DOI: 10.1111/cgf.12422
COMPUTER GRAPHICS *forum*
Volume 33 (2014), number 8 pp. 145–156

Optimized Generation of Stereoscopic CGI Films by 3D Image Warping

José M. Noguera[1,2], Antonio J. Rueda[1], Miguel A. Espada[3] and Máximo Martín[3]

[1]Grupo de Gráficos y Geomática de Jaén, University of Jaén, Jaén, Spain
{jnoguera, ajrueda}@ujaen.es
[2]Central Broadcaster Media, Granada, Spain
[3]Greyman Studios S.L., Ogíjares (Granada), Spain
{maespada, mmartin}@greyman.es

Abstract

The generation of a stereoscopic animation film requires doubling the rendering times and hence the cost. In this paper, we address this problem and propose an automatic system for generating a stereo pair from a given image and its depth map. Although several solutions exist in the literature, the high standards of image quality required in the context of a professional animation studio forced us to develop specially crafted algorithms that avoid artefacts caused by occlusions, anti-aliasing filters, etc. This paper describes all the algorithms involved in our system and provides their GPU implementation. The proposed system has been tested with real-life working scenarios. Our experiments show that the second view of the stereoscopic pair can be computed with as little as 15% of the effort of the original image while guaranteeing a similar quality.

Keywords: novel applications of the GPU, rendering, image-based rendering

ACM CCS: I.3.3 [Computer Graphics]: Picture/Image Generation Display algorithms

1. Introduction

The year 2009 marked the final consolidation of stereoscopic 3D films. Although several critics argue against the interest of this format, currently most commercial theatres support the stereoscopic technology and almost every blockbuster comes with a 3D version, especially in the case of animation films.

In this paper, we focus on the production of stereoscopic computer-generated animation films [SKK*11]. During the creation of these films, modelling programs require a special setup with two synthetic parallel cameras as described in [WDK93]. This camera configuration enables two views of the same scene to be captured which when provided to each eye separately, create the illusion of a real 3D image in the viewer's brain.

Rendering complex 3D models and scenarios with film quality implies very high computation times that have a major impact on the final production costs. Each frame usually requires a processing time of several hours, which is multiplied by two in the case of stereoscopic 3D films.

In this work, we have developed a technique to reduce this extra cost. It is based on the 'Depth Image–Based Rendering' (DIBR) [Feh04] paradigm, that allows reuse of the information in the image of one eye to generate the image of the second one, avoiding a full rendering. Although DIBR techniques are not new, they have never been applied to generating stereoscopic images for animation films. In this work, we solve the problems involved with these techniques, adapting them to the high-quality visual requirements of the film industry. In the last part, the results are evaluated and compared to the original images using several real production frames. Therefore, the main contributions of this paper are:

- A simple and efficient GPU-based method to generate the image of the second eye from the image of the first eye using its depth map.
- A novel approach to build a mask with the areas of the second image that we cannot infer from the first image.
- Methods for ensuring the high-quality visual requirements of animation films, avoiding problems arising from overlapping objects, artefacts at the edges of the objects, etc.

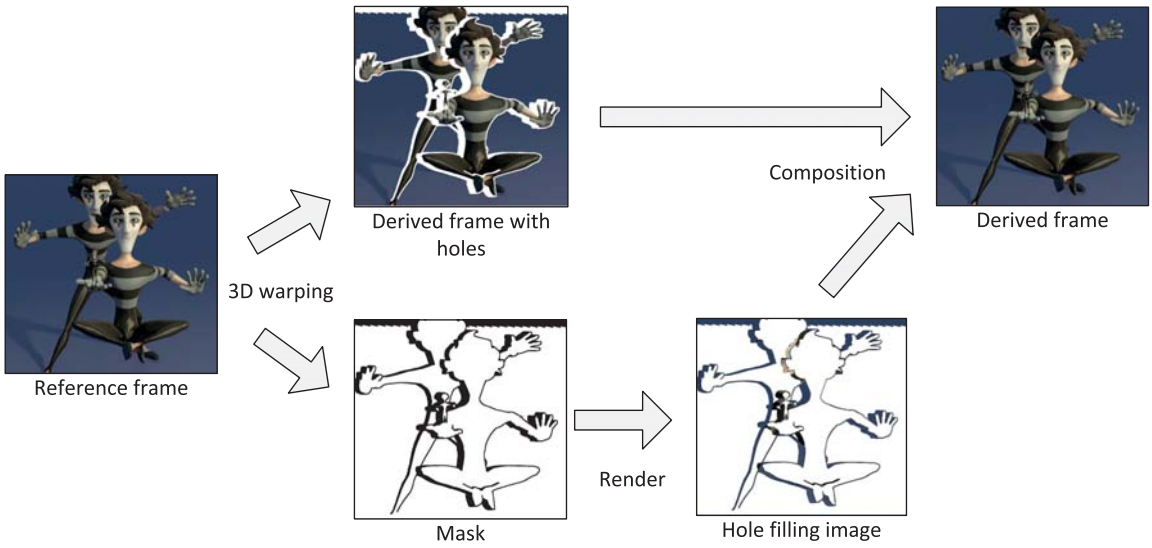

Figure 1: *An overview of the proposed method.*

The rest of the paper is organized as follows. Section 2 provides a necessary background and contextualizes our work. Section 3 presents and describes our proposal. Following, Section 4 provides an empirical evaluation of our method. These results and the limitations of our method are discussed in Section 5. Finally, Section 6 concludes the paper and outlines future work.

2. Previous Work

Image-based modelling and visualization techniques have received considerable interest in the literature as a feasible alternative to traditional geometry-based rendering techniques [SK00]. According to the amount of geometric information used to synthesize an image, we can classify the various image-based rendering methods as follows [CSN07]:

- Methods purely based on images that require no explicit geometry. These methods determine the geometry in an implicit way, e.g. by determining correspondences between a small set of images (see [LH96, GGSC96]). They can be used to synthesize new views of an object from a set of pre-defined images.
- Methods that require explicit geometry, e.g. depth values or explicit 3D coordinates. There exist different approaches in the literature depending on the known geometric properties [SKK*11], e.g. DIBR [Feh04], layered depth images (LDIs) [SGHS98] and intermediate view reconstruction (IVR) [ZKU*04].

In the context of stereoscopic filmmaking, we are mostly interested in the second kind of techniques, as only the frame corresponding to one eye of the stereoscopic camera is available, and our aim is to synthesize the second one. Specifically, our work focuses on the DIBR solutions [MMB97, McM97, Feh04, ZT05], which allow the generation of new views of a scene from an existing image and its associated depth map. This depth map is usually available in the production of a film, as it is heavily used in several stages of the post-production.

From a conceptual point of view, the DIBR techniques consist of two steps. First, the pixels of the original image are unprojected to their original 3D location using their respective depth values, and next, these 3D points are projected again according to the second viewpoint. This concatenation of 2D-to-3D and 3D-to-2D projections is usually referred to as 3D warp. As a result, the pixels of the image are shifted to generate a new one as it were captured from the second viewpoint.

However, the main drawback of these techniques is that holes can arise in the new image (also known as disocclusions) from the lack of information about newly exposed areas which were occluded in the original image [ZVK11]. Some authors suggest pre-processing the depth map in order to reduce the apparition of holes during the warping [PKGS13] at the risk of modifying the resulting warped image. However, most solutions reported in the literature directly assume the apparition of these holes and try to fill them afterwards. For example, some authors [MMB97, PSM04] employ several images in order to recover the missing information of the new image. Unfortunately, these solutions do not apply in our context, as only one frame is available. When the holes are small enough, simple inter- and extrapolation [SKK*11] can be enough. For large holes, in-painting methods based on texture replication and/or structure continuation [TLD07, CLL11] are suggested.

Vázquez *et al.* [VTS06], on the other hand, presented a study comparing different hole filling methods varying in complexity from a very simple filling with a constant colour to a more complex variational in-painting. However, they concluded that the success rate of these methods heavily depends on the scene characteristics and the size of the holes. Also, the synthesized regions typically present a blurred look [CLL11]. Therefore, these solutions lack

the accuracy and robustness required for a professional filmmaking environment.

3. Method

The approach described in this paper has been specifically designed for its use in the production of animation films, where it is safe to assume the availability of the entire 3D structure of the scenes and the corresponding textures required for rendering the film. Our method is based on the DIBR method but is novel in the following aspects:

- In contrast to other approaches, we do not try to infer the missing information of the warped image. Instead, we perform a selective partial render of the 3D scene to recover this information while guaranteeing the maximum visual quality.
- We determine a Boolean mask on the fly, which is used to drive the selective rendering of the 3D scene.
- We also avoid visual abnormalities introduced by overlaps [CW93] and the lack of full-screen anti-aliasing filtering on the warped image.
- It can be fully implemented in the GPU.

The main idea behind our method resides in the fact that rendering small areas of a frame requires considerably less time than rendering the whole frame. In what follows, we will refer to the frames rendered in the conventional manner as reference frames, and the frames inferred by our method as derived frames.

Figure 1 graphically summarizes our proposal. The process begins with the rendering of the reference frame in the usual way. From that point, our method can be summarized in the following steps:

(1) Warp the reference frame to obtain a preliminary derived frame.
(2) Detect the pixels of the preliminary derived frame that belong to disoccluded areas. Mark these pixels in a Boolean mask.
(3) Extend this mask to include the boundary of the objects of the scene. This step is required to remove potential jagged (i.e. not anti-aliased) edges that can be introduced during the warping process.
(4) Carry out a selective render of those pixels included in the mask. Combine the resulting partial image with the preliminary derived frame to generate a complete derived frame.

The following sections describe these steps in detail. A GPU implementation of our method is also provided. Figure 2 depicts this implementation.

3.1. Step 1: image 3D warping

Let us consider a stereoscopic camera system composed of two individual cameras, A and B, following a parallel configuration [WDK93]. Both cameras share the same intrinsic parameters and viewing line, but are located along a horizontal line that contains their respective focal points. This configuration guarantees that the two images obtained by the stereoscopic camera do not present vertical parallax.

```
//output to the geometry processor
out VertexData {
  vec2 texCoord;
  float worldDepth;
} VOut;

//camera A matrices
uniform mat4 Mp;
uniform mat4 Mv;
//camera A location
uniform vec3 OA;
//camera B matrices
uniform mat4 Mp';
uniform mat4 Mv';
//depth map
uniform sampler2D Z;
//vertex in screen coordinates
in vec4 Vs;

void main()
{
  vec2 texCoord = (Vs.xy + vec2(1))/2.0;
  float ZVs = texture2D(Z, texCoord).r;
  mat4 inverseM = inverse(Mp * Mv);
  vec4 Vw = inverseM * Vs;
  Vw = Vw/Vw.w;
  vec3 Vw' = OA + ZVs * normalize(Vw.xyz-OA);
  gl_Position =  Mp' * Mv' * vec4(Vw', 1.0);
  VOut.texCoord = texCoord;
  VOut.worldDepth = Vw'.z;
}
```

Listing 1: *GLSL vertex program of the first pass of the method.*

Now let us consider an image that represents the projection in a 2D plane of a natural 3D scene across the projection defined by the camera A. This image is composed of a regular grid of pixels. Let Z be a depth map defined over the same grid, which contains the distance from the focal point of the camera A to the corresponding point of the 3D scene projected onto such pixel. The 3D warping method aims at generating a new derived frame equal to the image that would result from projecting the same scene from the camera B.

This 3D warping process can be efficiently implemented in the vertex processor of the GPU (see Figure 2). Listing 1 shows the GLSL code of our proposed implementation.

First, a triangulated planar mesh is generated and textured with the reference frame. The vertex dimension of this mesh is coincident with the pixel resolution of the reference frame. Then, the mesh is placed parallel to the projection plane of camera A ensuring that it perfectly occupies the viewing frustum as shown in Figure 3(a). This can be easily performed by regularly spacing the vertices of the mesh between $(-1,-1)$ and $(1,1)$, expressed in screen coordinates. Their z-coordinate is irrelevant as long as it is in the interval $(0,1)$. Next, each vertex V_s of the mesh in screen coordinates is unprojected in order to obtain its equivalent V_w in world coordinates. This transformation requires the multiplication of the coordinates of each vertex with the inverse of the product between the projection and viewing matrices of the camera A, M_p and M_v, respectively.

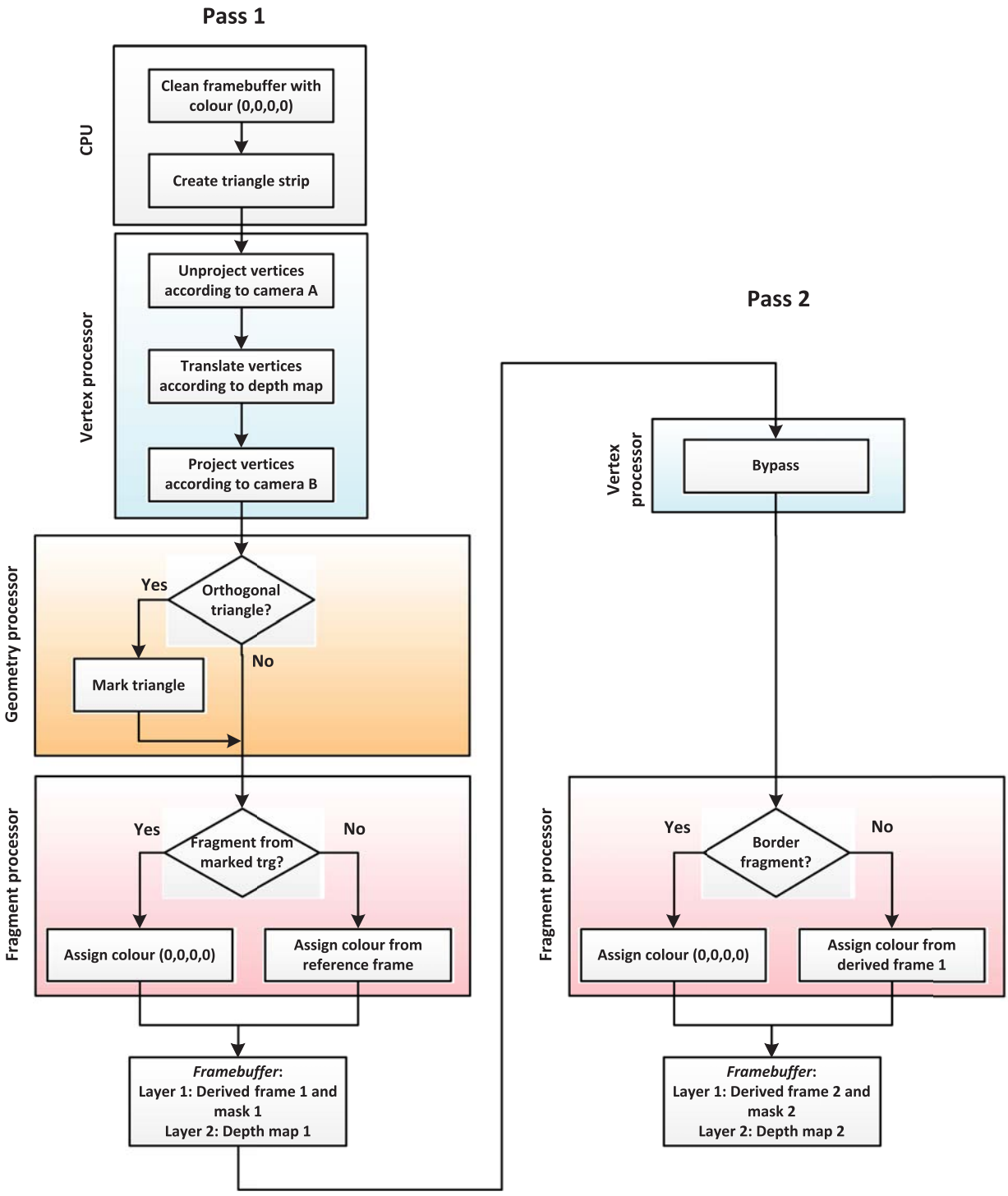

Figure 2: *GPU implementation of the method.*

Similarly to [MMB97], these mesh vertices are perturbed to obtain a new surface that approximates the 3D scene. The perturbation consists in a translation of each vertex V_w of the mesh along the line $V_w O_A$ that connects such vertex with the focal point O_A of the camera A. The distance of the translation is defined by the depth value of the pixel $Z(V_s)$ corresponding to each vertex V_s:

$$V_w = (M_p M_v)^{-1} V_s,$$

$$V'_w = O_A + Z(V_s)(V_w - O_A).$$

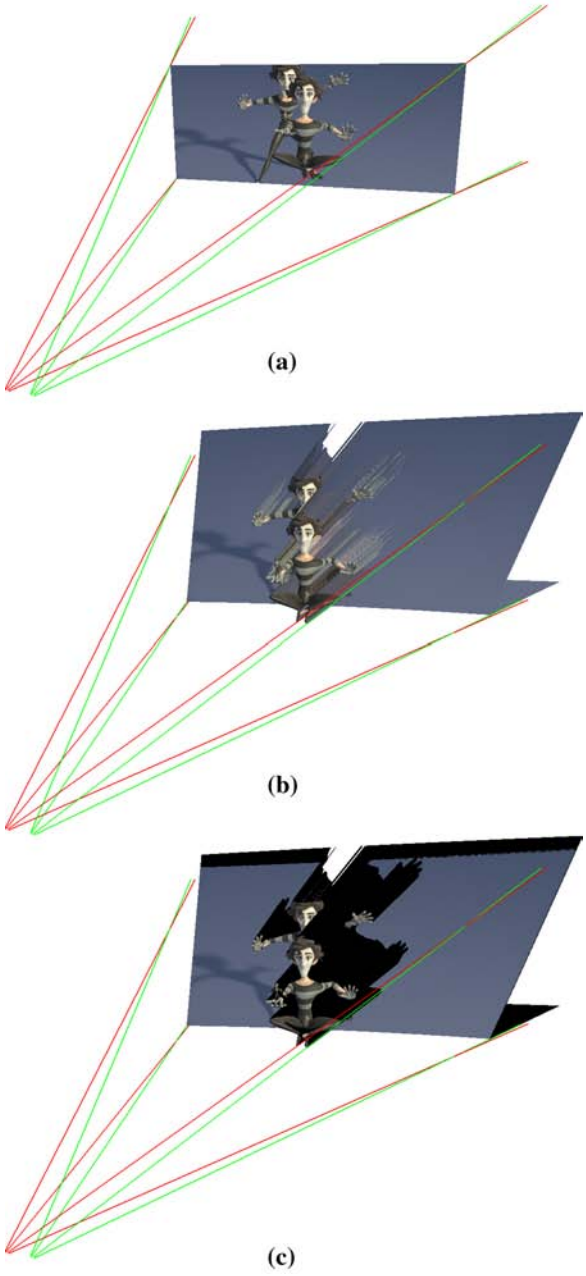

Figure 3: *The lines represent the frustum of the stereoscopic cameras. (a) Triangular mesh inserted in the camera A frustum. (b) The same mesh after translating its vertices according to the depth map. (c) The triangles in black conform the mask.*

Given the tessellated nature of the original mesh, the resulting mesh is still a continuous surface that can contain folds and stretched triangles. The colour of the mesh is determined by colour interpolation between the colours of its vertices. Thus, the use of triangles instead of points as basic primitives avoids the apparition of geometric and colour discontinuities on this surface.

```
//lambda, in OpenGL world coordinates
uniform float λ;

layout(triangles) in;
layout (triangle_strip, max_vertices=3) out;

//input from the vertex processor
in VertexData {
  vec2 texCoord;
  float worldDepth;
} VIn[];

//output to the fragment processor
out VertexData {
  vec2 texCoord;
  float worldDepth;
  float rubber_sheet;
} VOut;

//some definitions to simplify the code
#define v0 gl_in[0].gl_Position
#define v1 gl_in[1].gl_Position
#define v2 gl_in[2].gl_Position

void main()
{
  VOut.rubber_sheet = 0.0;
  float depthRange =
    max( abs(v0.z-v1.z), abs(v0.z-v2.z) );
  if( depthRange >= λ ){
    VOut.rubber_sheet = 1.0;
  }
  for(int i = 0; i < gl_in.length(); i++){
    gl_Position = gl_in[i].gl_Position;
    VOut.texCoord = VIn[i].texCoord;
    VOut.worldDepth = VIn[i].worldDepth;
    EmitVertex();
  }
  EndPrimitive();
}
```

Listing 2: *GLSL geometry program of the first pass of the method.*

Finally, the derived frame is obtained by projecting the perturbed mesh according to the matrices that define the camera B, i.e. M'_p and M'_v:

$$V'_s = M'_p M'_v V'_w.$$

The proposed GPU implementation of this step ends with the emission of the vertex to the following step of the graphics pipeline (see Listing 1).

3.2. Step 2: mask construction

A major source of problems with 3D image warping algorithms is the apparition of visual artefacts between objects in the derived frame. As the mesh is treated as a continuous surface, implicit surfaces are introduced at silhouette boundaries between foreground and background objects of the image [MMB97]. These implicit surfaces were not present in the original 3D scene, and are formed by stretched triangles as shown in Figure 3(b). In the literature, these

```
//input from the geometry processor
in VertexData {
  vec2 texCoord;
  float worldDepth;
  float rubber_sheet;
} VIn;

//output: derived image and mask
layout(location = 0) out vec4 colorOut;
//output new depth map
layout(location = 1) out float depthOut;

uniform sampler2D RGB;

void main()
{
  if( VIn.rubber_sheet == 1.0 ){
    colorOut = vec4(0,0,0,0);
    depthOut = 0;
  }else{
    colorOut = vec4(
      texture2D(RGB, VIn.texCoord).rgb,
      1.0);
    depthOut = -VIn.worldDepth;
  }
}
```

Listing 3: *GLSL fragment program of the first pass of the method.*

are usually known as *rubber sheet triangles or surfaces* [MMB97, PSM04].

This section describes how our algorithm overcomes this problem by detecting the problematic pixels resulting from the projection of the rubber sheet surfaces. These pixels are marked in a mask for their subsequent rendering in the fourth step of our method. The mask can be determined in an efficient way by means of the GPU. Listings 2 and 3 show the GLSL code of the geometry and fragment programs, respectively, of our proposed implementation.

Conceptually, the mask is represented by an image with the same pixels dimension as the derived image. Each pixel stores a Boolean flag indicating the inclusion of the corresponding pixel of the derived image in the mask. In practice, our proposed GPU implementation stores the mask M in the same RGBA framebuffer as the derived image I. The mask is formed by the pixels $P_{i,j}$ of I that fulfil the following condition:

$$M = \{P_{i,j} \in I : P_{i,j} = (0, 0, 0, 0)\}.$$

Also, at the beginning of the rendering process the framebuffer is initialized with this colour, see Figure 2, ensuring that all pixels that remain unpainted after the warping are included in the mask.

In order to differentiate between legitimate and rubber sheet surfaces of the mesh, we employ a GPU implementation of the orthogonality test on triangles proposed by Pajarola *et al.* [PSM04]. This test is based on the observation that rubber sheet triangles have the following common property: their normal is almost perpendicular to the vector from the viewpoint to the centre of the triangle. As a consequence, rubber sheet triangles span a greater depth range Δz in the camera coordinate system than the rest of the triangles. In our solution, we use the geometry shader to check whether each triangle satisfies $\Delta z > \lambda$ for some threshold λ (see Listing 2).

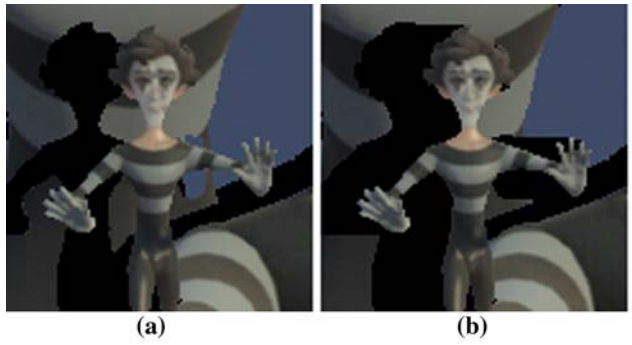

Figure 4: *Detail of a derived frame. Black areas correspond to the mask. (a) Rubber sheet triangles are discarded, resulting in holes and overlaps. (b) Rubber sheet triangles are added to the mask, avoiding such problems.*

Once a rubber sheet triangle is detected, an immediate solution could be discarding it from the mesh containing the reference frame, an operation that can be easily implemented in the geometry processor. Unfortunately, such action would break the mesh continuity, resulting in holes on its surface. Given that the 3D warp can produce folds in the warped mesh, several pixels from the reference frame can be warped to the same location causing overlaps. These overlaps are usually solved by the z-buffer algorithm of the GPU in an automatic way. But the existence of holes in the mesh could cause occluded background objects to become visible in the derived frame through one of such holes. Figure 4(a) shows an actual example of an overlap. We see a blue object that should be occluded from the new viewpoint but that still remains visible through a hole.

Therefore, these rubber sheet surfaces should not be discarded but instead be added to the mask. We propose a simple, direct method to create the mask by simply assigning a special colour to the rubber sheet triangles, (0, 0, 0, 0) in our implementation. Figure 3(c) depicts the warped mesh with all the rubber sheet triangles coloured in black. When this mesh is projected from camera B, we obtain the image shown in Figure 4(b). When comparing this figure with (a) we clearly see that the rubber sheet triangles have marked the pixels that should be rendered again in step 4 of our method, avoiding the apparition of holes and overlaps.

However, we also see that this criterion can add valid parts of the derived frame to the mask, which in turn increases the rendering cost of the fourth step of our method. But given the high-quality requirements of a film production environment, this is an assumable trade-off since it is preferable to discard valid parts of the image (which can be rendered again anyway) than to risk the visual quality of the derived frame.

Another important observation is that the λ threshold controls this trade-off between performance and visual quality, and therefore it should be adequately chosen for each scene of the movie.

```
//input from the vertex processor
in VertexData {
  vec2 texCoord;
  vec4 color;
} VIn;

layout(location = 0) out vec4 colorOut;
layout(location = 1) out vec4 maskOut;

uniform sampler2D RGB;
uniform sampler2D Z;
//lambda, in OpenGL world coordinates
uniform float λ;
//Width in pixels of the silhouette
uniform int B;
//Inverse of the screen size in pixels
uniform vec2 invImage;

bool silhouette( float currentZ )
{
  for( int i=-B; i<=B; i++ ){
    for( int j=-B; j<=B; j++ ){
      vec2 texC = vec2(
        VIn.texCoord.x + invImage.x*i,
        VIn.texCoord.y + invImage.y*j );
      float neighbourZ = texture2D(Z, texC).r;
      if( abs(neighbourZ - currentZ) > λ )
        return true;
    }
  }
  return false;
}

void main()
{
  colorOut = texture2D(RGB, VIn.texCoord);
  float z = texture2D(Z, VIn.texCoord).r;

  if( silhouette(z) ){
    maskOut = vec4(1,1,1,1);
    colorOut = vec4(0,0,0,0);
  }
}
```

Listing 4: *GLSL fragment program of the second pass of the method.*

3.3. Step 3: anti-aliasing treatment

Anti-aliasing is a view-dependent filter that blends silhouette pixel colours from a particular viewpoint. This filter is an unavoidable step for achieving the desired quality for a cinematographic production. Unfortunately, it poses some additional difficulties that should be addressed if we want our method to be useable under real-world studio conditions.

The first problem caused by these filters stems from the depth map. When rendering the reference frame, the depth map should not be anti-aliased in any way. Otherwise, the anti-aliased silhouette pixels of the image may attach to both foreground and background objects causing ambiguity in the subsequent warping process [CW93]. Fortunately, this problem can be avoided by turning off the filtering of the depth map, an option that is typically available on most render engines.

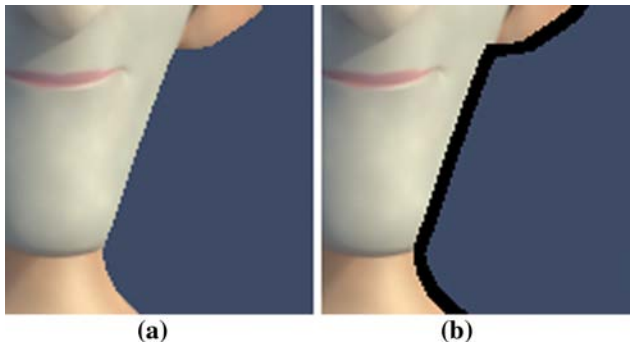

Figure 5: *(a) Jagged, non anti-aliased borders can be introduced by the warping process. (b) All silhouettes between foreground and background objects are added to the mask.*

A second, more problematic issue is caused by the view-dependent nature of the anti-aliasing filters. The folds generated in the mesh after the warping can introduce new edges in the derived frame that will look unacceptably jagged due to the lack of a proper anti-aliasing filtering. Figure 5(a) depicts a clear example of a jagged silhouette introduced by the warping process.

Our solution to overcome this problem consists in performing a second GPU pass, see Figure 2, that uses as inputs the derived frame and the mask generated after the previous steps of the method. This additional pass carries out a depth-based silhouette detection algorithm over the derived frame in order to add the silhouettes to the mask.

Listing 4 shows the GLSL code that implements this solution. For each fragment, the filter determines the greater depth-range Δz between the fragment and its neighbourhood. The fragment is added to the mask if it verifies that $\Delta z > \lambda$ for the λ threshold defined in Section 3.2. Figure 5(b) shows the result after applying this filter. The width of the silhouette border is a configurable parameter that controls the trade-off between visual quality and performance.

Note that this solution is very conservative in the sense that it can potentially discard valid parts of the image. But as remarked in Section 3.2, it is an assumable sacrifice in order to ensure the high visual quality requirements considered in this work.

3.4. Step 4: partial render and composition

After steps 1–3, we obtain a derived frame that contains only a partial view of the scene as seen from camera B and a mask of invalid pixels. In this step, we call the render engine used in the production to make a partial render of the original scene from camera B using the mask. It goes without saying that this new render pass is much faster than that required for the reference frame, as under normal conditions the mask only includes a fraction of the pixels of the reference frame. As a result of this step, we obtain a new image containing the required pixels to fill the holes of the derived frame. Our method ends with the composition of both images to generate the complete derived frame.

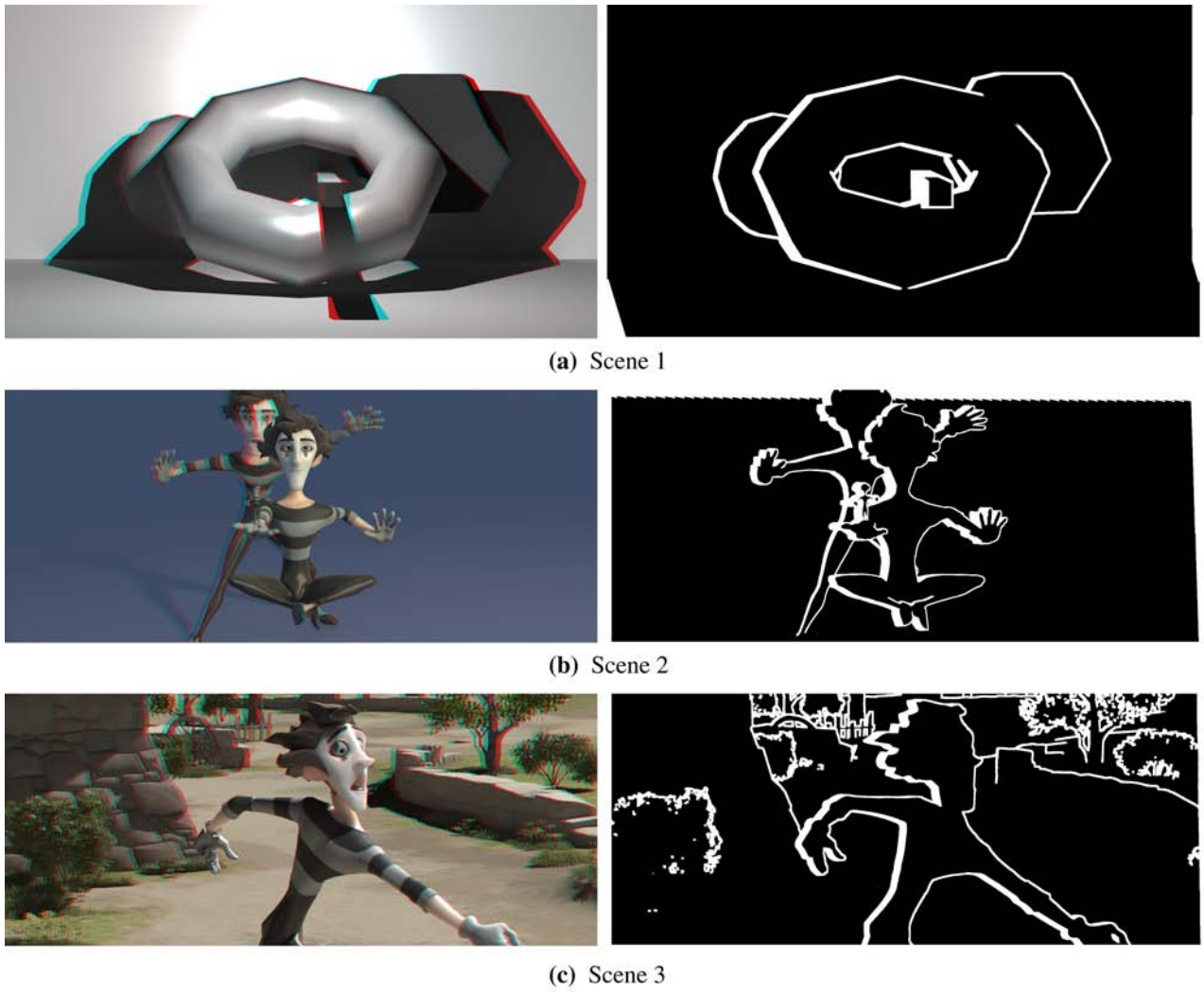

Figure 6: *Scenes used in our experiments. © Kandor Graphics S.L.*

4. Results

This section describes the experimental evaluation carried out to validate our method. We have assessed the quality of our method from a quantitative point of view with diverse error metrics. A user study was also carried out to validate the perceptual quality of our solution. Finally, the performance of the new proposal was measured and contrasted against the usual procedure.

Figures 6 and 7 show the scenes used in our experiments. For each scene, we show the anaglyph generated by our method and the corresponding mask. Scenes from Figure 6 were specifically created to test our method so they feature a strong parallax. Scenes from Figure 7, on the other hand, came directly from the industry and therefore they make a more prudent use of the stereoscopy. These scenes were selected to cover a large number of situations, including indoor and outdoor scenarios, vegetation, close ups and specular reflections. The parameters of the method (λ and the width of the silhouette borders) have been carefully selected in order to minimize the area of the mask while ensuring the removal of all rubber sheet triangles on the warped frame. Note that the anaglyphs are encoded as red (left) and cyan (right) images. For an optimal viewing experience, we suggest watching the images zoomed in.

4.1. Image quality

The purpose of this study was to assess the quality of the frames inferred by our solution. Note that we do not explicitly aim at generating geometrically consistent images. On the contrary, our goal is to ensure that the frames generated by our method have the same visual and stereoscopic quality that the original ones from a perceptual point of view.

Table 1 compares the quality of the inferred frames using as a reference the fully rendered frames. Two different error metrics have been used in the study, the root-mean-square deviation (RMSE) and the structural similarity index (SSIM) proposed by [WBSS04].

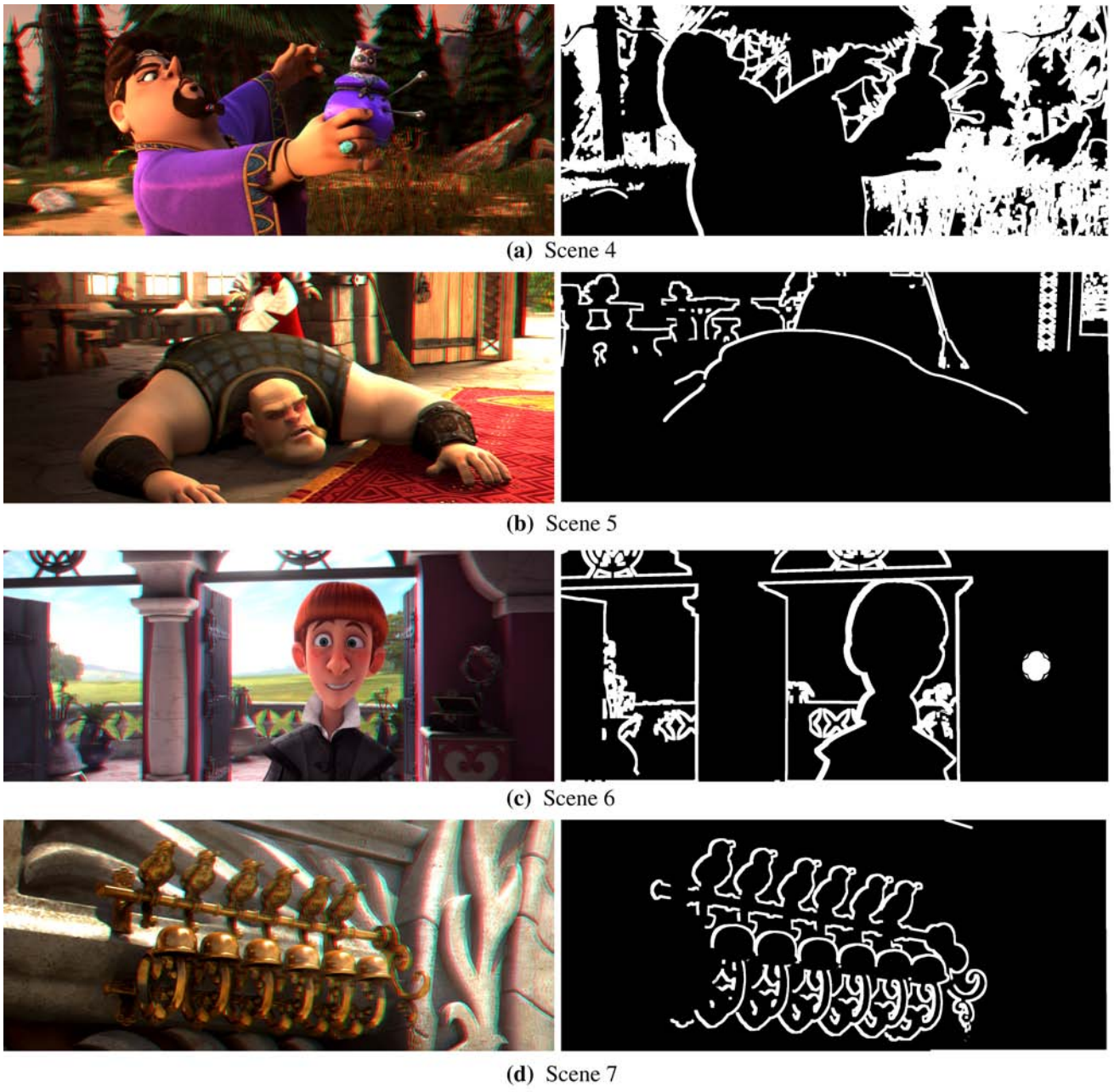

Figure 7: *Scenes used in our experiments. Images from the film 'Justin and the Knights of Valour' © Kandor Graphics S.L.*

These metrics are reported in columns 2 and 3 of Table 1. RMSE values range from 0 to 255, whereas SSIM ranges from 0 to 1. RMSE = 0 and SSIM = 1 mean no difference to the original frame. The last column of Table 1 reports the size of the mask for each scene during our experiments, that is, the proportion of the derived frame that had to be rendered in the traditional way.

The RMSE is a widely used quality metric that is very convenient in the context of optimization. However, this metric is very limited for this type of comparison. For example, a small shift of just one pixel in the projection of the warped image is almost imperceptible for a human, but it can lead to a high RMSE error. Therefore, perceptual metrics such as SSIM are an interesting alternative to quantify the perceived visual quality of an image according to the characteristics of the human visual system.

As Table 1 reports, the RMSE varies from 2.8 to 0.4. However, the SSIM remains above 0.99 in most cases, which shows that our method can be used to obtain images that are virtually identical to the original.

Table 1: *RMSE and SSIM values of a comparison between the derived and the original frame for different data sets. The last column reports the proportion of the frame covered by the mask.*

Data set	RMSE	SSIM	Mask (%)
Scene 1	2.8180	0.9908	5.7481
Scene 2	0.5937	0.9940	10.0553
Scene 3	1.2740	0.9932	11.7979
Scene 4	0.5883	0.9957	37.0910
Scene 5	0.7691	0.9968	7.4913
Scene 6	0.4890	0.9963	12.4791
Scene 7	2.5379	0.9856	10.2763

In addition, a user study was carried out to further validate our solution. The research question to be answered in this study was whether our warping method generates images with the same perceivable visual and stereoscopic quality as the original ones.

Forty-one subjects volunteered to participate in this study. Their ages varied from 22 to 60 years old, the average age being 33.6 ± 1.5 years. The evaluators presented a good technological background. Most of them (92.5%) had already watched at least one 3D film. Also, around half of them (45.0%) affirmed having watched at least one during the last year. The study was conducted with a 47-inch *LG 47LW65* full HD screen which features passive 3D technology.

After a brief explanation of the purpose of the study, we presented them with a pairwise comparison of the 3D images shown in Figures 6 and 7. Each comparison comprised the original 3D frame, generated by two independent full renders and the equivalent frame inferred by our method. The locations of the original and inferred images on the screen were taken randomly. The scenes, on the other hand, were always presented in order, from scenes 1 to 7. Evaluators could freely move back and forth during the evaluation. Also, they were not subjected to any time restriction. For each comparison, the evaluators had to answer the following question: '*which image has a better visual quality?*' The possible answers were: top, bottom or '*I do not know*' (N/K). Their answers were collected by means of an anonymous questionnaire.

Data collected from the questionnaires were imported in R for statistical analysis. In total, we received 287 votes. Overall, 161 votes (56%) answered N/K, 60 votes (21%) successfully recognized the image rendered in the usual way as the image with superior visual quality, but on the contrary, 66 votes (23%) thought that the image inferred by our method presented better visual quality. From these results, we clearly see that N/K was the predominant response. That is, in most cases the evaluators either found the images to be equal or were unable to identify which one had better quality.

Following, for each evaluator we counted the number of times that he/she opted for each of the two images and calculated the difference between both values. N/K votes counted as zero. Then, we analysed the median of these differences with the Wilcoxon signed rank test in order to check whether it was different to zero. The α level was considered significant at $p < 0.05$. We used the Wilcoxon test because our samples cannot be assumed to be normally distributed. The results of the test ($p = 0.674$) do not contradict the null hypothesis (the median is zero) and therefore we found no evidence of the evaluators preferring traditional over inferred images or vice versa.

In addition, we felt that a global analysis could be hiding potential differences in the evaluation of a particular scene. Therefore, we carried out an individualized analysis of the data collected for each one. In this test, for each scene we compared the number of evaluators that preferred the original version over the corresponding one inferred by our method. The N/K votes did not sum to either method. In this case, the Wilcoxon test was equivalent to using a binomial test with a 50% chance of an evaluator choosing each image. We considered an α level of $p < 0.05$. Table 2 summarizes the results. For each image, the table reports the probability of identifying the image rendered in the traditional way as the one with better visual quality (95% CI), and the p-value. Again, the high p-values did not allow us to reject the null hypothesis (the traditional image has a 50% chance of being chosen). That is, we found no significant evidence of the evaluators preferring traditional over inferred images or vice versa for any of the studied scenes.

4.2. Performance

The experimental setup of our performance evaluation consisted of a PC equipped with an Intel Core i7 CPU running at 2.67 GHz with 12 GB of RAM and a GeForce GTX 580 GPU. The software used to render the images was the path-tracer *Arnold* 4.0.11 for Windows 64 bits.

Table 3 shows the time required to render a frame from the left camera of the scenes shown in Figure 6. From left to right, Table 3 provides the name of the scene, the time in seconds required by Arnold to render it in the usual way, the time in seconds required by our proposal to generate the same frame, and finally, the speed-up achieved by our solution.

Note that for providing a fair comparison, the time reported in Table 3 includes all the stages of our method, including (i) reading and adapting the Arnold scene files from disk; (ii) performing the 3D warping; (iii) rendering the holes with Arnold; (iv) compositing the complete frame and (v) deleting all the auxiliary files from disk. Nevertheless, we found that depending on the scene, approximately 95–98% of the time required by our technique was actually employed in rendering the holes with Arnold. That is, the duration of the GPU-based warping process was almost negligible in comparison.

From Table 3, two things should be noticed. First, in all cases the time yielded by our technique is notably inferior to the time required by the traditional approach, with speed-ups ranging from $\times 6$ to $\times 9$. Secondly, the speed-up achieved by our method corresponds to the size of the mask reported in Table 1. Note that there are some fixed costs involved in opening and processing the scene files, which are independent of the number of pixels of the frame to be rendered.

5. Discussion and Limitations

The values for the λ threshold and the width of the border used by our algorithm during the experiments have been chosen to minimize

Table 2: *Exact binomial test in the distribution of votes that recognizes the image rendered in the traditional way as the superior one.*

Scene	CI 95%	p-Value
Scene 1	20.25–66.50	0.647
Scene 2	22.98–72.18	1.000
Scene 3	35.74–82.70	0.480
Scene 4	40.99–86.65	0.237
Scene 5	15.63–55.32	0.151
Scene 6	23.03–76.96	1.000
Scene 7	15.19–64.56	0.454

Table 3: *Performance statistics for several scenes. Time measured in seconds.*

Scene	T (s) traditional	T (s) warping	Speed-up
Scene 1	589.0	62.4	×9.4
Scene 2	803.0	117.2	×6.8
Scene 3	8585.6	1011.9	×8.5

the size of the mask, and therefore limit the impact of the second render, without compromising the quality of the derived image. Consequently, the choice of these parameters is of great importance for the performance of our solution. As a general rule, we can set these parameters for each shot (per frame is impractical), assuming a logical spatial and temporal coherence among the frames of the shot.

During the production, the stereoscopy supervisor proceeds as follows. First, it tries a set of default warping parameters with one or a few frames of the shot, adjusting them as required. Then, the complete set of frames of the shot with the parameters are scheduled for processing. Finally, when the shot is ready, the result is inspected, doing complete or partial retakes with new warping parameters when necessary.

Taking a look at the achieved performance, our improvement compared to a full stereoscopic render is due to the fact that the performance of a raytracer depends on the number of pixels to process, which is dramatically reduced with our approach. The speed-up is always inversely proportional to the mask size. Unfortunately, this speed-up is limited by the setup time required by the raytracer to load geometry, prepare textures, etc. that is dependent on the scene complexity. This time can be considerable and it is fixed regardless of rendering the full image or only the area defined by the mask. Despite this, our experiments have clearly shown that there is a significant advantage in the use of our method even if it involves some additional complexity in the production pipeline.

Regarding the final quality, our quantitative study has shown a negligible difference with a frame rendered in the usual way. The SSIM perceptual metric remains above 0.99 (out of 1) on most cases. The user study suggests that our evaluators did not consider the original frame to have higher visual o stereoscopic quality than the inferred one. In fact, we believe that the small percentage of users that chose one of the images over the other did so because they were possibly conditioned by the knowledge that the images were actually different. Nevertheless, our user evaluation reported no significant differences between the number of hits and misses among this group of users, demonstrating that their choice may well be due to chance.

Our method shows its limits when the scene contains effects depending on the observer's position (specular highlights, reflections, refractions, etc.). This is the reason the error in scenes 1 and 7 is higher than in the rest: they contain multiple objects with specular highlights. Interestingly, when the final stereoscopic pair is viewed these differences in the location of the lights become subtler due to the compensation mechanism inherent to our visual system. The work of Lang *et al.* [LHW*10] reported similar observations.

Motion blur and translucent surfaces (smoke, fire, etc.) are problematic because of the ambiguity between the effect object and the background objects in the depth map. This problem could be avoided or at least mitigated by rendering the effect object and the background in different passes and putting everything together later through compositing. Note that this is the usual procedure in most animation productions. Finally, the problem with reflective surfaces can be fixed adding these surfaces to the mask. For instance, this is what has been done to correctly render the reflection in the little mirror on the right wall in scene 6 of Figure 7.

6. Concluding Remarks

In this paper, we have presented an efficient depth-based method for computing a stereoscopic pair from a reference image and its associate depth map. We have proved that the proposed technique leads to a dramatic reduction in the rendering costs of the second image of the stereoscopic pair without sacrificing its perceived quality. Moreover, the method only requires from a human operator the calibration of two parameters: the λ threshold and the width of the silhouette borders, which typically remain constant for all frames of a given shot of a film.

As limitations of the proposed technique, those visual effects that depend on the view position (such as motion blur, specular highlights, caustics, reflections, refractions, etc.) could cause problems when warped from the position of the second camera. Therefore, as future work we plan to carry out a depth study of the behaviour of our method under such conditions, as well as to find solutions to overcome these potential problems. We also plan to study potential automatic or semi-automatic ways to choose the correct parameters for our algorithm.

Acknowledgements

We would like to thank *Solid Angle* for kindly providing us with a license of their renderer *Arnold*. We also would like to thank Dr. M. Catalina Osuna-Pérez and Dr. Antonio Conde-Sánchez for their technical support.

Copyrights of the used images belong to *Kandor Graphics S.L.*

This work has been partially financed by Kandor Graphics S.L. It has also been partially financed by the Centre for Industrial Technological Development (CDTI) of the Government of Spain through the research project *Innterconecta Adapta* (ITC-20111030).

References

[CLL11] Cheng, C.-M., Lin, S.-J., Lai, S.-H.: Spatio-temporally consistent novel view synthesis algorithm from video-plus-depth sequences for autostereoscopic displays. *IEEE Transactions on Broadcasting 57*, 2 (2011), 523–532.

[CSN07] Chan, S., Shum, H.-Y., Ng, K.-T.: Image-based rendering and synthesis. *IEEE Signal Processing Magazine 24*, 6 (2007), 22–33.

[CW93] Chen, S. E., Williams, L.: View interpolation for image synthesis. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM, pp. 279–288.

[Feh04] Fehn, C.: Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV. In *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI* (San Jose, CA, USA, 2004), vol. *5291*, pp. 93–104.

[GGSC96] Gortler, S. J., Grzeszczuk, R., Szeliski, R., Cohen, M. F.: The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 43–54.

[LH96] Levoy, M., Hanrahan, P.: Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 31–42.

[LHW*10] Lang, M., Hornung, A., Wang, O., Poulakos, S., Smolic, A., Gross, M.: Nonlinear disparity mapping for stereoscopic 3D. In *SIGGRAPH '10: ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), ACM, pp. 75:1–75:10.

[McM97] McMillan, Jr.L.: An Image-Based Approach to Three-Dimensional Computer Graphics. PhD thesis, Chapel Hill, NC, USA, 1997. UMI Order No. GAX97-30561.

[MMB97] Mark, W. R., McMillan, L., Bishop, G.: Post-rendering 3D warping. In *I3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1997), ACM, pp. 7–16.

[PKGS13] Plath, N., Knorr, S., Goldmann, L., Sikora, T.: Adaptive image warping for hole prevention in 3D view synthesis. *IEEE Transactions on Image Processing 22*, 9 (2013), 3420–3432.

[PSM04] Pajarola, R., Sainz, M., Meng, Y.: Dmesh: Fast depth-image meshing and warping. *International Journal of Image and Graphics 4*, 4 (2004), 653–681.

[SGHS98] Shade, J., Gortler, S., He, L.-W., Szeliski, R.: Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), ACM, pp. 231–242.

[SK00] Shum, H., Kang, S. B.: Review of image-based rendering techniques. In *Proceedings of Communications and Image Processing* (2000), K. N. Ngan, T. Sikora and M.-T. Sun (Eds.), SPIE, pp. 2–13.

[SKK*11] Smolic, A., Kauff, P., Knorr, S., Hornung, A., Kunter, M., Muller, M., Lang, M.: Three-dimensional video postproduction and processing. *Proceedings of the IEEE 99*, 4 (2011), 607–625.

[TLD07] Tauber, Z., Li, Z.-N., Drew, M.: Review and preview: Disocclusion by inpainting for image-based rendering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37*, 4 (2007), 527–540.

[VTS06] Vazquez, C., Tam, W. J., Speranza, F.: Stereoscopic imaging: Filling disoccluded areas in depth image-based rendering. In *Proceedings of SPIE 6392, Three-Dimensional TV, Video, and Display V* (Boston, MA, USA, 2006).

[WBSS04] Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612.

[WDK93] Woods, A., Docherty, T., Koch, R.: Image distortions in stereoscopic video systems. In *Proceedings of SPIE: Stereoscopic Dispalys and Applications IV* (San Jose, CA, USA, 1993), vol. *1915*, pp. 36–48.

[ZKU*04] Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 600–608.

[ZT05] Zhang, L., Tam, W. J.: Stereoscopic image generation based on depth images for 3D TV. *IEEE Transactions on Broadcasting 51*, 2 (2005), 191–199.

[ZVK11] Zhang, L., Vazquez, C., Knorr, S.: 3D-TV content creation: Automatic 2D-to-3D video conversion. *IEEE Transactions on Broadcasting 57*, 2 (2011), 372–383.