THE USE OF HYPERLINES IN LIGHT FIELDS

George Chen[1]
Li Hong[1]
Peter McGuinness[1]
Kim Ng[1]
Christian Hofsetz[2]
Yang Liu[2]
Nelson Max[2]

(1) STMicroelectronics
(2) Center for Image Processing and Integrated Computing
University of California, Davis

In this paper, we introduce the concept of hyperlines in light fields. When represented by the two-plane parameterization, hyperlines are 2-Degree-of-Freedom (2-DOF) linear entities in the 4-D light field. The light field can be thought of as a dual space of the world space. In this dual space, cameras appear as hyperlines with heterogeneous colors which we call camera hyperlines (CHL), while scene points appear as hyperlines with homogeneous colors (assuming Lambertian object surfaces), which we call geometry hyperlines (GHL). They cross each other at the corresponding pixels. We derive equations for both types of hyperlines, which provide the basis for new unstructured light field rendering algorithms. We will present experimental results using both CHL's and GHL's as well.

1.INTRODUCTION

Current approaches in image-based rendering fall into two main categories: image-warping-based and light-field-based. When motion parallax exists, the warping based approaches require information on the scene geometry, which can be in the form of per-pixel depth [14] or polygonal models [1,3,5]. The depth value defines a correspondence between the pixel and a point in the scene.

When the point is warped to the virtual view, it carries over its color from the source image. A problem here is that a background point may leak through the foreground point cloud, bringing the wrong color to the projected pixel. One way to handle this is to increase the size of the scene point so that it occupies multiple pixels [9,11,12] and use a Z-buffer to resolve occlusion.

Alternatively, one can ensure that the point samples are dense enough to match the resolution of all the desired views [7]. Often in practice, however, finding an inexpensive way to obtain accurate and dense range data may be problematic.

Polygon-based image warping is grounded on the knowledge that two images of a planar facet are related by a homography [4]. Thus source images can be piecewise warped to the destination image, then, combined according to some weighting function.

To handle occlusion, image patches must be warped following a certain order [3] if a metric model is not available; otherwise, Z-buffer can be used. Similar to the previous case, recovering polygonal models from images using computer vision techniques is a no easy task.

The light field approach [2,6,10,13] has become attractive lately because it has been demonstrated that virtual views of reasonable visual quality can be synthesized, almost in real time, without any scene geometry. The key idea is to think of the scene as a space full of rays (thus the name light field), a portion of which is recorded by the cameras. Image synthesis then is nothing but rebinding the recorded rays according to the geometry of the virtual camera. Therefore, we need to address two related issues: how to store the rays sampled by the input cameras efficiently and how to retrieve rays from the storage quickly.

Two methods have been reported. The first method considers the light field as a four-dimensional space and samples at regular grid [6,10] or in some uniform fashion [2]. The color of a virtual ray is calculated by interpolating neighboring samples. When the two-plane parameterization is used [10], rendering can be done very fast, due to support from existing hardware. Spherical parameterization [2] is introduced to handle the discontinuity problem that occurs at the border of two two-plane parameterizations. Since normally the cameras are not regularly placed, rays do not distribute uniformly (instead, they form bundles), so input rays must be resampled in a pre-processing stage, which may cause aliasing effects that cannot be removed later. This method also needs to deal with compression of the sampled 4-D light field.

The second method uses the input images directly without resorting to the intermediate resampling step and allows free form camera placement. There are three issues here: camera selection, ray selection (in the selected cameras) and blending of the chosen rays. In [13], for example, a convex camera mesh whose vertices are the camera projection centers is formed. Given a virtual ray from a virtual camera, the triangle that intersects the ray is determined, and the three cameras at the vertices are chosen. To determine the ray in each camera, the virtual ray is further intersected with the image plane and the ray corresponding to the intersection is selected. Alternatively, if scene geometry is available, the ray that has the common intersection with the virtual ray at the geometry is selected. Finally, the blending weights are set proportional to the inverse distance between the virtual ray/triangle intersection and the triangle vertices. In [5], a different camera selection criterion is used based on the triangulation of the projection of the camera centers at the virtual image plane. In [1], the calculation of the blending weights is based on a list of desirable properties that an ideal image-based rendering should have.

1.1 Overview of approach

Our approach belongs to the last type, i.e., unstructured light field rendering without intermediate resampling. However, it uses different camera and ray selection criteria and weighting functions. Furthermore, our method is based on a generalization of the 4-D light field concept, as compared to the existing ones such as [1,5,13], which are defined in the 3-D world space.

We introduce the concept of hyperlines as linear entities in the 4-D light field with two degrees of freedom (DOF). We then show that cameras can be represented as hyperlines which are projective maps of the image planes. Instead of sampling a light field at discrete point locations, we think of it as already discretized at the camera hyperlines (CHL). Since in a light field a ray appears as a 4-D point, camera selection can be based on the 4-D point-hyperline distance. At the same time, the orthogonal projections of the 4-D point to the selected CHL's can be projectively mapped to the source images to retrieve rays for later blending. Finally, inverse 4-D distance can be used to compute blending weight.

In addition to the above, we show that scene points also appear as hyperlines in 4-D light fields, which we call geometry hyperlines (GHL). CHL's and GHL's cross each other at the corresponding pixels. Using GHL's for rendering results in

warping-equivalent algorithms. Towards that end, we have developed a novel algorithm to address occlusion. Last but not least, we can easily incorporate a dynamic focal plane [8] into the proposed scheme.

1.2 Paper organization

Section 2 derives the dual relationship between the light field and the world space. It then introduces the concept of hyperlines. Section 3 proposes algorithms for light field rendering using hyperlines. Section 4 presents some experimental results. Finally, section 5 offers our conclusions. In this paper, an image plane is considered continuous and infinite. A pixel is any point on the image plane. A ray is a line that passes through the camera center-of-projection and a pixel. On a camera, since pixels and rays are uniquely paired, we use them interchangeably in this paper.

2. DUAL REPRESENTATIONS OF LIGHT FIELDS

2.1 Light field parameterization

Typically, a scene consists of several objects and cameras. Cameras observe points on object surfaces. If a ray intersects at the visible portion of a surface, it inherits the color from the intersection point. We assume object surfaces are Lambertian so that all rays that observe the same surface point have the same color.

We define the light field as the set of rays recorded by the cameras in the scene. Using the well-known two-plane parameterization [6,10], a ray has a quadruple representation (s,t,u,v) where (s,t) and (u,v) respectively are the coordinates of intersections of the ray with the two parameterization planes (Figure 1). This suggests a dual relationship between the world space, where rays appear as lines and a 4-D space, where rays appear as points. From now on, the phrase light field means the two-plane parameterized 4-D space.

2.2 Hyperlines in dual world space

In Fig. 1, $P=[x,y,z,]^T$ is a scene point. Let $Q=[s,t,u,v,]^T$ be the light field coordinate of a line through P. Without loss of generality, assume the ***x*-*y*** plane is parallel to the ***s*-*t*** plane at z_{st} and the ***u*-*v*** plane at z_{uv}. It follows from simple trigonometry calculation that

$$\frac{s-x}{u-x}=\frac{t-y}{v-y}=\frac{z-z_{st}}{z-z_{uv}} \tag{1}$$

which has two equivalent matrix forms:

$$\begin{bmatrix} z_{uv}-z_{st} & 0 & s-u \\ 0 & z_{uv}-z_{st} & t-v \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} sz_{uv}-uz_{st} \\ tz_{uv}-vz_{st} \end{bmatrix} \tag{2a}$$

$$\begin{bmatrix} z-z_{uv} & 0 & z_{st}-z & 0 \\ 0 & z-z_{uv} & 0 & z_{st}-z \end{bmatrix}\begin{bmatrix} s \\ t \\ u \\ v \end{bmatrix}$$

$$=\begin{bmatrix} x(z_{st}-z_{uv}) \\ y(z_{st}-z_{uv}) \end{bmatrix}. \tag{2b}$$

Since (2b) describes the intersection of two 4-D hyperplanes, thus defining a two dimensional subspace of R^4, the resulting 4-D entity is called a hyperline. A general form of (2b) is

$$\begin{bmatrix} a & 0 & b & 0 \\ 0 & a & 0 & b \end{bmatrix}\begin{bmatrix} s \\ t \\ u \\ v \end{bmatrix}=\begin{bmatrix} c \\ d \end{bmatrix}, \tag{3}$$

where a and b cannot be zero at the same time, or equivalently, $a^2+b^2<0$.

We describe the dual relationship between world space and light field in the following three remarks:

(1) World space and light field are dual spaces: points in one space correspond to lines or hyperlines in the other space, and vice versa. For example, the point P in world space corresponds to the hyperline in a light field whose equation is given by (2b); and the point Q in the light field corresponds to the line in world space whose equation is given by (2a).

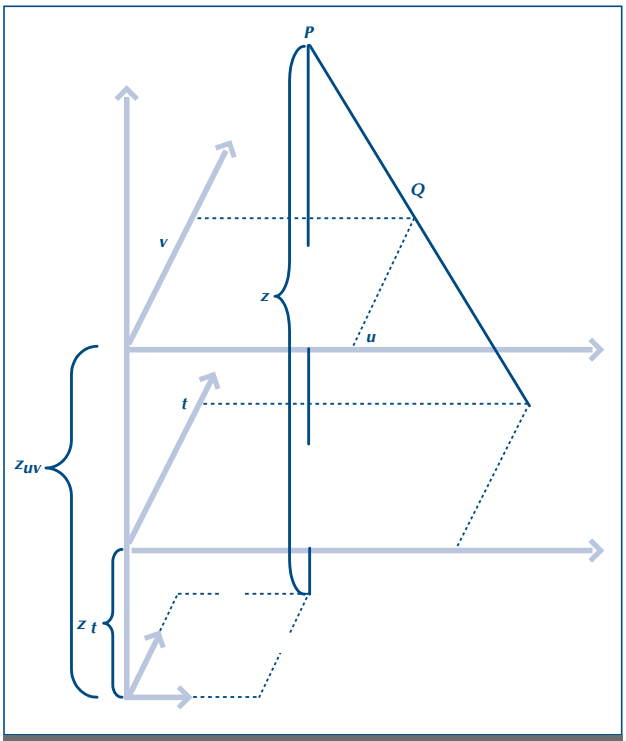


FIGURE 1: TWO-PLANE PARAMETERIZATION AND DERIVATION OF DUAL RELATIONSHIP.

(2) A bundle of lines in world space correspond to a set of co-hyperlinear points in light field. For example, when *x, y, z* are fixed while *s, t, u, v* are varying, (2a) describes a bundle of lines in world space all passing through *P*, and (2b) describes a set of co-hyperlinear points in R^4.

(3) A set of collinear points in world space corresponds to a bundle of hyperlines in light field. For example, when *s, t, u, v* are fixed while *x, y, z* are varying, (2a) describes a set of collinear points and (2b) describes a bundle of hyperlines all passing through the 4-D point *Q*.

We would also like to point out the relationship between the slope *k* of a hyperline and the depth *z* of the corresponding 3-D point: $k=b/a=(z_{st}-z)/(z-z_{cv})$. Particularly, for two 3-D points of the same depth, their hyperlines are parallel.

2.1 Camera hyperlines (CHL) and geometry hyperlines (GHL)

In world space, a viewing ray is a line segment with color, one end at a camera projection center, another at a scene point. Rays form two kinds of bundles: one around the camera centers with heterogeneous colors; the other around scene points with a homogeneous color. In the dual space, or light field, the bundles transform to two types of hyperlines: the first type – a camera hyperline – has heterogeneous colors. It represents bundles of rays emitting from camera projection centers.

The second type – a geometry hyperline – has a homogeneous color. It represents bundles of rays going to points on object surfaces. Intersections of these two types of hyperlines are corresponding pixels on the image planes. From now on, a camera and its basic elements – center-of-projection, image plane, ray, pixel – all have dual appearances. However, we do not phrase them differently. For instance, a ray can be either a 3-D line of a 4-D point, but is unvaryingly called a “ray”. Readers are expected to figure out the actual appearance from the context.

3. LIGHT FIELD RENDERING WITH HYPERLINES

Light field rendering with hyperlines is rather straightforward. First, convert cameras and scene points to hyperlines. Second, for each virtual ray, select hyperlines by minimizing a cost function.

Third, for the same virtual ray, choose and then blend rays from the just selected hyperlines if CHL’s are used. If GHL’s are used instead, since they are homogeneous in color, ray selection is unnecessary. However, the occlusion issue must be addressed. Hyperline selection can be accelerated if the hyperlines are organized into a hexadeca-tree by dividing the light field in consideration into quad-cells, and register hyperlines to those cells they pass through.

3.1 CHL-based rendering

Here, we perform camera selection and ray selection in one step. The main issue is deciding the source ray (s_i, t_i, u_i, v_i) on a given

camera hyperline (a_i, b_i, c_i, d_i) close to a virtual ray (s_o, t_o, u_o, v_o) in some optimal sense. The cost then determines which cameras and rays should be selected. One intuitive criterion is to minimize the squared 4-D distance between the source and the virtual rays:

$$f(R_i) = (s_i - s_o)^2 + (t_i - t_o)^2 + (u_i - u_o)^2 + (v_i - v_o)^2$$

subject to $a_i, s_i+b_iu_i=c_i$ and $a_it_i+b_iv_i=d_i$. The solution is

$$R_i = \begin{bmatrix} \frac{b_i^2 s_o - a_i b_i u_o + a_i c_i}{a_i^2 + b_i^2}, \\ \frac{b_i^2 t_o - a_i b_i v_o + ad}{a_i^2 + b_i^2}, \\ \frac{a_i^2 u_o - a_i b_i s_o + b_i c_i}{a_i^2 + b_i^2}, \\ \frac{a_i^2 v_o - a_i b_i t_o + b_i d_i}{a_i^2 + b_i^2} \end{bmatrix}$$

The 4-D distance is

$$f_{min}(R_i) = \sqrt{\frac{(a_i s_o + b_i u_o - c_i)^2 + (a_i t_o + b_i v_o - d_i)^2}{a_i^2 + b_i^2}}. \quad (4)$$

Based on (4), we choose and blend the first few source rays with the shortest 4-D distance according to the inverse distance.

The above criterion works well when the ***s-t*** plane is close to the cameras and the ***u-v*** plane is close to the objects. To see this, we place the camera on the ***s-t*** plane, *i.e.*, $z_c=z_{st}$ or $b_i=0$.

The ray with the shortest distance is $[c_i/a_i \;\; d_i/a_i \;\; u_o \; v_o]^T$ which coincides with the virtual ray at the *u-v* plane, as they have the same *u-v* coordinates. Since ideally we would like to choose a source ray that coincides with the virtual one on the object surface, the further away the *u-v* plane is from the object, the worse the choice determined by this criterion. Another conclusion we draw from the previous analysis is that two rays with non-zero 4-D distance may have zero 3-D distance.

One possible option is to require additionally that the selected source ray have a direction similar to the virtual one. We therefore minimize

$$f(R_i) = \left[(s_i - s_o)^2 + (t_i - t_o)^2 + (u_i - u_o)^2 + (v_i - v_o)^2\right] + \beta\left[(s_i - s_o - u_i + u_o)^2 + (t_i - t_o - v_i + v_o)^2\right], \quad (5)$$

subject to $a_is_i+b_iu_i=c_i$ and $a_it_i+b_iv_i=d_i$. The first term accounts for light field distance as before. The second term accounts for orientation difference in world space. For example, it becomes zero when the selected source ray (s_i, t_i, u_i, v_i) and the virtual one (s_o, t_o, u_o, v_o) are parallel. β is the balancing factor.

Without listing the solution formulas, let us analyze the role of β by looking at the depth of the intersection of the aforementioned two rays (recall our earlier declaration that the source rays are from a continuous rather than sampled image so that we can always choose one to intersect the virtual ray), which is

$$Z(\beta) = \frac{a_i z_{uv} - b_i z_{st} - \beta(z_{uv} - z_{st})^2}{a_i - b_i}.$$

First, notice that although we introduce a balancing factor to each CHL, all of them become linearly related if the selected source rays intersect with the virtual one at the same depth. Thus we only need to designate one of these factors. We can automatically computer the rest. Second, we can choose β so that the intersection is at a desired depth. Said differently, with a given β, objects at depth $Z(\beta)$ appear sharp in the virtual image, while those at different depths have ghost images. In this way, β performs a similar role to the dynamic focal plane of [8] but is derived from quite a different motivation here. Third, if $\beta=0$, it is not hard to deduce that the sufficient and necessary condition for $Z(0)=z_{uv}$ is $b_i=0$, which is consistent with the conclusion we drew previously. Fourth, if $\beta=\infty$, we select the source ray that is parallel to the virtual one, which is nothing but a rephrase of $Z(\infty)=\infty$.

3.2 GHL-based rendering

Selecting GHL's using distance-based cost functions does not handle occlusion correctly. In Fig. 2, (a) and (c) show a vicinity of a virtual ray going through a point cloud sampled from two object surfaces; (b) and (d) show the same vicinity in the light field. Notice how points and lines change appearances in the dual image pairs. For the scenario of Fig. 2(a), we would like the ray to have the color of the foreground. However, as depicted in Fig. 2(b), our selection criterion wrongly assigns the background color to the ray. This is the *background leakage* problem. For Fig. 2(c), the ray should have the background color. However, according to Fig. 2(d), it is closer to the foreground in the light field. This is the *foreground expansion* problem.

We base our solution to the above problems on two observations. First, in general, the dense point cloud in the vicinity of a ray is clustered around the same depth. As a result, GHL's are clustered according to their depth too (recall $z=a+z_{uv}=z_{st}-b$). The second observation is that if a ray *passes through* a cluster, GHLs in the vicinity surround it. If it passes by a cluster, all GHL's are at one side. We thus conclude that we should search for the cluster that has the minimum depth and at the same time is surrounding the virtual ray.

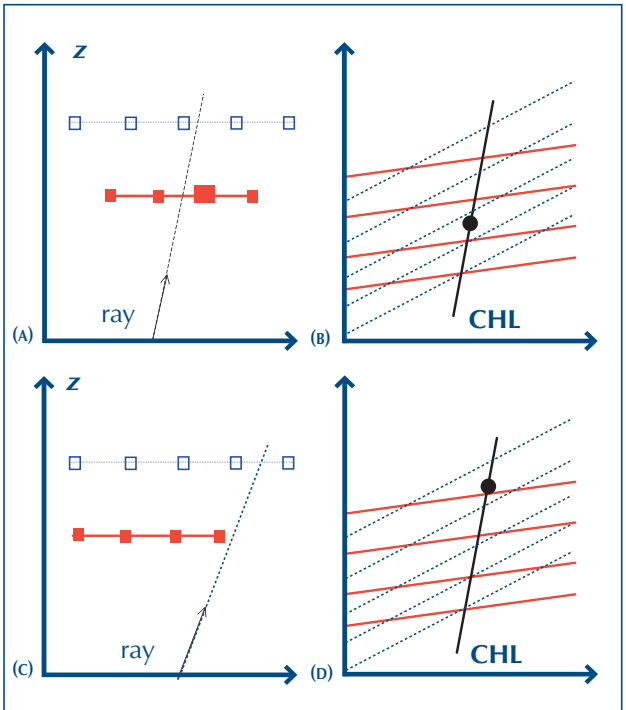


FIGURE 2: (A) A RAY GOES THROUGH BOTH FOREGROUND AND BACKGROUND. (B) THE DISTANCE CRITERION SELECTS THE WRONG GHL. (C) A RAY GOES BY THE FOREGROUND. (D) THE DISTANCE CRITERION AGAIN SELECTS THE WRONG GHL.

Determining virtual ray vicinity

Conceptually, the light field vicinity of a ray $\boldsymbol{r}$ is the circular region on the virtual camera hyperline centered at $\boldsymbol{r}$. We are interested in the set of GHL's, denoted as $G(\boldsymbol{r})$, that intersect with the region. Ideally, the vicinity size should be adaptive, keeping $|G(\boldsymbol{r})|$ almost constant. Currently, it is pre-defined. To make the specification easy, it is given in image space. This is possible because the ***s-t*** and ***u-v*** planes are related to the virtual image plane by homography. Thus we can map image vicinity to light field vicinity by a function which we denote as h. Let V_s be the vicinity size in image space, then, $G(\boldsymbol{r})=\{g \mid \|I(g)-\boldsymbol{r}\|2 < h(V_s)\}$, where $I(g)$ is the intersection of hyperline g with the virtual CHL.

GHL clustering based on depth

A clustering of $G(\boldsymbol{r})$ is a set $\{G_1, G_2, \ldots, G_n\}$ such that $\cup G_i = G(\boldsymbol{r})$, and $G_i \cap G_j = \Phi$ for $i \neq j$. Each G_i is called a cluster.

The observation that each cluster is around the same depth implies that, for a given dataset, there exists a threshold V_d such that the depth range of each cluster is no greater than V_d.

More formally, a depth based clustering of $G(\boldsymbol{r})$, denoted as $C(G(r))$, is defined to satisfy the following additional constraints:

(1) Compactness: $max(G_i)-min(G_i) \leq V_d$ where $max(G_i)$ and $\min(G_i)$ are respectively the maximal and minimal depth of G_i;

(2) Enclosure: for $\forall g$, if $min(G_i) \leq depth(g) \leq max(G_i)$, then $g \in G_i$;

(3) Maximum: for $\forall g \in G(\boldsymbol{r})$, but $g \in G_i$, $max(G_i \cup \{g\})-min(G_i \cup \{g\}) > V_d$.

Provided the existence of V_d, these constraints guarantee the uniqueness of such a clustering. In practice, it is not hard to decide V_d because in most cases the clusters are well separated, as shown in Fig. 2. Optionally, we can compute V_d from the input data given a priori image regions where occlusion does not happen. A quick way of computing $\boldsymbol{C}(G(\boldsymbol{r}))$ is to sort all GHL's according to their depth in a pre-processing stage; then, in the order of ascending depth, each $g_i \in G(\boldsymbol{r})$ belongs to a cluster G_i either if G_i is empty or $depth(g_j)–depth(g_o)<V_d$, where g_o is the first GHL of G_i.

Opacity of a GHL-cluster

We now define a boolean function - *opacity* - to indicate whether or not a cluster is surrounding a virtual ray. Let us look at the foreground plane of Fig. 2 from a top view, now displayed in Fig. 3. We claim that the necessary and sufficient condition for all points of a cluster to surround the ray is that the maximal spanning angle from the ray to the cluster be equal or greater than 180°.

For $g_i \in G_i \in \boldsymbol{C}(G(\boldsymbol{r}))$, we can rewrite the 3-D coordinate of the scene point P_j corresponding to g_j in terms of (a_j, b_j, c_j, d_j) , and the line equation of r in terms of (s_o, t_o, u_o, v_o). We can shown that the vector from P_j towards $\boldsymbol{r}$ that is parallel to the $\boldsymbol{x}$-$\boldsymbol{y}$ plane is a multiple of $v_j(\boldsymbol{r})=(a_j s_o+b_j u_o-c,\ a_j t_o+b_j v_o-d_j,\ 0)$. We thus define

$$\underset{G_i \in C(G(r))}{opacity}(G_i) = \begin{cases} true & \text{if } v_j(r)\text{'s span equal} \\ & \text{or greater than } 180^{o}; \\ false & \text{otherwise.} \end{cases} \quad (6)$$

When *opacity* (G_i) is true, we say the cluster is opaque, meaning that the color of $\boldsymbol{r}$ should come from G_i. Otherwise, the cluster is transparent, meaning the color should come from another cluster farther back. Comparing (4) and (6), we realize that what has been missing in the former is the direction of $v_j(\boldsymbol{r})$.

We now summarize the steps in GHL-based rendering:

— Sort all GHL's according to their depth during pre-processing.

— For each virtual ray $\boldsymbol{r}$, find $\boldsymbol{G}(\boldsymbol{r})$ in a vicinity of $\boldsymbol{r}$, and compute $\boldsymbol{C}(G(\boldsymbol{r}))$. Note that $\boldsymbol{C}$ is already in the ascending order of depth.

— Find the first $G_i \in \boldsymbol{C}(G(\boldsymbol{r}))$ such that *opacity* (G_i) is true.

— Blend colors of GHL's in G_i, weighted by $1/\|v_j(\boldsymbol{r})\|_2$.

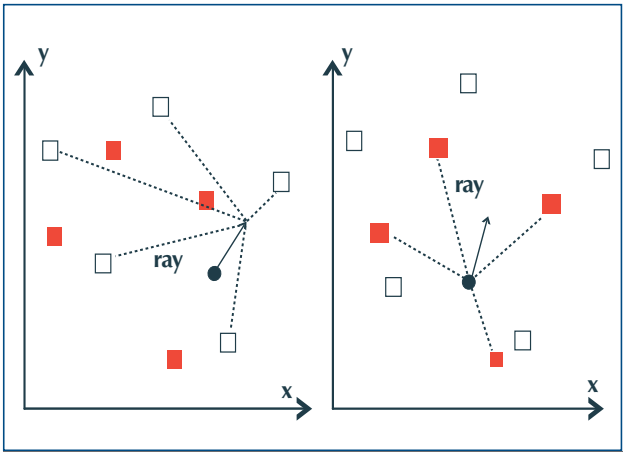


FIGURE 3: ILLUSTRATION OF THE OPACITY FUNCTION. LEFT: BOTH CLUSTERS ARE OPAQUE. THE CLOSER ONE IS SELECTED. RIGHT: FOREGROUND CLUSTER IS TRANSPARENT. THE BACKGROUND ONE IS SELECTED.

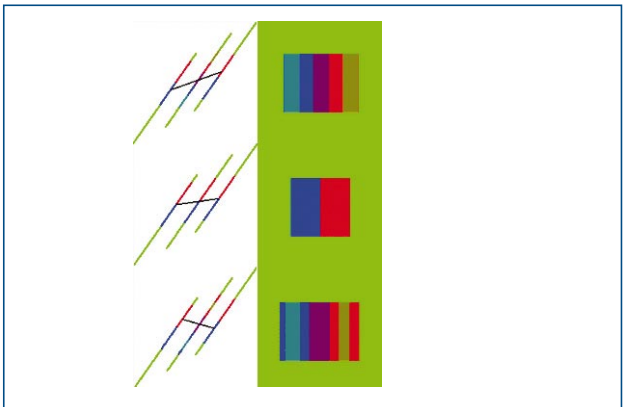


FIGURE 4: ILLUSTRATION OF CHL-BASED RENDERING AND THE EFFECT OF β. THE FIRST COLUMN SHOWS A SLICE OF THE LIGHT FIELD AT T=V=0, WHICH CONSISTS OF THREE TILTED CHL'S. THE CENTRAL ONE COMES FROM THE VIRTUAL CAMERA. THE BLACK LINE INDICATES RAYS SELECTED TO SYNTHESIZE THE CENTRAL PIXEL. THE SECOND COLUMN SHOWS THE SYNTHESIZED IMAGE.

4. EXPERIMENTAL RESULTS

First, we use a very simple example to illustrate CHL-based rendering and the effect of the β factor. The scene consists of two rectangles at depth 0, one red, and the other blue, and a green

background. The two input cameras are at (-0.75, 0, 6) and (0.65, 0, 6), both looking at the negative z direction. The virtual camera is placed at (0, 0, 6), also looking at negative z. The two planes are z_{st} = 4 and z_{uv} = 1. Results are displayed in Fig. 4. Observe that changing β in effect rotates the black line in the first column. This is like guessing the depth of the central ray (or slope of the corresponding GHL). When the guess is correct, the black line has the correct slope, which implies correct correspondence. As a result, the correct image is synthesized. Otherwise ghost images are generated.

In the second example, we equally place twenty-five cameras spanning a 45-degree arc around a teapot and set β so that the focal plane is located at the teapot body. In Fig. 5, images in the top row use the 4 closest CHL's for blending. We observe slight blurring. The lower left one uses the single best CHL. As a result, it looks sharp. For comparison, we display a ground truth image at lower right.

Fig. 6 demonstrates the effect of a large aperture coupled with a dynamic focal plane. The foreground of the scene is a shiny teapot; the background is a texture-mapped plane. There are 50 source views. Each virtual pixel is equally blended from the 25 closest CHL's. Notice how the background becomes blurry in the lower left image, and the teapot almost disappears in the lower right image. Using hardware support for texture mapping, we have reached 10fps average frame rate at the 512×512 resolution on a 550MHz Pentium III.

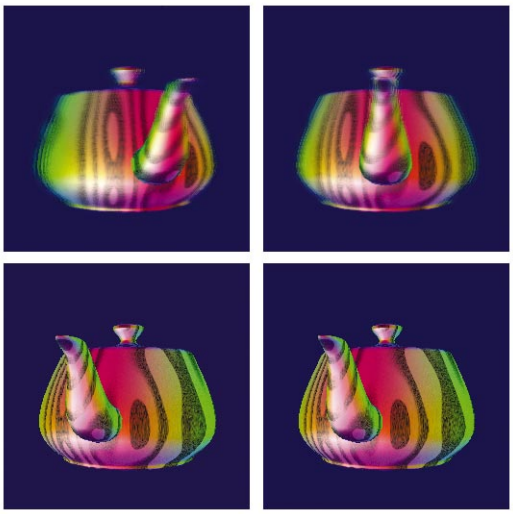


FIGURE 5: CHL-BASED RENDERING OF A TEAPOT.

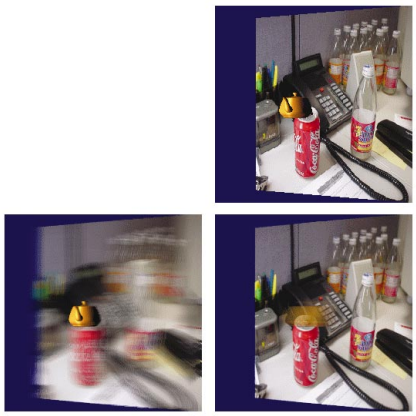


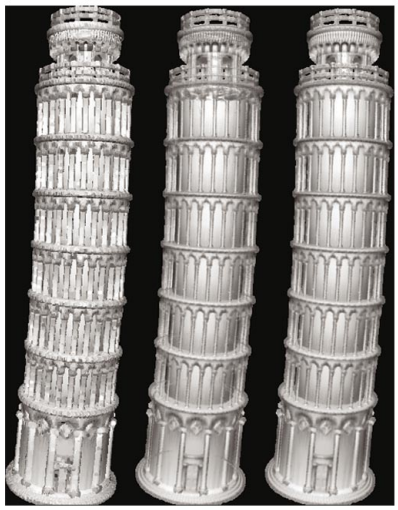
FIGURE 6: DYNAMIC FOCAL PLANE BY ADJUSTING β.

Next are results from GHL-based rendering. Fig. 7 depicts the use of the depth variance threshold for the Pisa Tower dataset. If V_d=0 as in (a), the solution is aliased. This happens because each cluster contains only one GHL. If V_d is large (b), two clusters are mixed. As a result, we can see through the tower. Using a proper V_d as in (c), the image looks smooth and occlusion is negligible.

Fig. 8 shows the effect of changing the vicinity size V_s. If V_s is too small (a), $G(r)$ may be empty for some r, generating a hole there. Also, if $G(r)$ contains only background GHL's, then there is a background leakage at r. If V_s is large and we turn off the opacity check (b), the image becomes blurry and the silhouettes look fattened. When the opacity check is turned on (c), a good-looking solution is reached.

Fig. 9 compares rendering results of the "car wreck scene" between a polygonal renderer and our GHL renderer (range data courtesy of University of North Carolina at Chapel Hill IBR group).

Not all the differences are noticeable at the current resolution, so we point a few out here. On the left car, notice the window, the hubcap and the seam between the front and back doors. On the



(A) VD =0 (B) VD =100 (C) VD =3

FIGURE 7: THE "PISA TOWER", USING 6 IMAGES AND 197190 GHL'S.

(A) VS =0.3 (B) VS =0.3, WITHOUT OPACITY (C) VS =0.3, WITH OPACITY

FIGURE 8: THE "MATE", USING 4 IMAGES (300X300 RESOLUTION) AND THEIR DEPTH MAPS.

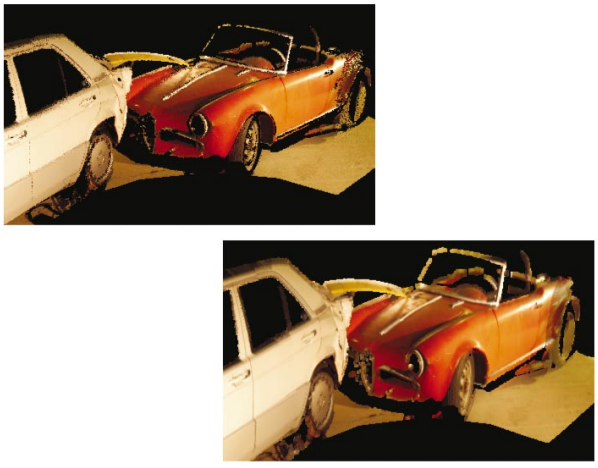


FIGURE 9: THE "CAR WRECK SCENE", USING 9 IMAGES (864X576 RESOLUTION) AND THEIR RANGE DATA.

right car, notice the areas around the real wheel and the vertical ridge on the hood. In all these cases, the right image looks smoother and more visually pleasant.

Currently, our implementation is purely software-based. Also, compared to CHL-based rendering, we spend significant time on occlusion handling. On the same machine, the frame rate at the 512×512 resolution is about 0.2~2 fps depending on the number of hyperlines we use. Below is a summary of our statistics of computation workload.

SYSTEMMODULE	RENDERING	OCCLUSION	OVERHEAAD
Pisa	85.30%	13.42%	1.28%
Mate	92.84%	6.40%	0.76%
Car	61.80%	37.82%	0.38%

TABLE 1. WORKLOAD STATISTICS OF GHL-BASED RENDERING.

5. CONCLUSION

Our contributions in this paper are in three aspects: the derivation of the dual relationship between 4-D light field and 3-D world space, the introduction of hyperlines into light fields, and finally, the use of hyperlines for light field rendering. The proposed formalism also provides some new insights into the study of light fields. Implementation-wise, we address four main issues, namely, hyperline selection, dynamic focal plane, foreground expansion and background leakage. Since both camera and scene geometry have a single representation, it is possible to design an algorithm that freely adapts itself between light field rendering and geometry-based rendering according to the density of cameras and scene points, a direction that we plan to pursue in the future.

ACKNOWLEDGMENTS

This work has been co-sponsored by the Advanced System Technology of STMicroelectronics and the Digital Media Innovation Program (DiMI) from the University of California. The authors would like to thank Lars Nielsen, Anselmo Lastra, Nick England and David McAllister of the University of Carolina at Chapel Hill for the car wreck models. Hofsetz's study has also been sponsored by UNISINOS – Universidade

do Vale do Rio dos Sinos and CAPES – Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

REFERENCES

1. C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, "UNSTRUCTURED LUMIGRAPH RENDERING," SIGGRAPH 2001.

2. E. Camahort, A. Lerios, D. Fussell, "UNIFORMLY SAMPLED LIGHT FIELDS," Eurographics Rendering Workshop '98, 117-130, 1998.

3. Q. Chen and G. Medioni, "IMAGE SYNTHESIS FROM A SPARSE SET OF VIEWS," IEEE Visualization' 97, 269-275, 1997.

4. R. Hartley and A. Zisserman, "MULTIPLE VIEW GEOMETRY," Cambridge University Press, 2000.

5. B. Heigl, M. Pollefeys, J. Denzler and L. van Gool, "PLENOPTIC MODELING AND RENDERING FROM IMAGE SEQUENCES TAKEN BY A HAND-HELD CAMERA," 21 Symposium für Mustererkennung (DAGM'99), 94-101, 1999.

6. S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "THE LUMIGRAPH," SIGGRAPH 96, 43-54.

7. J.P. Grossman, William J. Dally, "POINT SAMPLE RENDERING," Eurographics Rendering Workshop 98, 181-192, 1998.

8. A. Isaksen, L. McMillan, S. J. Gortler, "DYNAMICALLY REPARAMETERIZED LIGHT FIELD," SIGGRAPH 2000, 297-306.

9. M. Levoy and T. Whitted, "THE USE OF POINTS AS A DISPLAY PRIMITIVE," Technical Report TR 85-022, Univ. of North Carolina at Chapel Hill, 1985.

10. M. Levoy and P. Hanrahan, "LIGHT FIELD RENDERING," SIGGRAPH 96, 31-42, 1996.

11. H. Pfister, M. Zwicker, J. van Baar, M. Cross, "SURFELS: SURFACE ELEMENTS AS RENDERING PRIMITIVES," SIGGRRAPH 2000, 335-342, 2001.

12. S. Rusinkiewicz, M. Levoy, "QSPLAT: A MULTIRESOLUTION POINT RENDERING SYSTEM FOR LARGE MESHES," SIGGRAPH 2000, 343-352, 2000.

13. H. Schirmacher, C. Vogelgsang, H.-P. Seidel, G. Greiner, "EFFICIENT FREE FORM LIGHT FIELD RENDERING," Vision, Modeling, and Visualization 2001, 249-256, 2001.

14. H. Schirmacher, L. Ming, H.-P. Seidel, "ON-THE-FLY PROCESSING OF GENERALIZED LUMIGRAPHS," Eurographics 2001, 20(3):C165-C173, 2001.

■ CONTACT: ST.JOURNAL@ST.COM ■