Image Flow in Light Fields

Jarno van der Linden

Graphics and Visualisation Group, University of Auckland

Abstract

Image-based modeling can simplify the scene modeling process in computer graphics by assuming that it is not the geometry of a scene that is of interest, but rather the look of a scene. Instead of obtaining detailed surface geometry and colour data, visual data alone is acquired. This can be done with relative ease through photography. Light Fields are a recent development in how such information is stored. Images of a scene are used to create a four dimensional radiance function. A light field determines what a scene looks like when viewed from anywhere outside of the space enclosing the scene.

With light fields, as in many other image-based modeling and rendering systems, there is a trade-off between quality and memory. A high quality result requires storage of a large number of samples obtained from high-resolution input images. The number of images required can be reduced by using image warping techniques based on image flow. Image warping computes intermediate images from the existing images in the system, thereby giving the illusion of a higher sampling density than that actually used during acquisition. In this paper, we show how to use image flow in light fields to produce high quality output from only a small set of input images. Unlike previous work, we do not generate intermediate images, but rather use image flow to render a new view directly.

High quality new views can be generated from as few as 8×8 images. We demonstrate several rendering methods for image-flow augmented light fields.

1 Introduction

Traditional scene modeling using geometric primitives is a long and expensive process. To capture the fine detail of the geometry of a scene, millions of polygons may need to be created. On top of this, surface properties such as texture and reflection need to be determined.

Image-based modeling promises to greatly simplify the modeling of an existing scene by assuming that it is not the *geometry* of a scene that is of interest, but rather the *look* of a scene. The way a scene looks is readily captured in high detail by photographic means. The aim of image-based rendering is then to reconstruct an arbitrary view of the scene based on a set of input images. Such a set of input images constitutes a model of the scene. How this model is stored and used has been investigated in various rendering systems.

In the QuickTime VR system [1], a cylindrical panorama is stored to create outward-looking views of a scene. To create a new view, a small region of the panorama is extracted and warped from a cylindrical image to a planar view. The result is a view of the scene in which the camera can be zoomed and freely rotated about a fixed point. For rotation around an object, QuickTime VR uses a different method. Here the model consists of a set of images of the object, one for each rotation of the object. A view is rendered by selecting the image which best matches the requested viewing direction. In this way the user can arbitrarily rotate the object around a fixed rotation point.

Plenoptic modeling [7] overcomes the fixed point restriction of QuickTime VR by storing cylindrical projections from various points in the scene, and predicting how the view of the scene changes when moving between cylinders. McMillan and Bishop define the goal of image-based rendering to be the generation of a continuous representation of a five-dimensional *plenoptic function*. The plenoptic function defines what can be seen from any point in space in any direction.

Light fields (or Lumigraphs) [6] [3] simplify the plenoptic function to four dimensions. A light field model of a scene is stored as a radiance function over the space of directed lines through a volume of space containing the scene. Because only those lines that intersect the volume are of interest, only four dimensions are needed rather than five. The price for this simplification is that the camera position is restricted to lie outside the scene volume.

Being image based, light fields have several advantages over traditional geometry based models:

- Models of existing real objects or scenes are easily acquired in high detail by photographic means
- Photorealistic views are easily rendered
- Rendering times are independent on scene complexity
- Storage complexity is dependent not on scene complexity, but on image complexity
- Rendering times are independent on scene complexity, making light fields useful for displaying views of synthetic scenes as well

One disadvantage of light field models is that they require a large number of images to accommodate viewing from a large range of angles and positions. A light field can range from hundreds to thousands of megabytes of total storage. Another disadvantage is that the result of reconstructing a view is often disappointing. Images show obvious sampling and depth of field artifacts, unless the sampling rate is very high or the light field is filtered with a low-pass filter to remove high frequency data.

Here we propose to include depth information in light fields, in the form of image flow. This approach solves both problems by reducing, even eliminating, the need for pre-filtering the light field, and generating high quality output even with a coarse sampling rate. Although our scheme requires image flow information to be computed, the advantages to using image flow, particularly image quality, outweigh the extra effort.

Our scheme is particularly suited to computer generated data, making it an attractive rendering method for e.g., real-time walk-throughs of complex scenes, or the fast compositing of computer models with real scenes.

After reviewing previous related work, we discuss the meaning of image flow in light fields. In section 4 we show how image flow can be used to generate new views through two different methods. In section 5 we demonstrate and compare our implementation of these methods. This is followed by our conclusion and ideas on future work.

2 Related work

Image flow (also called optical flow or stereo disparity) has been closely associated with image-based rendering and computer vision. The use of optical flow for interpolating between two or more views of a scene is typified by the work of Chen and Williams [2]. After determining pixel correspondences between a pair of images, the resulting disparity vectors are linearly interpolated to translate the pixels from the image pair to a new in-between view. Overlaps can be handled through the use of a Z-buffer algorithm, using the length of the disparity vector as a measure of depth. Holes which would otherwise appear in the final image are reduced or eliminated by considering the pixels of the source images as squares rather than point samples. Alternatively, the holes can be filled in with a colour interpolated from the nearest filled pixels of the final image.

Park, Lee, and Shin [9] generate an optical flow set from a set of light field images, by chaining together runs of matching pixels. Each element in the set contains information about a representative pixel's position, colour, disparity, and origin image, and the range of images over which the element can contribute to a valid result. Reconstruction is done by applying optical flow to those elements in the optical flow set whose valid image range contains the image to be reconstructed. The amount of flow to apply is dependent on the distance between the element's origin image and the new image. Chaining is applied in only one direction, either horizontal or vertical. Our approach on the other hand employs optical flow both in the horizontal and vertical directions at the same time.

Recently, Heidrich et al. [4] have shown how to use traditional image warping methods such as in [2] to increase the density of images in the light field at load time. Rendering new views is still achieved by means of bilinear or quadralinear interpolation methods. Like Heidrich et al. we employ the depth buffer of synthetically rendered scenes to create the warps. Our work however differs significantly in several areas. Most importantly, we use image flow to render high quality arbitrary new views directly from sparsely sampled light fields. This means a significant saving not only in acquisition time and disk storage, but also in memory requirements. Secondly, we do reconstruction using a splatting-like method, rather than bilinear or quadralinear interpolation.

3 Image flow in light fields

Image flow is typically defined as the apparent shift on the image plane of the projection of a point in space when viewed from different camera positions or orientations. In order to use image flow in light fields, the meaning of image flow in terms of light fields must first be defined.

3.1 Parameterisation of the light field

The light field is parameterised by a light slab or slice. Directed lines are specified by their point of intersection with two known parallel planes (Figure 1). Six such slabs can be used to completely surround the scene. Alternatively, they can surround the camera to create outward looking as opposed to inward looking views.

When acquiring a light field model, the UV plane forms the camera plane, and the ST plane the image plane. By positioning the camera at various (u, v) coordinates and imaging the scene through the ST plane, a set of (u, v, s, t) samples is obtained to form an image-based model of the scene. To create a new view of the scene from an arbitrary camera position, the (u, v, s, t) coordinates of the rays through the camera and its image plane are computed. The radiance along a ray is then determined by sampling the light field at those coordinates (Figure 2).

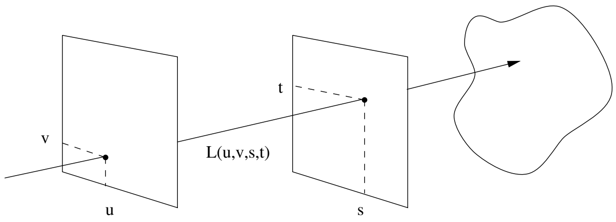

Figure 1: Light slab representation of a light field

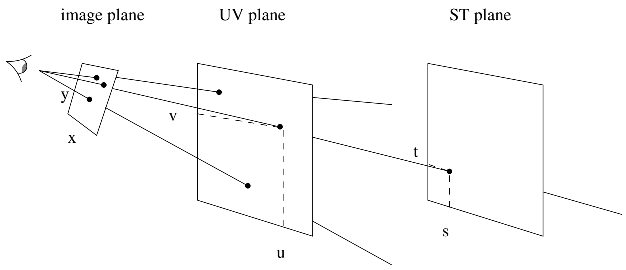

Figure 2: Creating a new view of a scene using a light slab. The colour of a pixel in the camera's image plane is given by the associated ray's intersection with the UV and ST planes

3.2 Image flow in light fields

We define image flow in the context of light fields as follows. Consider a point P in the scene. Let L be the set of all directed lines (u, v, s, t)[1] that intersect P. As the UV and ST planes are parallel, there is a linear relationship between $\underset{\sim}{u} = (u, v)$ and $\underset{\sim}{s} = (s, t)$:

$$\underset{\sim}{s} = f \times (\underset{\sim}{u} - \underset{\sim}{u}_0) \tag{1}$$

where $\underset{\sim}{u}_0$ is the (u, v) coordinate such that the line through $\underset{\sim}{u}_0$ and P intersects the ST plane at $\underset{\sim}{s} = (0, 0)$. The scaling factor f is taken as the definition of the *flow* of P. As we deal exclusively with opaque objects, each directed line in the light field is associated with at most one point P. We can therefore say that the directed line (u, v, s, t) has flow f, where f is the flow of point P seen along the directed line.

Intuitively, the flow field determines how the points in a scene imaged on the ST plane appear to move as a camera is shifted on the UV plane by one unit. It is described by a scalar field, not a vector field. The direction of camera movement, which is known, determines the direction of flow. Hence only a scalar flow value needs to be stored.

Contrary to the image flow obtained from stereo vision, the flow associated with a point P increases with its distance from the ST plane, and hence the camera. If P is on the ST plane the flow is zero as the point is coincident with its projection on ST. Let d be the distance of P from the UV plane. Then the flow f of P is related to the distance d by (Figure 3):

$$f = \frac{d-1}{d} = 1 - \frac{1}{d} \tag{2}$$

Note that as d has a value greater or equal to one, the flow value is

[1]Note that in this paper we assume a normalized coordinate system

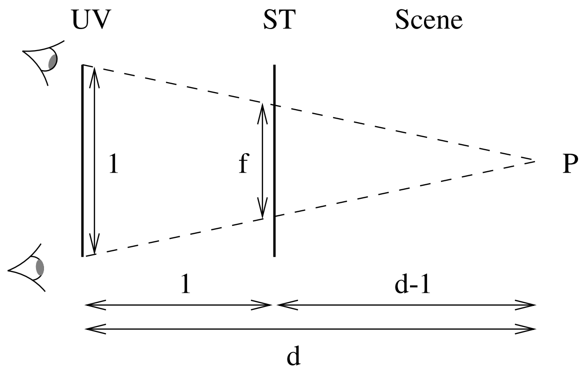

Figure 3: The relation between flow and depth of a point P in the scene

in the range

$$0 \le f < 1 \tag{3}$$

Given a point P in a scene seen in the light field along the directed line (u, v, s, t) with flow f, it is also seen (barring occlusion) along all directed lines (u', v', s', t') which intersect at P. From the definition of image flow in light fields given by equation 1 we obtain:

$$\begin{aligned} \underset{\sim}{s}' - \underset{\sim}{s} &= f \times (\underset{\sim}{u}' - \underset{\sim}{u}_0) - f \times (\underset{\sim}{u} - \underset{\sim}{u}_0) \\ &= f \times (\underset{\sim}{u}' - \underset{\sim}{u}) \end{aligned}$$

Solving for $\underset{\sim}{s}'$:

$$\underset{\sim}{s}' = \underset{\sim}{s} + f \times (\underset{\sim}{u}' - \underset{\sim}{u}) \tag{4}$$

Equation 4 relates two light field samples to the same 3D point in the scene, and where a projection of a point seen in one view is located in another view. It forms the heart of our system.

4 Rendering using image flow

Image flow can be used to improve the quality of rendered output of a light field. Two methods are discussed: one which we call flow forward rendering, and one called scan backward rendering.

4.1 Image flow for interpolation

A light field is acquired by imaging the scene through the ST plane with a camera at grid points on the UV plane. Each pixel in the images corresponds with a (u, v, s, t) sample. The camera position determines the (u, v) coordinates, while the (s, t) coordinates are given by the pixel positions in the image. To reduce memory requirements while maintaining image quality, and because positioning the camera and taking an image takes time, the UV plane is sampled at a much coarser level than the ST plane. Both Levoy and Hanrahan [6] and Gortler et al. [3] demonstrate light fields where the UV sampling is an order of magnitude less than the ST sampling (e.g. 32×32 for UV versus 256×256 for ST).

The radiance function of a light field using such a sparse set of (u, v) coordinates can be reconstructed by various interpolation methods [10]. For example, the (u, v) coordinates can be simply rounded to the nearest sampled coordinate. Another approach is to do a bilinear interpolation on UV, or even a quadralinear interpolation on $UVST$. However, the results of these approaches are not satisfactory, especially when a new synthetic view encompasses a large range of (u, v) coordinates. The final image contains noticeable discontinuities, or is excessively blurred. The effect of blurring due to colour interpolation can be reduced significantly by using the adaptive focusing method described in [10] and [5]. With image flow, a different approach can be taken. Instead of interpolating the colours, samples are warped to new positions to form the new view. Two approaches are used: scan backward, which finds a nearby sample that flows to the desired position, and flow forward, which warps all nearby samples to new positions and resamples the result at the desired positions.

4.2 Scan backward

Given a desired directed line (u', v', s', t') the aim is to find a known sample (u, v, s, t) with flow f such that it flows to the desired $\underset{\sim}{s}' = (s', t')$ coordinate due to the camera translation from $\underset{\sim}{u} = (u, v)$ to $\underset{\sim}{u}' = (u', v')$. To reduce the complexity of the search, only the nearest known $\underset{\sim}{u}$ coordinates to $\underset{\sim}{u}'$ are considered. This restriction is based on the observation that the image formed by placing the camera at $\underset{\sim}{u}$ is very similar to the image formed by displacing the camera a small amount to $\underset{\sim}{u} + \Delta\underset{\sim}{u} = \underset{\sim}{u}'$.

The search is limited to a line of $\underset{\sim}{s} = (s, t)$ coordinates which satisfy a rearrangement of equation 4:

$$\begin{aligned} \underset{\sim}{s} &= \underset{\sim}{s}' - f \times (\underset{\sim}{u}' - \underset{\sim}{u}) \\ &= \underset{\sim}{s}' - f \times \Delta\underset{\sim}{u} \end{aligned} \tag{5}$$

The search can be further constrained to a line segment by taking into account the limits of f given by equation 3.

For each directed line (u', v', s', t') in the view the scan backward method attempts to find those samples which satisfy equation 5. There may be multiple samples which solve the equation. Some samples may become occluded by other samples. The desired sample will be the one nearest the camera, i.e. the sample with the smallest flow value. From equation 5 we can see that the sample with smallest f must necessarily have $\underset{\sim}{s}$ nearest $\underset{\sim}{s}'$.

In practice, equation 5 will almost never be satisfied exactly due to the fact that the light field is sampled, rather than continuous. In section 5.2 we show how to implement the backwards scanning idea in a discretely sampled light field.

4.3 Flow forward

Given the set of all known samples, we can construct a new view by applying image flow. The projection $\underset{\sim}{s}' = (s', t')$ onto the ST plane of a sample (u, v, s, t) with flow f as seen along a line through $\underset{\sim}{u}' = (u', v')$ is given by equation 4. By applying the equation to all the known samples in the light field, a new view is generated. We are taking all the known samples and flowing them towards the new view. In practice this is too time consuming for a real-time display as tens of millions of samples are involved, so restrictions have to be placed on which samples are considered.

Assume for a start that the new view is from a camera positioned on the UV plane. All rays emanating from the camera share the same UV coordinate $\underset{\sim}{u}'$. As with the scan backwards method, of the known samples in the light field only those with a UV coordinate nearest $\underset{\sim}{u}'$ are considered. The ST coordinate $\underset{\sim}{s}'$ of each such sample is now displaced by their flow according to equation 4.

The image formed in this way is a correct view of the scene from a camera at $\underset{\sim}{u}'$.

Now consider the camera positioned off the UV plane. The $\underset{\sim}{u}'$ coordinate is no longer the same for each ray. Instead, it varies as a function of $\underset{\sim}{s}'$. As discussed in section 3.2, there is a linear relationship between $\underset{\sim}{u}'$ and $\underset{\sim}{s}'$:

$$\underset{\sim}{u}' \quad = \quad \underset{\sim}{u}_0 + r \times \underset{\sim}{s}' \tag{6}$$

where r plays a similar role to f. In fact, equation 6 is equation 1 rewritten from the point of view of P positioned before the UV plane instead of after the ST plane. Here, $\underset{\sim}{u}_0$ is the point of intersection on the UV plane by a ray through the camera and the coordinate $(0, 0)$ on the ST plane. Substituting into the flow equation 4, we find that $\underset{\sim}{s}'$ is given by:

$$\begin{aligned} \underset{\sim}{s}' &= \underset{\sim}{s} + f \times ((\underset{\sim}{u}_0 + r \times \underset{\sim}{s}') - \underset{\sim}{u}) \\ &= \frac{1}{(1 - f \times r)} \times (\underset{\sim}{s} + f \times (\underset{\sim}{u}_0 - \underset{\sim}{u})) \end{aligned} \tag{7}$$

What values should be used for $\underset{\sim}{u}$ and $\underset{\sim}{s}$? In the restricted case presented earlier, we chose to use all $\underset{\sim}{s}$ values, with $\underset{\sim}{u}$ fixed based on $\underset{\sim}{u}'$. In the general case here however, this can not be done. $\underset{\sim}{u}'$ is dependent on where on the ST plane ($\underset{\sim}{s}'$) the sample is moved to (equation 6), which in turn is determined by $\underset{\sim}{u}' - \underset{\sim}{u}$ and the sample's flow f (equation 4). We break this cyclic dependency by choosing $\underset{\sim}{u}$ for each $\underset{\sim}{s}$ such that (u, v, s, t) is closely oriented to the camera's lines of sight. The idea is that such samples require only a small amount of displacement to bring them into line with the view, thus reducing warping effects such as holes [2]. The camera's lines of sight satisfy the same relation as equation 6, hence for a given $\underset{\sim}{s}$ we take $\underset{\sim}{u}$ to be:

$$\underset{\sim}{u} \approx \underset{\sim}{u}_0 + r \times \underset{\sim}{s} \tag{8}$$

Creating a new view using the flow forward method involves determining a $\underset{\sim}{u}$ for each coordinate $\underset{\sim}{s}$ in the light field, retrieving the flow and colour of sample (u, v, s, t), and combining these with the view parameter r in equation 7 to obtain the sample's new position $\underset{\sim}{s}'$ on the ST plane. The image thus formed on the ST plane can then be projected on the new view's image plane.

As with the scan backwards method, multiple samples may flow to the same screen coordinate due to occlusion. A proper ordering must be defined to ensure that only the sample representing the point nearest the camera is visible. This sample will have the smallest flow value among the colliding samples, as flow increases with depth (equation 2).

Note that unlike the scan backwards method, the flow forward algorithm can not be used to quickly resample the light field at a given point. This is not a problem for normal light field viewing which involves the reconstruction of an entire image.

5 Implementation

We have implemented the image flow methodology for light fields in our light field generator and rendering system. The system can generate synthetic light fields by rendering different views of a model using OpenGL. For a range of UV coordinates, a skewed perspective view of the scene is rendered [10]. Every pixel in the image $I(u, v)$ then corresponds with an ST coordinate on a regular grid. The flow values are computed from a depth map. Hence we obtain a sampling of the light field at a range of regularly spaced (u, v, s, t) coordinates.

To demonstrate and compare the various methods, a light field was synthetically generated using a model of a tricycle. The light field is of size 8×8 in UV and 512×512 in ST. For reference, figure 4 shows a perspective view of the model rendered using OpenGL. Figure 5 shows the same view rendered from the light field with the camera off the UV plane using only quadralinear colour interpolation. The light field is not filtered for high frequency data ([6]) to emphasise the high degree of filtering, and hence loss of detail, needed if colour interpolation is used on light fields with low sampling rates. Our image flow techniques require no such filtering.

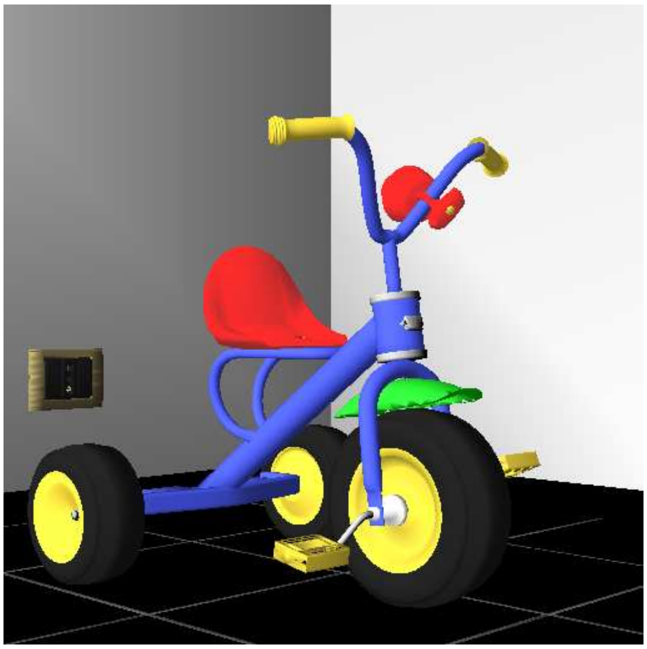

Figure 4: OpenGL rendering of the tricycle scene.

5.1 Reconstruction

Because the light field is sampled only at discrete points, we must take care with reconstruction [8] in order to avoid "tearing" (see appendix A). Reconstruction is done with a Gaussian distribution of the sample's colour centred on $\underset{\sim}{s}'$, the sample position on the ST plane after applying image flow. The colour C of the sample contributes to the colour C'' of a nearby pixel $\underset{\sim}{s}''$ according to the distance between them:

$$\begin{aligned} C''_{new} &= (1 - h) \times C''_{old} + h \times C \\ h &= e^{-\frac{1}{2} \times \left(\frac{l}{\theta}\right)^2} \\ l &= \|\underset{\sim}{s}'' - \underset{\sim}{s}'\| \end{aligned} \tag{9}$$

We use a value of 0.4 for θ. This blending requires that points in the scene near the camera are rendered after points further away. By storing all the flowed samples in a list sorted according to flow value (i.e. depth order), and then rendering the sorted list of points, the proper order is guaranteed. The flow forward method uses this

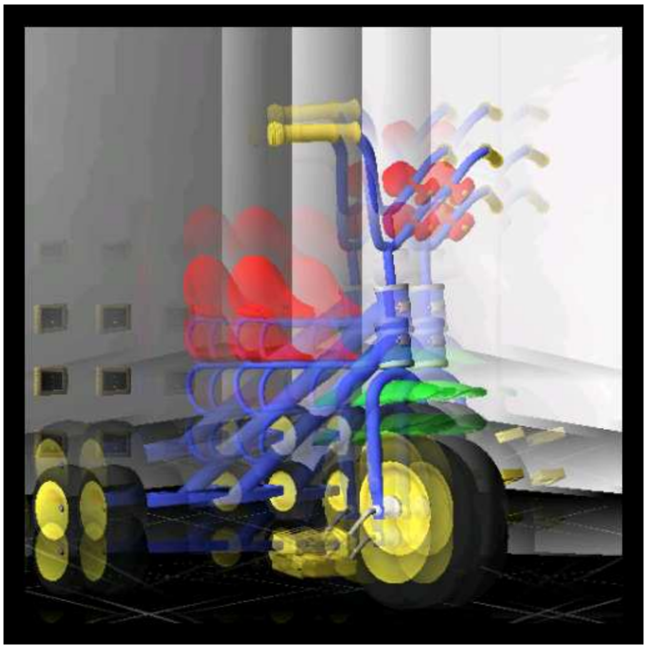

Figure 5: Light field rendered using quadralinear colour interpolation.

method. The scan backward method uses the ordering implicit in the search direction, as discussed in section 4.2.

For the flow forward renderer, the actual rendering and blending is implemented by drawing alpha-blended rectangular polygons with OpenGL. The rectangles are texture-mapped with a texture whose alpha values follow a 2D Gaussian distribution centred on the texture. The scan backward rendering is done to a bitmap which is then texture-mapped on the ST plane.

There remains the question of what size the footprint of the Gaussian should be, i.e. the size of the polygon drawn for each point. Consider one sample. It is taken to be a representative of a range of ST values. Let the grid spacing in ST be k_{ST}. A reconstruction of the sample with no image flow applied should have a footprint of size proportional to k_{ST}, $c \times k_{ST}$ say. Applying image flow has the effect of distorting this to:

$$k'_{ST} = c \times k_{ST} \times \frac{1}{1 - f \times r} \qquad (10)$$

For the tricycle scene sampled at 512×512 in ST we use $c = 2$.

5.2 Scan backward

In practice, equation 5 will almost never be satisfied exactly due to the fact that the implementation of a light field uses discrete samples, rather than a continuous field. In view of the reconstruction issue discussed above, we must relax the equality condition of equation 5. Instead of a yes/no selection of a single sample, each sample located on or near the line segment searched contributes a certain amount to the colour of the desired light field sample P. The contribution is determined by the Gaussian reconstruction specified in equation 9.

In order to have the samples contribute in the correct order of near points on top of far points, we step along the segment to be searched in the direction towards the desired ST coordinate. Figure 6 demonstrates the result of this method. The effects of limiting the search to the nearest UV coordinate are clearly seen as holes. To determine if a hole is formed, along with the colour for sample P a density value is computed. If this density is less than a threshold value, we consider it a hole. On detecting a hole, the search is repeated using other nearby UV coordinates until a density is found at least equal to the threshold. Figure 7 shows the result of using this filling technique. We limited the search to the nearest 16 UV neighbours, and set the threshold at 0.99.

5.3 Flow forward

For all integer (s, t) grid positions on the ST plane, the UV plane intersection coordinates of the rays from the camera to (s, t) are computed by rounding the right hand side of equation 8 to the nearest UV grid position. The resulting (u, v, s, t) coordinate is then used to index into image $I(u, v)$ and extract the colour and image flow value at pixel location (s, t). Combined with the view parameters, these are used in equation 7 to compute a new position $\underset{\sim}{s}'$ on the ST plane. The Gaussian reconstruction kernel is then drawn on the ST plane centred at $\underset{\sim}{s}'$.

Our implementation of this method does not attempt to fill holes. We have found them to be a rare occurrence. Holes could be filled by considering a larger range of UV coordinates, at the expense of speed. Alternatively, the scan backward algorithm may be used to fill any holes.

Figure 8 shows an example output of the synthetically generated tricycle light field. The image requires about 1.9 seconds to render on a 400MHz Pentium-II class machine. Approximately half this time is spent constructing and rendering the polygons. Reducing the ST resolution to 256×256 causes these rendering times to drop by a factor of 4. A similar speedup can be achieved by using only a subset of the samples for warping and drawing. When high frame rates are desired, for example while repositioning the camera, image quality can be traded off for frame rate. Figure 9 show the result of only using every third column of every third row of the ST grid when generating a new view, i.e. a total of 29241 samples out of 262144 are used. The reconstruction kernel size is scaled up accordingly to cover the gaps left by the missing samples. The rendering time is 420 ms, of which about two thirds are taken by the polygon construction and rendering phase.

6 Conclusion

We have shown how to use image flow information derived from depth maps to dramatically improve image quality in light field rendering. Unlike previous work, image flow is not used to increase the resolution of the light field, but directly in the rendering stage to create arbitrary new views. This leads to significant savings in time generating a light field, disk space, and memory requirements.

By employing a Gaussian reconstruction kernel we obtain excellent results free of ghosting effects inherent in interpolation-based methods used previously. Rendering can be done at useable frame rates on widely available consumer hardware.

We have implemented two methods of using image flow: the scan backward searching method which tries to determine what sample would flow to a given light field coordinate, and the flow forward method which applies image flow from nearby coordinates to construct a new view. Of these two methods, flow forward results in slightly better results at much faster frame rates. Furthermore, our experience shows that no high frequency filtering is required on the light field, unlike methods based on bilinear of quadralinear interpolation.

Our implementation does not take into account changes in colour as a result of camera movement. Scenes with significant specular components therefore cause discontinuities in colour when rendered from a new viewpoint. Work remains to be done to combine

Figure 6: Result of the scan backward method.

Figure 7: Result of the scan backward method with hole filling.

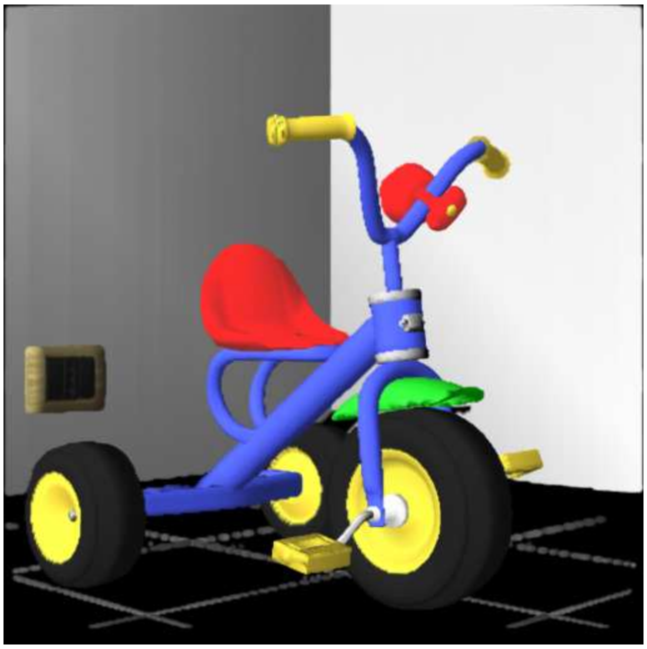

Figure 8: Result of the flow forward method.

Figure 9: Flow forward, using $\frac{1}{9}$th of the samples.

image flow interpolation with colour interpolation in order to render such scenes better.

The rendering times of our system are significantly affected by bus speed and polygon count limitations of the graphics hardware used. Reducing the number of samples that need to be used to create an image of a certain quality is important.

7 Acknowledgments

The tricycle model was made by Adriano Del Fabbro.

References

[1] S.E. Chen. Quicktime VR — an image-based approach to virtual environment navigation. In *Computer Graphics Proceedings*, Annual Conference Proceedings, pages 29–38. ACM SIGGRAPH, 1995.

[2] S.E. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics Proceedings*, Annual Conference Proceedings, pages 279–288. ACM SIGGRAPH, 1993.

[3] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The Lumigraph. In *Computer Graphics Proceedings*, Annual Conference Proceedings, pages 43–54. ACM SIGGRAPH, 1996.

[4] W. Heidrich, H. Schirmacher, H. Kück, and H.-P. Seidel. A warping-based refinement of lumigraphs. Technical Report 20, Universit"at Erlangen-N"urnberg, 1998. accepted for publication at WSCG '99.

[5] A. Isaksen, L. McMillan, and S.J. Gortler. Dynamically reparameterized light fields. Technical report, MIT LCS Computer Graphics Group, May 1999.

[6] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics Proceedings*, Annual Conference Proceedings, pages 31–42. ACM SIGGRAPH, 1996.

[7] L. McMillan and G. Bishop. Plenoptic modelling: An image-based rendering system. In *Computer Graphics Proceedings*, Annual Conference Proceedings, pages 39–46. ACM SIGGRAPH, 1995.

[8] D.P. Mitchell and A.N. Netravali. Reconstruction filters in computer graphics. In *Computer Graphics Proceedings*, Annual Conference Proceedings, pages 221–228. ACM SIGGRAPH, August 1988.

[9] T.-J. Park, S. Lee, and S.Y. Shin. Optical flow rendering. In N. Ferreira and M. Göbel, editors, *Eurographics '98*, volume 17, pages C75–C85, 1998.

[10] J. van der Linden. Auto-focusing of light fields. In *Proceedings of the Twenty Second Australasian Computer Science Conference*, pages 146–157, January 1999.

Appendix

A Tearing

As well as holes caused by occlusion, images rendered from light fields also suffer from a tearing effect. This effect is demonstrated by considering two samples $P_1 = (u, v, s_1, t_1)$ and $P_2 = (u, v, s_2, t_2)$, both with equal flow values f. Applying the flow forward algorithm to these (equation 7), we find that:

$$||\underset{\sim}{s}'_2 - \underset{\sim}{s}'_1|| \quad = \quad \left\| \frac{1}{(1 - f \times r)} \times (\underset{\sim}{s}_2 - \underset{\sim}{s}_1) \right\| \qquad (11)$$

As f and r are between 0 and 1, equation 11 implies that

$$||\underset{\sim}{s}'_2 - \underset{\sim}{s}'_1|| \quad \geq \quad ||\underset{\sim}{s}_2 - \underset{\sim}{s}_1||$$

In fact, the distance between the flowed positions of the sample on the ST plane is always greater than their original distance, unless the new view is taken with a camera on the UV plane ($r = 0$), or the samples are on the ST plane ($f = 0$). The gap is not a uniform scaling on all samples, as it is dependent on flow value. Furthermore it is not caused by occlusion, which is responsible for holes appearing.

Tearing is a stretching effect. As the light field is implemented as a grid of point samples, the gap will between samples can be several pixels. In order to fill these gaps, we draw a flowed sample with a reconstruction kernel of a size proportional to

$$\frac{1}{(1 - f \times r)}$$

Figure 10 shows the effect of tearing. The light field is of a sphere, and is $8 \times 8 \times 512 \times 512$ in size. The gaps have been filled with a green colour. By using the reconstruction kernel, the gaps are filled up as seen in figure 11.

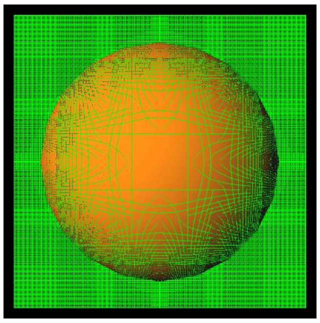

Figure 10: With tearing, shown in green.

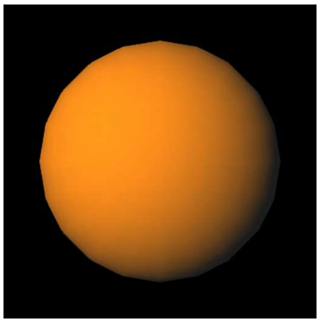

Figure 11: No tearing visible, using reconstruction.