FAST ALGORITHM FOR CREATING IMAGE-BASED STEREO IMAGES

Przemysław Kozankiewicz

Institute of Computer Science,
Warsaw University of Technology,
ul. Nowowiejska 15/19,
00-665 Warsaw, Poland
pkozanki@ii.pw.edu.pl

ABSTRACT

The process that creates stereo images typically takes from 1.5 to 2 times longer than the algorithm for mono images. This paper presents a fast image-based algorithm for computing stereo images, that needs a little more time than the algorithm for mono images. The presented algorithm uses a single image with depth information (e.g. z-buffer) as an input and produces two images for left and right eye. This algorithm is a simplification of the 3D warping algorithm.

Keywords: stereo-vision, image-based rendering, warping

1 INTRODUCTION

Using stereo images instead of a single image gives better comprehension of a visible scene. Two images provide the feel of the third dimension and the distance to objects.

Generating a stereo frame takes from 1.5 to 2 times longer than a mono frame, because one must compute two scene images instead of single image. The lower bound comes from the fact, that only one coordinate (x) must be computed twice and the second coordinate (y) is the same on both images. Both images reveal similarities and many visible surfaces are computed twice. Image-based rendering methods make possible to take advantage of this similarities and compute the surfaces once.

This paper describes a method that takes as an input a frame with depth information that was generated for some position of the observer. The method produces two ,,translated” frames for left and right eye using image-based algorithm.

Section 2 describes stereo-vision and rules to create stereo images. Section 3 presents image-based rendering (IBR) technique and description of the previous work in this field. The fast algorithm for creating stereo images using IBR is presented in section 4, and section 5 shows some experimental results and section 6 discusses visible artifacts.

2 STEREO-VISION AND STEREO IMAGING

2.1 Human perception

The third dimension is perceived by humans from two groups of cues, monocular and binocular (see: [Uttal99a]). The first group consists of the following cues: geometrical perspective, surface perspective of textures, shading, focus and relative motion. The second group cues are the differences between two images seen by both eyes. For some visible object the difference between its coordinates is bigger if the object is closer to the observer. For far objects the differences are small and can be smaller than a size of a single pixel, so on both images far objects projections can almost be the same.

That means that the binocular cues apply to the perception of the near objects while the monocular cues to the perception of the further objects. As we will see later, the objects that are very far can have the same position on the left and right images and still from monocular cues the image depth is visible.

2.2 Stereo images

For perception of the scene depth two images are needed, one for the left and one for the right eye. The first image has to be created for camera positioned in the left eye and the second one in the right eye. Figure 1 shows the positions of the cameras and a plane of projection relatively to the visible object. The plane of projection is common for both images.

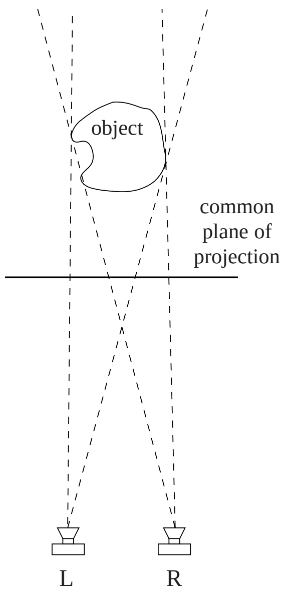

Figure 1: The positions of the cameras and the plane of projection.

Consider three images created for three camera positions: the left eye (L), the right eye (R) and in the middle of the segment connecting eyes (O). Consider also 2D points K_L, K_R and K_O that are the projections of a single 3D point K onto these three images (see figure 2).

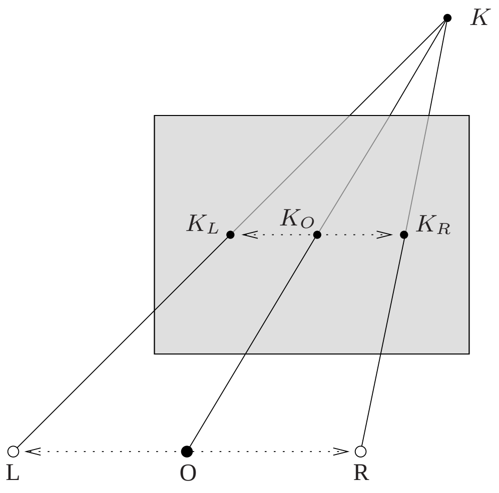

Figure 2: Projections of a single point for three camera positions (see text).

Assume that the distance between cameras L and O and between cameras R and O is $e/2$ (e is the distance between eyes). Assume that point K_O and its depth value z_K are known (notice, that the depth values for that point in all images are equal). We have to compute the coordinates of the points K_L and K_R.

The segment L-R is horizontal, thus K_L-K_R is horizontal also. That means that y-coordinates of points K_L, K_R and K_O are equal. The distance between K_L and K_R is called the stereo parallax of the point K.

Assume that the distance from cameras to the plane of projection is d and that z_K is a depth value of the pixel (the distance of point K from the plane of projection). s_K, the distance between K_L and K_O (case of K_R and K_O is analogical) can be computed from triangle similarity (figure 3). We obtain:

$$s_K = \frac{e}{2} \cdot \frac{z_K}{z_K + d}$$

Since s_K is a length and we need a distance in pixels Δ_K, s_K must be divided by w, the distance between the projections of the centers of adjacent pixels. Thus

$$\Delta_K = \frac{e}{2} \cdot \frac{z_K}{w \cdot (z_K + d)} \tag{1}$$

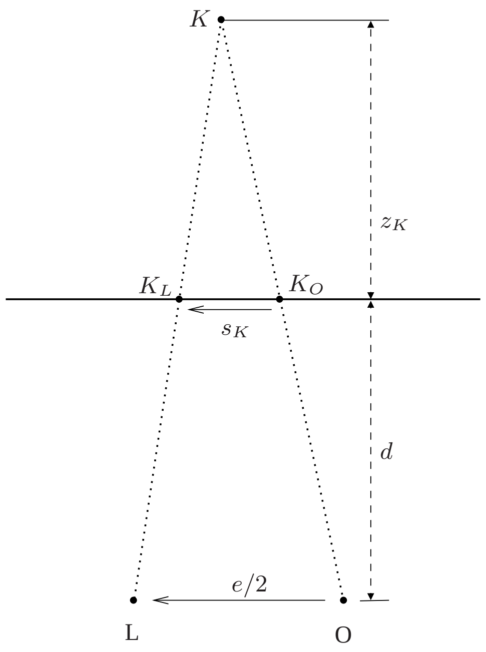

Figure 3: Distance between K_L and K_O.

3 IMAGE-BASED RENDERING

Image-based rendering allows to reduce computational costs in computer animation. The image-based algorithm takes as an input a single or few images (with depth information) and generates new images for other camera positions.

Few image-based methods create intermediate representation of the visible scene. The first piece of research, Light-field and Lumigraph, was developed independently by two groups of researchers, Light-field by Levoy and Hanrahan [Levoy96a] and Lumigraph by Gortler *et al.* [Gortl96a]. Light-field/Lumigraph is a 4-dimensional discrete function representing the light incoming and outgoing from a bounded region. Other methods, developed by Shade et *al.* [Shade98a] and Lischinski [Lisch98a], involve building of the Layered Depth Image structure storing subsequent layers of occluded pixels. Unfortunately the preparation time of intermediate structures is significant, so, for example, these methods cannot be used during animations for rendering acceleration.

Other methods render destination frame directly from source frame or frames. These methods are well described in PhD. dissertations of McMillan [McMil97a] and Mark [Mark99a]. In short, the algorithm consists of two turns. In the first one the pixels coordinates are transformed from source frame to the destination frame. This operation consists of affine transformation of vector $[uz, vz, z]$, where (u, v) are source coordinates of the pixel and z is a depth value of the pixel. In practice, the destination frame is created with one of the following methods: splats [Shade98a] or a mesh. In the first case, each transformed pixel is drawn as a splat on the destination frame, in the second pixels are connected in a mesh and the quadrilaterals of the mesh are filled with the color depending on the neighbour pixels.

The disadvantage of using splats is the possibility of appearance of holes between splats. Fortunately, if the destination pixels are visited in post-processing in a special order, the holes can be filled efficiently (see [Mark99a]).

Few papers address an application of image based rendering in generating stereo images. McMillan and Bishop in [McMil95a] describe a system for head-tracked stereoscopic display. They use image warping for computing arbitrary stereo views from a number of stereo images. The goal of their method is to compute new views sufficiently fast so the delay between head movement and image displaying would not be noticeable. Sawhney *et al.* in [Sawhn01a] present an algorithm for computing high resolution stereo images for films. Two high resolution images for a frame are computed from one high resolution image (for one eye) and one low resolution (for the second eye). The algorithm consists of two steps, computation of a disparity map (at a lower resolution) and then high resolution image warping to the view of the second eye.

McAllister in [McAll98a] uses linear morphing to produce stereo image from a single image with bilateral symmetry such as a human face. The needed second input image is produced by a horizontal reflection. The main point of this method is that a linear morphing between matching features on source images produce the correct parallax in the intermediate images.

4 THE ALGORITHM FOR STEREO IMAGES

In this section I present a method of warping a source image into two images for both eyes. The algorithm consists in two independent image warping operations, first for the left eye and second for the right eye.

As it was stated in section 2.2 the pixels of the source frame are translated only horizontally, thus each row of destination frames can be computed independently.

Consider the transformation of a single row to the left eye camera position. The camera is translated left by the distance $e/2$. Each pixel of this row is also translated left, but from the equation (1), the translation distance is bigger if the value of depth component of that pixel is lesser. That implies that the order of the painting of the points should be from right to left, because in this case it would not be necessary to check z-buffer. The pixels with lesser depth values cover the pixels with greater depth value, as illustrated in figure 4. Covered points are marked with gray circles.

If the distance between successive translated pixels is greater than one pixel, then the area between these pixels is filled with the colour of the second pixel (pixel that is on the left in the source image). Figure 5 explains, why the area should be filled with the colour of the second pixel. The source image does not contain information about surfaces in a gray area with question mark. When the camera is moved to the left, the surface containing the second pixel (further surface) „emerges" from behind the surface containing the first pixel (surface closer to the camera). Thus if it is assumed that there is no other surface in the gray area and the further surface can be extended.

The algorithm for processing a single row of the image is the following:

```
if left eye then
    $\delta_x := -1$
    $x_0 := width - 1$
    $x_1 := 0$
else
    { right eye }
    $\delta_x := 1$
    $x_0 := 0$
    $x_1 := width - 1$
end if
$x'_{\text{prev}} := x_0$
```

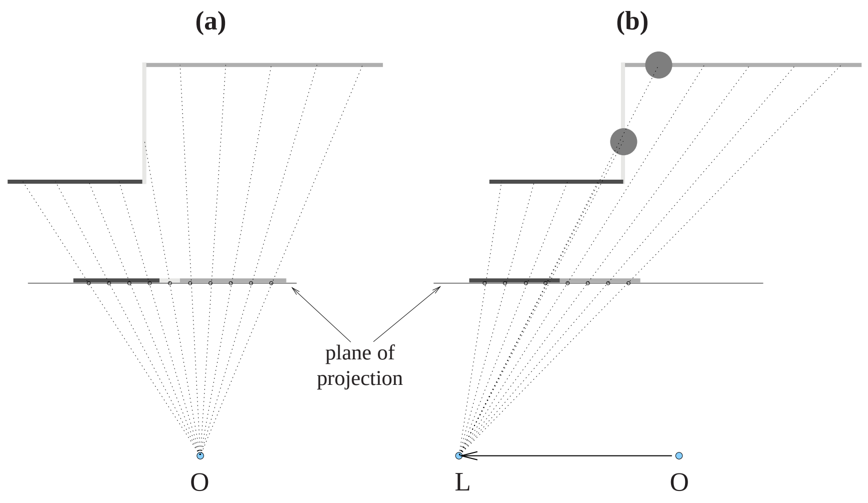

Figure 4: Pixels placements on a source image **(a)** and on an image for the left eye **(b)**. The points that are not visible on a destination image are marked with gray circles.

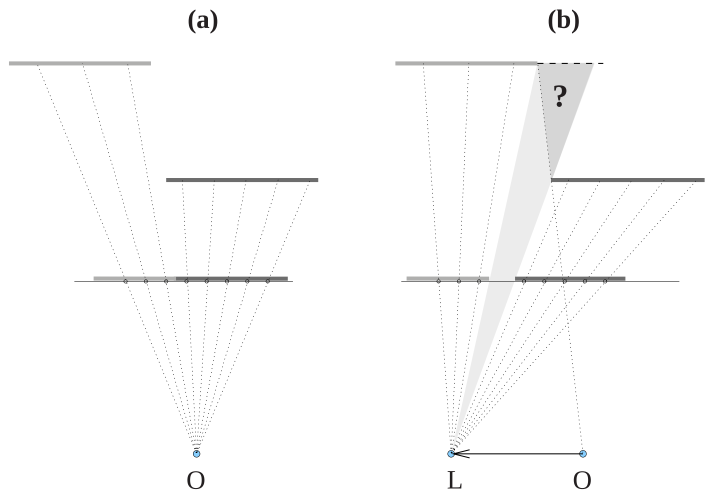

Figure 5: A hole can appear between the pixels of different visible surfaces.

```
for $x := x_0$ to $x_1$ step $\delta_x$ do
    $x' :=$ new coordinate from the equation (1)
    paint the pixel $x'$ with the colour from the source image from pixel $x$
    if $(x' - x'_{\text{prev}}) \cdot \delta_x > 1$ then
        { the area between $x'$ and $x$ is filled in order to remove holes between pixels }
        for $p := x'_{\text{prev}} + \delta_x$ to $x' - \delta_x$ step $\delta_x$ do
            paint the pixel $p$ with the colour from the source image from pixel $x$
        end for
    end if
    $x'_{\text{prev}} := x'$
end for
```

REMARK: The condition $(x' - x'_{\text{prev}}) \cdot \delta_x > 1$ means that the distance between x' and x'_{prev} is greater than one pixel and that, for the left eye image, x'_{prev} is to the right of x'. Thus that area should be filled according to the rule described earlier.

5 EXPERIMENTAL RESULTS

In this section I show some experimental results of the presented algorithm. Each figure consists of two images for left and right eye and can be viewed by looking in parallel on it.

Figure 6 shows a scene with a sailing boat. The source frame was generated using OpenGL. The surface of the water was textured for better visibility of the scene depth. The scene is pretty simple and the OpenGL rendering was accelerated by hardware, thus the warping algorithm worked longer than OpenGL rendering. But if the scene had been more complex or if the warping algorithm had been implemented in the hardware it would have been much faster.

Figure 7 presents a ball laying on a checker floor. The source frame was generated with a ray-tracing method by a modified version of the Rayshade that was able to save the z-buffer to the external file. Both the ball and the floor are reflective. In the image there are three areas of light reflection: reflection of the ball in the floor, reflection of the checker in the ball and reflection of a light source in the ball. In reality, the reflections on both images should be placed differently. But the ball has big curvature and the distance between eyes is small. Thus the reflection positions on the ball on both images are considerably close to each other. The reflection on the floor is incorrect. The reflection is transformed together with the floor pixels and it looks as the reflection was painted on the floor.

Unfortunately it is not possible to take into account a proper computation of reflection position because image-based rendering methods can transform the pixels from the source frame. These pixels do not contain information about parts of the colour that comes from reflections and directly from the surface.

Figure 8 shows cylinders partially occluded by themselves. The source frame was generated with the ray-tracing method. There are visible artifacts behind the nearest cylinder consisting in that the pixels of the occluded floor and further cylinders are stretched horizontally. These artifacts are caused by the fact that there is no information in the source frame about the non-visible area behind that cylinder. These artifacts are visible because the texturing of that area differs from the texturing of the floor or of the cylinders (if the colour of these objects had been more uniform the artifacts would have been harder to notice). These artifacts does not disturb the observation of the depth of the scene.

The approximate time of an execution of the algorithm for image from the figure 8 is the following (CPU: Intel Celeron 450MHz, resolution of images: 320×200): source ray-traced frame 4.5s, both warped frames 0.35s. The time of the ray-traycing depends directly on the complexity of the scene and on the contrary the time of the warping algorithm is independent of the scene complexity. Thus in the case of more complex scenes the efficiency gain is bigger.

6 ARTIFACTS

This section describes proposed methods of getting along with the following artifacts: texture artifacts in “holes” between pixels and specular reflections.

Hole filling method in the presented algorithm extends the surface that is farther from observer, but unfortunately this introduce artifacts on the texture. If source frame was computed with ray-tracing it is possible to fill the holes by recomputing their pixels with ray-tracing, but such algorithm would be slower than the algorithm described above. The possible solution to this problem is to use two pass algorithm instead of a single pass one. The first pass is similar to the whole described algorithm, but the space between the pixels is not filled. In the second pass these pixels are filled with the color blended from the left or right neighbours of that pixel, similarly as described by Mark in [Mark99a]. This modification may only slightly elongate the processing time.

There are two problems with the specular reflections. The first one is the following. When the camera moves the reflection moves on the object surface. If the input of the algorithm is only an image with depth information it is not possible to determine which components of the pixel colour come from direct light and which from reflections. Fortunately, if the source image was rendered with ray-tracing it might be possible to change the ray-tracing algorithm to gather independently the direct and reflected light.

The second problem is to compute the proper reflection placement. In the case of planar reflections one may notice that the reflection move also horizontally like the surface directly visible. The distance of this translation depends on the sum of the distances between the observer and the reflecting surface and between the reflecting and reflected surfaces. Unfortunately the reflections from non-planar surfaces cannot be computed in this way, because they can move also vertically. This problem should be solved in the future.

7 CONCLUSION

This paper presented a fast 3D image warping algorithm for stereoscopic viewing. High efficiency comes from the fact that the camera position is allowed to change only horizontally relative to the image and that the plane of projection is invariable. Thus the algorithm can be a simplified version of the warping algorithm that was previously described in the literature.

If we are not considering the pixels of the filled holes, the resulting images contain objects with correct geometry visible in perspective view (after a camera translation). Similarly like in other IBR methods the holes may not be filled correctly, because there is no information in the source image about invisible surfaces.

Because the process of generating of the stereo images involves a computation of a source frame with the original algorithm the time gain is at most 50% of the original computation minus the time of processing of presented algorithm. In the case of ray-tracing the gain is just almost 50%.

The described method can be applied in order to accelerate generating of the stereo animations, if the time of generation of the single animation frame is big (at least twice as the time of the presented algorithm). It is especially easy to apply this algorithm as a post-processing to ray-tracing or OpenGL rendering, because these methods generate the z-buffer by themselves.

The advantages of the presented method are the following: simplicity and speed that do not depend on the scene complexity.

All camera parameters need not to be known. It is sufficient to know only the resolution of images, horizontal angle of view and the distance from the camera to the plane of projection.

It is easy to parallelise the algorithm since each row of the image can be computed independently.

REFERENCES

[Gortl96a] Gortler, S.J., Grzeszczuk, R., Szeliski, R. and Cohen, M.F.: The Lumigraph, *SIGGRAPH 96 Conference Proceedings*, pp. 43–54, 1996

[Levoy96a] Levoy, M. and Hanrahan, P.: Light field rendering, *SIGGRAPH 96 Conference Proceedings*, pp. 31–42, 1996

[Lisch98a] Lischinski, D. and Rappoport, A.: Image-based rendering for non-diffuse syntetic scenes, *Proceedings of Eurographics Rendering Workshop '98*, pp. 301–314, 1998

[Mark99a] Mark, W.R.: Post-Rendering 3D Warping: Visibility, Reconstruction and Performance for Depth-Image Warping, *Ph.D. thesis*, 1999

[McAll98a] McAllister, D.F.: Stereo Pairs From Linear Morphing, *Stereoscopic Displays and Virtual Reality Systems IX, IS&T/SPIE Electronic Imaging '98 Proceedings*, 1998

[McMil95a] McMillan, L. and Bishop, G.: Head-tracked stereoscopic display using image warping, *Stereoscopic Displays and Virtual Reality Systems II, SPIE Proceedings 2409*, 1995

[McMil97a] McMillan, L.: An Image-Based Approach to 3D Computer Graphics, *Ph.D. thesis*, 1997

[Sawhn01a] Sawhney, H.S., Guo, J., Hanna, K., Kumar, R., Adkins, S. and Zhou, S.: Hybrid Stereo Camera: An IBR Approach for Synthesis of Very High Resolution Stereoscopic Image Sequences, *SIGGRAPH 2001 Conference Proceedings*, 2001

[Shade98a] Shade, J.W., Gortler, S.J., He, L. and Szeliski, R.: Layered depth images, *SIGGRAPH 98 Conference Proceedings*, pp. 231–242, 1998

[Uttal99a] Uttal, W.M., Kakarala, R., Dayanand, S., Shephers, T., Kalki, J., Lunkis, C.F. and Liu, N.: Computational Modeling of Vision, *Marcel Dekker, Inc*, 1999

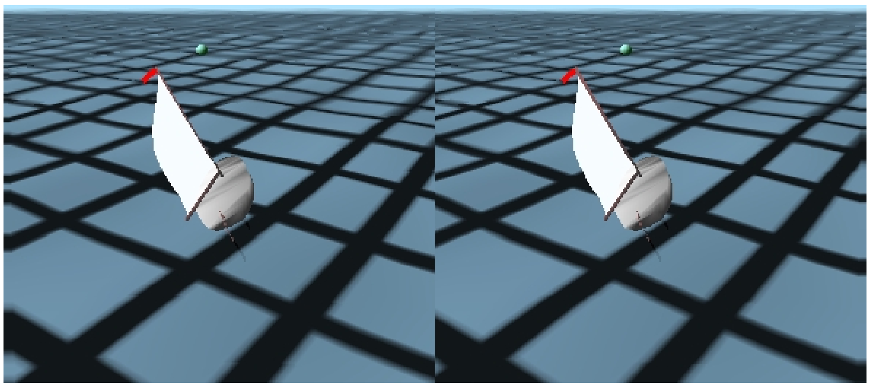

Figure 6: Sailing boat.

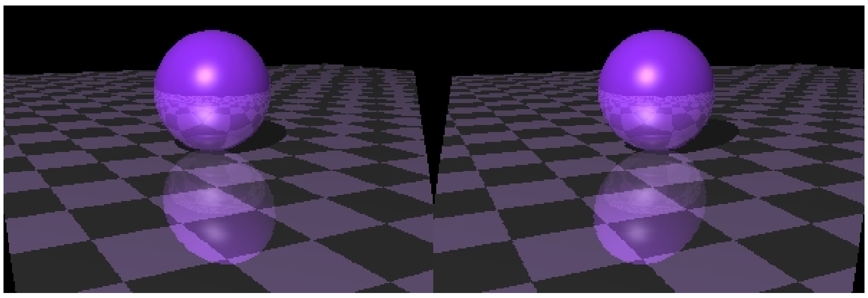

Figure 7: Ball on a surface.

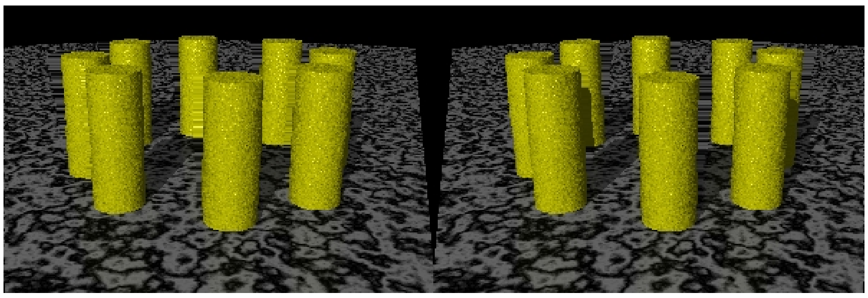

Figure 8: Cylinders partially occluded by themselves.