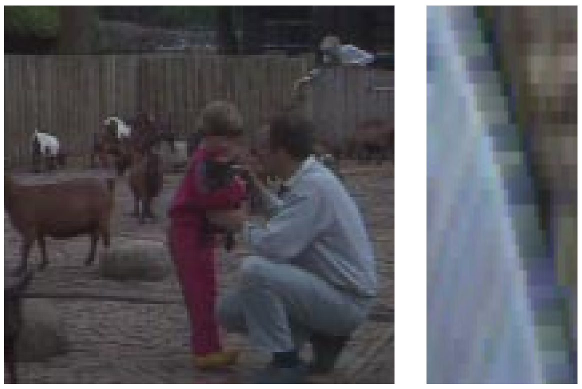

Figure 10: *Image generated using linear interpolating reconstruction filter. The viewpoint is right to the viewpoint of the original camera.*

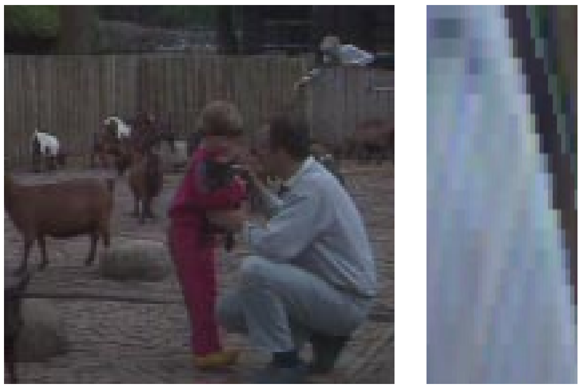

Figure 11: *Image generated using hidden layer rendering. The viewpoint is right to the viewpoint of the original camera.*

magnification factor per input-image pixel. Pixels that appear occluded can be dropped on the fly, the other pixels can be fed immediately to a FIR video filter block that pre-filters and samples the output scanline at screen resolution.

In the rendering process, we can also incorporate the input of hidden layers that contain information at potential deocclusions. The renderer can handle these hidden layers and retain the above properties of the renderer for a single layer.

References

1. D. Anderson. Hidden line elimination in projected grid surfaces. *ACM Transactions on Graphics*, pages 274–288, 1982. 3
2. N.L. Chang and A. Zakhor. Constructing a multivalued representation for view synthesis. *International Journal of Computer Vision*, 45:157–190, 2001. 5
3. G. de Haan and P. Biezen. Sub-pixel motion estimation with 3D recursive search block matching. *Signal Processing: Image Comm.*, 6:229–239, 1994. 6
4. F. Ernst, P. Wilinski, , and K. van Overveld. Dense structure-from-motion: an approach based on segment matching. In *Proc. ECCV, LNCS 2531*, pages 217–231. Springer, 2002. 1, 6, 6
5. J. Gomes and L. Velho. *Image Processing for Computer Graphics*, pages 208–209. Springer, 1997. 4
6. P.S. Heckbert. *Fundamentals of Texture Mapping and Image Warping*. PhD thesis, Department of EECS, U.C. Berkeley, 1989. 2, 3
7. G.J. Iddan and G. Yahav. 3D imaging in the studio (and elsewhere...). *SPIE SMPTE Journal*, 42983D, 1994. 1
8. J.G.W.M Janssen, J.H. Stessen, and P.H.N. de With. An advanced sampling rate conversion technique for video and graphics signals. In *Sixth International Conference on Image Processing and its Applications*, volume 2, pages 771–775, 1997. 4
9. L. McMillan. *An Image-Based Approach to Three Dimensional Computer Graphics*. PhD thesis, UNC Computer Science, TR97-013, 1997. 3
10. K. Meinds and B. Barenbrug. Resample hardware for 3D graphics. In *Proceedings of Graphics Hardware 2002*, 2002. 1, 3, 4
11. M. M. Oliveira, G. Bischop, and D. McAllister. Relief texture mapping. In *Proceedings of ACM Siggraph*, pages 359–368, 2000. 1
12. M. Pollefeys, R. Koch, M. Vergauwen, B. Deknuydt, and L. Van Gool. Three-dimensional scene reconstruction from images. In *Proceedings SPIE Electronic Imaging, Three-Dimensional Image Capture and Applications III, SPIE Proceedings series*, volume 3958, pages 215–226, 2000. 1
13. V. Popescu, J. Eyles, A. Lastra, J. Steinhurst, N. England, and L. Nyland. The warpengine: An architecture for the post-polygonal age. In *Proceedings of ACM Siggraph*, pages 433–442, 2000. 1
14. A. Redert, M. Op 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. In *Proc. 1st Int'l Symposium 3D data processing, visualization and transmission*, 2002. 1
15. P.A. Redert. *Multi-viewpoint systems for 3-D visual communication*. PhD thesis, Delft University of Technology, 2000. 1
16. G. Wolberg and T.E. Boult. Separable image warping with spatial lookup tables. *Computer Graphics*, pages 369–377, 1989. 5
17. M. Zwicker, H. Pfister, J. van Baar, and m. Gross. Surface splatting. In *Proceedings of ACM Siggraph*, pages 371–378, 2001. 1, 3