Light Field Duality: Concept and Applications

George Chen, Li Hong, Kim Ng, Peter McGuinness
Human-Computer Interface Laboratory
STMicroelectronics
4690 Executive Dr. Suite 200
San Diego, CA 92121
(01) 858 452 7715
Email: george-qian.chen@st.com

Christian Hofsetz, Yang Liu, Nelson Max
CIPIC – Department of Computer Science
University of California Davis
One Shields Avenue,
Davis, CA 95616
(01) 530 752 2376
Email: chofsetz@ucdavis.edu

ABSTRACT

We propose to look at light fields from a dual space point of view. The advantage, in addition to revealing some new insights, is a framework that combines the benefits of many existing works. Using the well known two-plane-parameterization, we derive the duality between the 4-D light field and the 3-D world space. In the dual light field, rays become hyper points. We introduce the concept of hyperline. Then, cameras appear as hyperlines – camera hyperline (CHL) – mostly heterogeneous in color; scene points also appear as hyperlines – geometry hyperline (GHL) – mostly homogeneous in color. CHL's and GHL's are independent. The existence of one does not require or replace the other. When both exist, they cross each other at the dual ray hyper points. Both CHL and GHL-based light field rendering results are presented.

Categories and Subject Descriptors

I.3.3 [**Picture/Image Generation**]: Display algorithms, Viewing algorithms; I.3.7 [**Three-Dimensional Graphics and Realism**]: Ray tracing, Virtual reality.

General Terms

Algorithms, Experimentation, Theory.

Keywords

Light field rendering, point sample rendering, dual space.

1. INTRODUCTION

Current approaches in image-based rendering fall into two main categories: geometry based and light field based. Geometry based approaches transfer input images to the virtual camera through the usage of scene geometry, which can be in the form of per-pixel depth [17] or polygonal models [2],[4],[6]. The depth value defines a correspondence between the pixel and a point in the scene. When the latter is warped to the virtual view, it carries over its color from the source view. 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 [11],[13],[15] 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 [8]. Often in practice, however, finding an inexpensive way to obtain accurate and dense range data may be problematic.

Using polygons as image transfer primitives is grounded on the knowledge that two images of a planar facet are related by a homography [5]. Thus source images can be piecewise warped to the destination image and combined according to some weighting function. To handle occlusion, image patches must be warped following a certain order [4] if a metric model is not available; otherwise, a Z-buffer can be utilized. Similar to the previous case, recovering polygonal models from images using computer vision techniques is a non-trivial task.

The light field approach [3],[7],[12] 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 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 source cameras. Image synthesis then is nothing but rebinding the recorded rays according to the geometry of the virtual camera. Therefore, two related issues need to be addressed: how to efficiently store the rays sampled by the source cameras and how to quickly retrieve rays from the storage.

Two methods have been reported. The first method considers the light field as a four-dimensional space and samples at regular grid [7],[12] or in some uniform fashion [3]. The color of a virtual ray is calculated by interpolating neighboring samples. When the two-plane parameterization (2PP) is used [7],[12], rendering can be done very fast due to support from existing hardware. Spherical parameterization [3] was introduced to handle the discontinuity problem that occurs when switching from one 2PP to another. 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 involved: camera selection, ray selection (in the selected cameras) and blending of the chosen rays. In [16], 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 chosen camera, the virtual ray is further intersected with its 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 [6], a different camera selection criterion is used based on the triangulation of the projection of the camera centers at the virtual image plane. In [2], 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 our approach

We model scene objects as point clouds. The points are called scene points. A light field is the set of rays either recorded by the cameras, or emitted from scene points. Both cameras (actually, projection centers) and scene points can be thought of as bundles of rays. Their roles are symmetric with respect to the rays. The essence of the proposed framework is treating the 4-D light field as a dual space of the 3-D world space so that the point-ray-camera relationship has a dual appearance. We introduce the concept of *hyperline* which is the intersection of two 4-D hyper planes in the dual light field, with two degrees of freedom (it is called a "line" because two hyperlines intersect into a hyper point). Then, both cameras and scene points appear as hyperlines, and rays appear as hyper points on hyperlines. Hyperlines representing cameras obtain their colors from the images, therefore, they are mostly heterogeneous. Hyperlines representing scene points inherit their colors from the latter, thus are mostly homogeneous. In the proposed framework, ray selection becomes hyper point selection from hyperlines (either kind or both), as opposed to 3-D line selection.

1.2 Contributions

The proposed light field rendering framework combines the advantages of many previous works. For example, it allows for implementing both geometry based and light field based algorithms; it uses the 4-D light field parameterization but removes the need of resampling, thus avoids dealing with the follow-on issues such as anti-aliasing and compression; it allows for free-form camera placement; it incorporates nicely the dynamic focal plane scheme [10]; finally, the resulting rendering algorithm can be efficiently implemented as projective texture mapping.

To our knowledge, light field has not been studied as a dual space before. There are three practically meaningful cases: cameras are dense, scene points are dense, and both are sparse. This paper addresses the first two. But we believe our framework has the potential to address the third one because it makes easier using the camera and geometry information concurrently, as a result of having a single representation for both. Our second intent is to stimulate some interest into looking at light fields from a dual space point of view, which might lead to some new insights into this increasingly popular topic.

1.3 Paper organization

Section 2 derives the dual relationship between the light field and the world space. It then introduces the concept of hyperline. Section 3 and section 4 propose algorithms for light field rendering using hyperlines. Section 5 presents some experimental results. Finally, in section 6, conclusions are offered.

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, they are used interchangeably in this paper.

2. A DUAL REPRESENTATION OF THE LIGHT FIELD

2.1 Derivation of the duality

Using the two-plane parameterization (2PP) [7],[12], 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 2PP parameterized 4-D dual space.

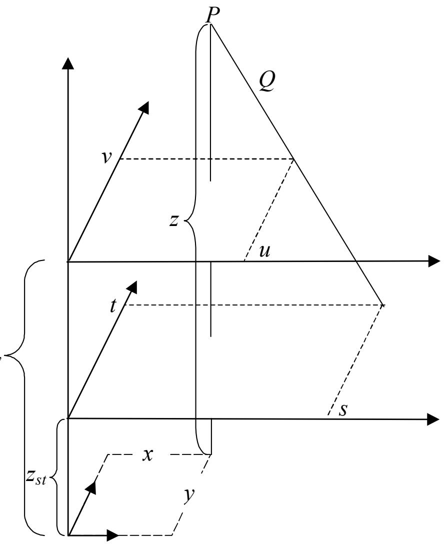


Figure 1: The two-plane- parameterization and derivation of dual relationship.

In Figure 1, $P = (x, y, z)$ is a scene point. Let $Q = (s, t, u, v)$ 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 similar triangles or 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 hyper planes, defining a two dimensional subspace of R^4, the resulting linear 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, $a^2+b^2\neq0$. From now on, we use a quadruple (a, b, c, d) to represent a hyperline. Obviously, a hyperline represented by (3) has two degrees of freedom (2-DOF). The intersection of two hyperlines is that of four hyper planes, thus a hyper point (0-DOF), unless the hyperlines are parallel.

The dual relationship between world space and light field is described by the following three remarks:

i). 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 the 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).

ii). A bundle of lines in world space correspond to a set of co-hyperlinear points in the 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.

iii). A set of collinear points in world space corresponds to a bundle of hyperlines in the 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 like to point out the relationship between the slope k of a hyperline and the depth z of the corresponding 3-D point: $k = -a/b = (z-z_{uv})/(z-z_{st})$. Particularly, for two 3-D points of same depth, their hyperlines are parallel. This observation was also made by Gu *et al.* in [9] where they characterized a bundle of lines in the world space as a 2-D affine subspace in the light field. They did not explicitly make use of the duality between the 3-D line bundle and the 4-D hyperline.

2.2 Camera hyperlines and geometry hyperlines

In world space, a ray is a vector with color, starting from a scene point and ending at a camera projection center. Rays form two kinds of bundles: one around the camera centers, which is mostly heterogeneous in color; the other around scene points, which is mostly homogeneous in color. In the dual space, or light field, the bundles transform into two types of hyperlines: the first type – camera hyperline (CHL) – representing bundles of rays collected by the cameras, is mostly heterogeneous. The second type – geometry hyperline (GHL) – representing bundles of rays emitted from the scene points, is mostly homogeneous. Based on our definition, the existence of a light field requires either the CHL's or the GHL's, not necessarily both. This allows for developing rendering algorithms using either kind of hyperlines. When both exist, their intersections are the dual ray hyper points. In our framework, camera placement does not have to be coplanar [12] or in some uniform fashion [3]. Consequently, unstructured light field rendering can be achieved.

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. UNSTRUCTURED LIGHT FIELD RENDERING USING CHL

Light field rendering using CHL's is rather straightforward. First, convert input images to CHL's. 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. Notice since there is a one-to-one mapping between a CHL and an image plane, there is no need to perform resampling on CHL's.

3.1 Selection by 4-D distance

Here, camera selection and ray selection are done in one step. The main issue is deciding the source ray $\boldsymbol{r}_i = (s_i, t_i, u_i, v_i)$ on a given camera hyperline $\boldsymbol{l}_i = (a_i, b_i, c_i, d_i)$ close to a virtual ray $\boldsymbol{r} = (s_r, t_r, u_r, v_r)$ 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 $\boldsymbol{r}_i$ and $\boldsymbol{r}$:

$d^2 = (s_i - s_r)^2 + (t_i - t_r)^2 + (u_i - u_r)^2 + (v_i - v_r)^2$ subject to $a_is_i+b_iu_i=c_i$ and $a_it_i+b_iv_i=d_i$. The solution is

$$\boldsymbol{r}_i = \begin{bmatrix} \dfrac{b_i^2 s_r - a_i b_i u_r + a_i c_i}{a_i^2 + b_i^2}, \\ \dfrac{b_i^2 t_r - a_i b_i v_r + a_i d_i}{a_i^2 + b_i^2}, \\ \dfrac{a_i^2 u_r - a_i b_i s_r + b_i c_i}{a_i^2 + b_i^2}, \\ \dfrac{a_i^2 v_r - a_i b_i t_r + b_i d_i}{a_i^2 + b_i^2} \end{bmatrix}$$

and

$$d_{\min}^2 = \frac{(a_i s_r + b_i u_r - c_i)^2 + (a_i t_r + b_i v_r - d_i)^2}{a_i^2 + b_i^2}. \tag{4}$$

Based on (4), the first few source rays with the shortest 4-D distance are chosen, and blended according to the inverse distance.

To see the geometrical meaning of this solution, place all cameras on the ***s-t*** plane, *i.e.*, $z_c=z_{st}$, or $b_i=0$. As a result, $x_c=s_i$, $y_c=t_i$. The ray with the shortest distance is $\boldsymbol{r}_i=(c_i/a_i, d_i/a_i, u_r, v_r) = (s_c, t_c, u_r, v_r)$ which coincides with $\boldsymbol{r}$ on the ***u-v*** plane as they have the same ***u-v*** coordinates. Since all selected rays pass through (u_r, v_r, z_{uv}), their costs are really dependent on their locations on the ***s-t*** plane. This is the basis for the quadrilinear interpolation adopted in [7].

3.2 Incorporating the dynamic focal plane

An alternative cost function is to require additionally that the selected source ray have a direction similar to the virtual one. We therefore minimize

$$d^2 = \left\{(s_i - s_r)^2 + (t_i - t_r)^2 + (u_i - u_r)^2 + (v_i - v_r)^2\right\} + \beta\left\{(s_i - s_r - u_i + u_r)^2 + (t_i - t_r - v_i + v_r)^2\right\} \qquad (5)$$

subject to $a_i s_i + b_i u_i = c_i$ and $a_i t_i + b_i v_i = d_i$. The first term accounts for the light field distance as before. The second term accounts for orientation difference in world space. For example, it becomes zero when $\boldsymbol{r}_i$ and $\boldsymbol{r}$ are parallel. β is the balancing factor. Solving (5) results in

$$\boldsymbol{r}_i = \begin{bmatrix} \dfrac{b_i^2\left((1+\beta)s_r - \beta u_r\right) - a_i b_i\left((1+\beta)u_r - \beta s_r\right) + c_i\left((1+\beta)a_i + \beta b_i\right)}{\beta(a_i+b_i)^2 + a_i^2 + b_i^2} \\ \dfrac{b_i^2\left((1+\beta)t_r - \beta v_r\right) - a_i b_i\left((1+\beta)v_r - \beta t_r\right) + d_i\left((1+\beta)a_i + \beta b_i\right)}{\beta(a_i+b_i)^2 + a_i^2 + b_i^2} \\ \dfrac{a_i^2\left((1+\beta)u_r - \beta s_r\right) - a_i b_i\left((1+\beta)s_r - \beta u_r\right) + c_i\left((1+\beta)b_i + \beta a_i\right)}{\beta(a_i+b_i)^2 + a_i^2 + b_i^2} \\ \dfrac{a_i^2\left((1+\beta)v_r - \beta t_r\right) - a_i b_i\left((1+\beta)t_r - \beta v_r\right) + d_i\left((1+\beta)b_i + \beta a_i\right)}{\beta(a_i+b_i)^2 + a_i^2 + b_i^2} \end{bmatrix}$$

and

$$d_{\min}^2 = \frac{(1+2\beta)\left((a_i s_r + b_i u_r - c_i)^2 + (a_i t_r + b_i v_r - d_i)^2\right)}{2\left(\beta(a_i+b_i)^2 + a_i^2 + b_i^2\right)}. \qquad (6)$$

It can be shown that $\boldsymbol{r}_i$ and $\boldsymbol{r}$ actually intersect. Let us analyze the role of β by looking at the depth of the intersection, which is

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

First, since each CHL contributes an equation like (5), multiple balancing factors are introduced. All of them become linearly related if the selected source rays $\boldsymbol{r}_i$ and $\boldsymbol{r}$ intersect at the same depth. Thus only one of these factors needs to be designated. The rest can be automatically computed. Second, β can be chosen 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, the plane at $z(\beta)$ performs a similar role to the dynamic focal plane of [10] 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 drawn previously. Fourth, $z(\infty)=\infty$. This is equivalent to say that the selected source ray is parallel to the virtual one, a result of assigning the orientation term infinite weight.

3.3 Implementation as projective texture mapping

The fact that the virtual rays and the selected source rays intersect at the dynamic focal plane indicates that the virtual camera is related to the source ones by homographies induced by the plane. Following is a brief derivation.

Denote the virtual camera projection matrix as C. A point X on the plane $z=z_\beta$ is projected to

$$\boldsymbol{a} = CX = C\begin{bmatrix} x \\ y \\ z_\beta \\ 1 \end{bmatrix} = \left(C\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z_\beta \\ 0 & 0 & 1 \end{bmatrix}\right)\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H_o(z_\beta)\boldsymbol{x}.$$

Similarly, in a source camera C_i, X is projected to $\boldsymbol{b}=H_i\,\boldsymbol{x}$. Therefore $\boldsymbol{a} = H_o H_i^{-1}\boldsymbol{b} = H(z_\beta)\boldsymbol{b}$. The 3×3 matrix $H(z_\beta)$ is the aforementioned homography.

Consequently, the virtual image can be synthesized via direct projective texture mapping and alpha blending. The blending weights can be computed from the inverse 4-D distance, or any other methods such as [2],[14]. In fact, any 3-D plane induces a homography relating two cameras [5]. Thus any plane can serve as a dynamic focal plane.

4. LIGHT FIELD RENDERING USING GHL

Due to the symmetric role of GHL with respect to CHL, when the scene consists of only colored points, light field rendering can be implemented using GHL's. This is equivalent to point cloud rendering in the 3-D space. Currently, our 3-D data are obtained from either laser scan or stereovision. In both cases, scene points have single colors. We thus assume in the following that GHL's are homogeneous in color, and leave dealing with viewpoint dependent colors as a future work.

In its simplest form, GHL rendering can be described as:

```
Convert all scene points into GHL's
For each virtual ray r
   Compute its 4-D coordinate [$s_r$, $t_r$, $u_r$, $v_r$]
   For each GHL g
      Compute $d_{\min}$ according to (4)
      Choose g's with minimum cost
      Blend colors of the just chosen g's
```

This naive implementation however will not handle occlusion correctly. In Figure 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 Figure 2(a), we would like the ray to have the color of the foreground. However, as depicted in Figure 2(b), our selection criterion wrongly assigns the background color to the ray. This is the *background leakage* problem. For Figure 2(c), the ray should have the background color. But according to Figure 2(d), it is closer to the foreground in the light field. This is the *foreground expansion* problem.

Our solution to the above problems is based on two observations. First, in general, the dense point cloud in the vicinity of a ray is clustered around the same depth. This is equivalent to say, without creating a polygon mesh of the object surfaces, we agree that a 2-D step waveform of piecewise constant depth is still a reasonable approximation. As a result, GHL's are clustered according to their slope (recall $k=(z-z_{uv})/(z-z_{st})$). The second observation is that if a ray *passes through* a cluster, it is surrounded by GHL's in the vicinity. 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 slope (also minimum depth) and at the same time is surrounding the virtual ray.

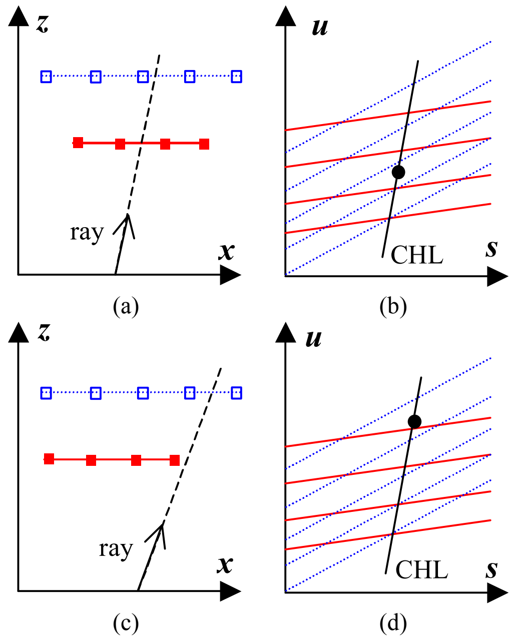


Figure 2: (a) A ray goes through both foreground and background. (b) The distance criterion selects the wrong GHL. (c) A ray goes by the foreground. (d) The distance criterion again selects the wrong GHL.

4.1 Determining ray vicinity

Conceptually, the light field vicinity of a virtual ray $\boldsymbol{r}$, denoted as $G(\boldsymbol{r})$, is a circular region on the virtual camera hyperline centered at $\boldsymbol{r}$. We are interested in the set of GHL's that intersect with the region. Let V_s be a vicinity threshold, then, $G(\boldsymbol{r})=\{\boldsymbol{g} \mid d_{\min}(\boldsymbol{g}, \boldsymbol{r}) < V_s\}$ where $d_{\min}(\boldsymbol{g}, \boldsymbol{r})$ is the shortest 4-D distance as defined in (4).

In implementation, $G(\boldsymbol{r})$ is computed in a pre-rendering stage. A GHL $\boldsymbol{g}$ is registered to $G(\boldsymbol{r})$ if $d_{\min}(\boldsymbol{g}, \boldsymbol{r})<V_s$. Deciding a proper value for V_s requires some trial-and-error. One hint is that it must be big enough so that some foreground GHL's are included in $G(\boldsymbol{r})$.

4.2 GHL clustering based on slope

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 GHL cluster. The observation that a cluster is around the same slope implies that, for a given dataset, there exists a threshold V_d such that the slope variance of each cluster is no greater than V_d. More formally, a slope based clustering of $G(\boldsymbol{r})$, denoted as $\boldsymbol{C}(G(\boldsymbol{r}))$, 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 slope of G_i;
2) Enclosure: for $\forall \boldsymbol{g}$, if $min(G_i) \leq slope(\boldsymbol{g}) \leq max(G_i)$, then $\boldsymbol{g} \in G_i$;
3) Maximum: for $\forall \boldsymbol{g} \in G(\boldsymbol{r})$, but $\boldsymbol{g} \notin G_i$, $max(G_i \cup \{\boldsymbol{g}\})-min(G_i \cup \{\boldsymbol{g}\}) > V_d$.

The uniqueness of such a clustering is guaranteed by the above constraints provided that one exists at all for the chosen V_d. In practice, it is not hard to find an appropriate V_d because in most cases the clusters are well separated, as shown in Figure 2.

A quick way of computing $\boldsymbol{C}(G(\boldsymbol{r}))$ is to sort all GHL's according to their slope in a pre-processing stage; then, in ascending order, each $\boldsymbol{g} \in G(\boldsymbol{r})$ belongs to a cluster G_i either if G_i is empty or $slope(\boldsymbol{g})-slope(\boldsymbol{g}_0)<V_d$ where $\boldsymbol{g}_0$ is the first GHL of G_i.

4.3 Opacity of a GHL cluster

We now define a boolean function $opacity(G_i)$ that indicates whether or not all $\boldsymbol{g}_j \in G_i \in \boldsymbol{C}(G(\boldsymbol{r}))$ are on one side of $\boldsymbol{r}$. Let us switch our minds to the world space, and think of $\boldsymbol{r}$ and $\boldsymbol{g}_j$ as 3-D entities. The necessary and sufficient condition for all $\boldsymbol{g}_j \in G_i$ to be on one side of $\boldsymbol{r}$ is that there exists a plane containing $\boldsymbol{r}$ such that all $\boldsymbol{g}_j$'s are on side of this plane. This says that there exists a wedge whose ridge is $\boldsymbol{r}$, covering all $\boldsymbol{g}_j$'s. The existence of such a wedge can be examined on a plane parallel to the parameterization planes. This is depicted in Figure 3 which is a top view of Figure 2. In Figure 3, $\boldsymbol{d}$ is the 2-D vector starting from $\boldsymbol{r}$, pointing to $\boldsymbol{g}_j$, constrained to the plane containing $\boldsymbol{g}_j$ and parallel to the 2PP. We thus define

$$opacity(G_i) = \begin{cases} true & \text{if the } \boldsymbol{d} \text{ vectors span} \\ & \text{greater than } 180^\circ; \\ false & \text{otherwise.} \end{cases}$$

When $opacity(G_i)$ is true, we say the cluster is opaque, meaning that the color of $\boldsymbol{r}$ should come from G_i because it is surrounded by GHL's in G_i. Otherwise, the cluster is transparent, meaning the color of $\boldsymbol{r}$ should come from another cluster farther back.

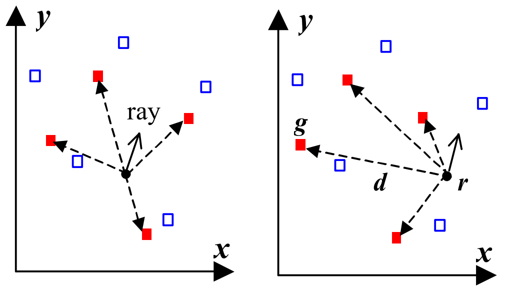


Figure 3: Understanding the opacity function in world space. Left: both clusters are opaque. The closer one is selected. Right: foreground cluster is transparent. The background one is selected.

It turns out that if $\boldsymbol{g}_j=(a_g, b_g, c_g, d_g)$ and $\boldsymbol{r}=(s_r, t_r, u_r, v_r)$, then $\boldsymbol{d}=(a_g s_r+b_g u_r-c_g,\ a_g t_r+b_g v_r-d_g)$. Notice that both components of $\boldsymbol{d}$ appear in $d_{\min}(\boldsymbol{g}_j, \boldsymbol{r})$. Therefore no extra computation is required for $\boldsymbol{d}$. This also reveals a relationship between the 4-D hyper point/hyperline distance and the constrained 3-D line/point distance. In fact, there exists a 4-D cone that covers the 4-D lines passing the hyper point $\boldsymbol{r}$ and perpendicular to the hyperline $\boldsymbol{g}_j$'s. This 4-D cone and the previous 3-D wedge are equivalent constructs. Below is the pseudo code of the modified GHL rendering algorithm:

```
Convert all scene points into GHL's
Sort all GHL's in ascending order of slope.
Virtual view rendering starts:
    Compute $G(\boldsymbol{r})$ for all virtual rays.
    For each virtual ray $\boldsymbol{r}$
        Compute $\boldsymbol{C}(G(\boldsymbol{r}))$.
        Find the first $G_i \in \boldsymbol{C}(G(\boldsymbol{r}))$ such that
        $opacity(G_i)$ is true.
        Blend colors of $\boldsymbol{g} \in G_i$, weighted by $1/d_{\min}(\boldsymbol{g}, \boldsymbol{r})$.
```

5. 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 parameterization planes are $z_{st} = 4$ and $z_{uv} = 1$. Results are displayed in Figure 4. Observe that changing β in effect rotates the black line in the first column. This is like guessing the slope of the GHL corresponding to a point on the central ray (or its depth). 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. The light field slice shown in the left column is similar to the Epipolar-Plane-Image (EPI), a common tool used in computer vision for motion analysis [1].

In the second example, twenty-five cameras spanning a 45-degree arc are equally placed around a teapot. β is set so that the focal plane is located at the teapot body. For images in the top row of Figure 5, each pixel is blended from 6 input rays from 6 closest CHL's. Notice how occlusion caused by the sprout appears correct although slight blurring is observed along the edges. For the lower left image, each pixel is copied from the single best CHL (different pixels may be copied from different CHL's). As a result, it looks sharp. For comparison, a ground truth image is displayed at lower right.

Figure 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 selected and 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 projective texture mapping, we have reached 15fps average frame rate at the 512×512 resolution on a 2Gz Pentium IV with a NVIDA GeForce2 MX graphics card.

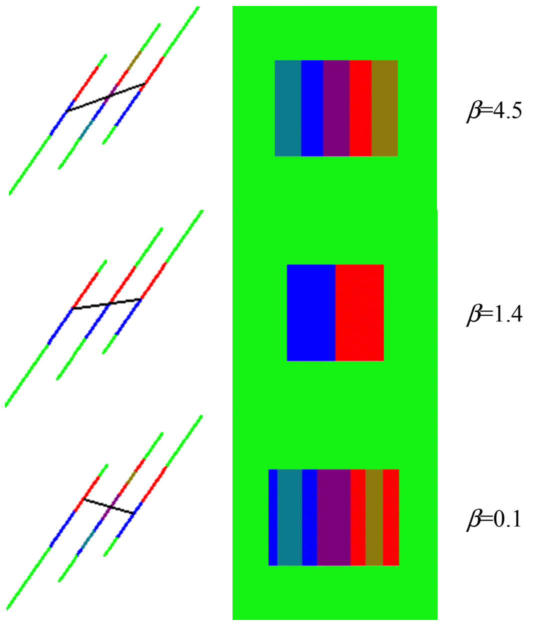


Figure 4: Illustration of CHL-based rendering and the effect of β. The first column shows a slice of the light field at $t=v=0$, which consists of three tilted CHL's. The central one comes from the virtual camera. The black line indicates rays selected to synthesize the central pixel. The second column shows the synthesized image. ***(This figure is reproduced in color on page 210).***

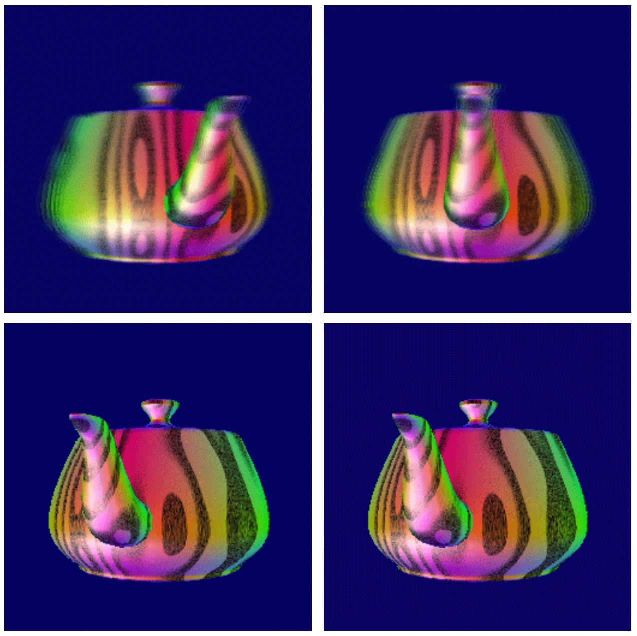
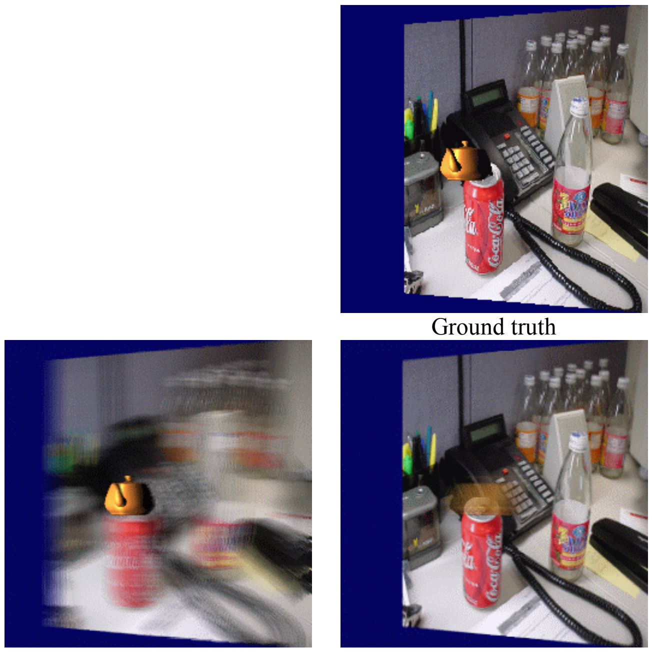


Figure 5: CHL-based rendering of a teapot. ***(This figure is reproduced in color on page 210).***



Ground truth

Focus at foreground Focus at background

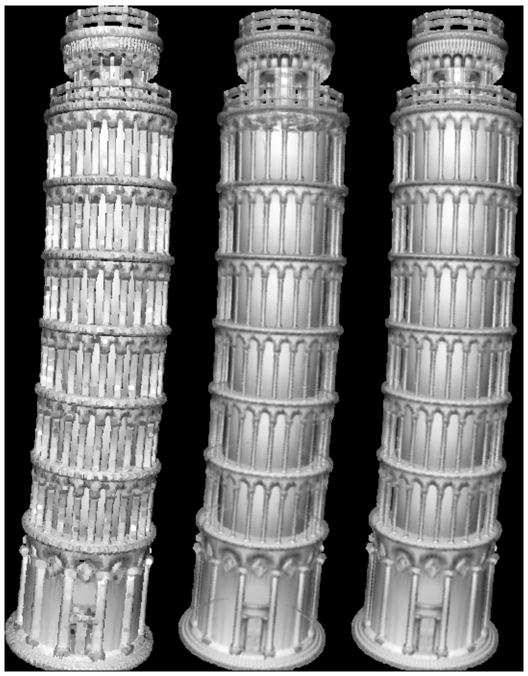
Figure 6: Dynamic focal plane by adjusting β. *(This figure is reproduced in color on page 210).*

Next, results from GHL-based rendering are presented. Figure 7 depicts the use of the slope threshold. If V_d=0 as in (a), the solution is aliased. This happens because each cluster contains only one GHL. If V_d is too large (b), two clusters are mixed. As a result, we can see through the tower. (Notice the outline of the base circle continuing behind the solid tower.) Using a proper V_d as in (c), the image looks smooth and occlusion is handled.

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

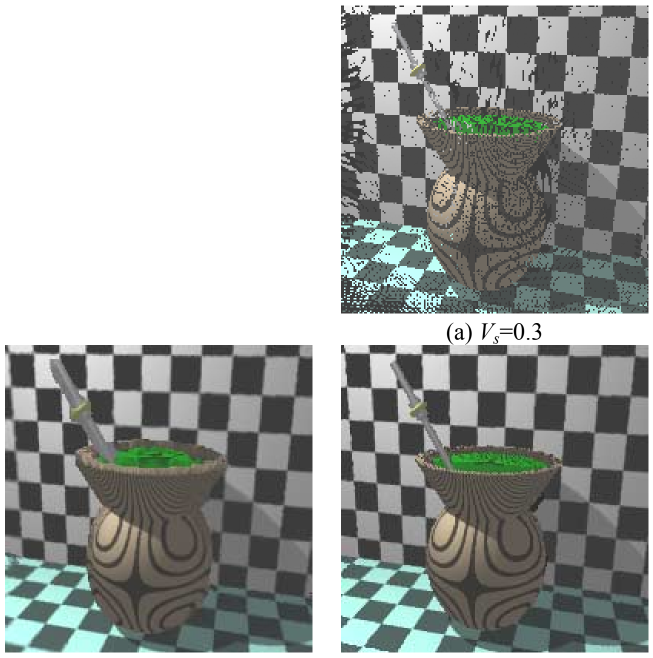
Figure 9 compares rendering results of the "car wreck scene" between a polygonal renderer and our GHL renderer (range data courtesy of University of North Carolina at Chapel Hill IBR group). Not all the differences are noticeable in the printed images, so a few are pointed out here. On the left car, notice the window, the hubcap and the seam between the front and back doors. On the right car, notice the areas around the real wheel and the ridge on the hood. In all these areas, the GHL rendered image looks smoother and more visually pleasant.

Compared to CHL rendering, GHL rendering is slower due to the occlusion handling. On a 2Gz Pentium IV, current frame rate is about 0.2~2fps at the 512×512 resolution depending on the number of hyperlines used.



(a) V_d =0 (b) V_d =100 (c) V_d =3

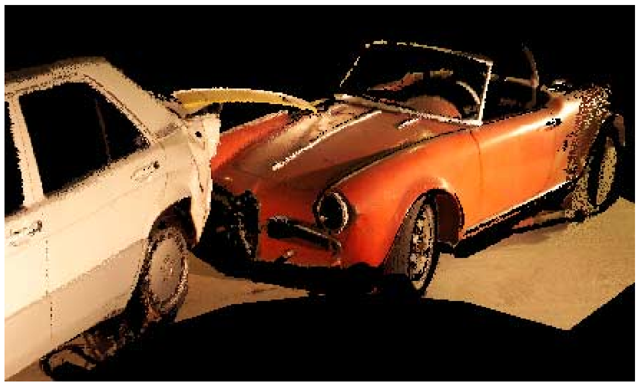
Figure 7: The "Pisa Tower", using 6 images and 467260 GHL's.



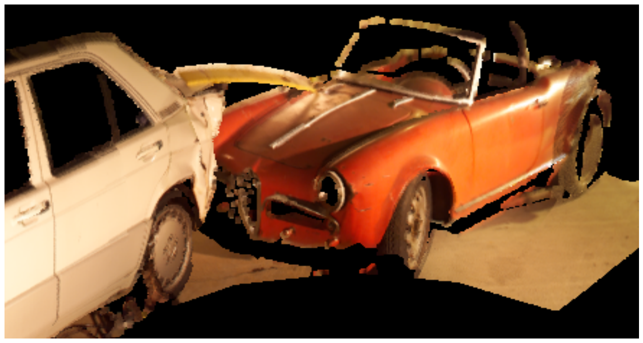
(a) V_s=0.3

(b) V_s=3, without opacity (c) V_s=3, with opacity

Figure 8: The "Mate", using 4 images (300×300) and their depth maps.



Polygon-based rendering



GHL-based rendering

Figure 9: The "car wreck scene", using 9 images (864×576) and their range data. ***(This figure is reproduced in color on page 210).***

6. CONCLUSIONS

We have presented a new light field rendering framework that combines the benefits of many existing approaches. It is based on a dual representation of the world space such that 3-D points and lines become 4-D hyperlines and hyper points respectively. We show that our algebraic ray selection criterion is in fact geometrical meaningful. For instance, the balancing factor in the cost function acts as a dynamic focal plane. The latter induces a homography which makes it possible to implement CHL rendering as projective texture mapping. The 4-D hyper point/hyperline distance is related to the 3-D line/point distance constrained to planes that are parallel to the parameterization planes. This allows us addressing the foreground expansion problem in GHL rendering. If we view CHL rendering as using one plane for texture mapping, and GHL as using multiple planes (each passing through a scene point, all parallel to the parameterization planes), we see that the real difference between the two is the number of depth planes used. This understanding opens up rooms for future hybrid algorithms that optimally explore the continuous spectrum between light-field based and geometry-based methods, and potentially address the problem when the geometrical data is incomplete and/or inaccurate.

7. ACKNOWLEDGMENT

The authors would like to thank the University of North Carolina IBR Group at Chapel Hill for their reading room and car wreck models. Funding was provided in part by the "Digital Media Innovation Program (DIMI)" from the University of California. Hofsetz is sponsored by UNISINOS – Universidade do Vale do Rio dos Sinos and CAPES – Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

8. REFERENCES

[1] R.C. Bolles, H.H. Baker, and D.H. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," International Journal of Computer Vision, 1(1):7-55, 1987.

[2] C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, "Unstructured Lumigraph Rendering," SIGGRAPH 2001.

[3] E. Camahort, A. Lerios, D. Fussell, "Uniformly Sampled Light Fields," Eurographics Rendering Workshop '98, 117-130, 1998.

[4] Q. Chen and G. Medioni, "Image Synthesis from A Sparse Set of Views," IEEE Visualization' 97, 269-275, 1997.

[5] R. Hartley and A. Zisserman, "Multiple View Geometry," Cambridge University Press, 2000.

[6] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. Van Gool, "Plenoptic modeling and rendering from image sequences taken by a hand-held camera," Mustererkennung 1999, 94-101, 1999.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The Lumigraph," SIGGRAPH 96, 43-54.

[8] J.P. Grossman, William J. Dally, "Point Sample Rendering," Eurographics Rendering Workshop'98, 1-12, 1998.

[9] X. Gu, S. J. Gortler, M. F. Cohen, "Polyhedral Geometry and the Two-Plane Parameterization," Eurographics Rendering Workshop'97, 181-192, 1997.

[10] A. Isaksen, L. McMillan, S. J. Gortler, "Dynamically Reparameterized Light Field," SIGGRAPH 2000, 297-306.

[11] 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.

[12] M. Levoy and P. Hanrahan, "Light Field Rendering," SIGGRAPH 96, 31-42, 1996.

[13] H. Pfister, M. Zwicker, J. van Baar, M. Cross, "Surfels: Surface Elements as Rendering Primitives," SIGGRRAPH 2000, 335-342, 2001.

[14] K.Pulli, M.Cohen, T.Duchamp, H.Hoppe, L.Shapiro and W.Stuetzle, "View-based rendering: Visualizing real objects from scanned range and color data," Eurographics Rendering Workshop'97, 23-34, 1997.

[15] S. Rusinkiewicz, M. Levoy, "Qsplat: A Multiresolution Point Rendering System for Large Meshes," SIGGRAPH 2000, 343-352, 2000.

[16] H. Schirmacher, C. Vogelgsang, H.-P. Seidel, G. Greiner, "Efficient Free Form Light Field Rendering," Vision, Modeling, and Visualization 2001, 249-256, 2001.

[17] H. Schirmacher, L. Ming, H.-P. Seidel, "On-the-Fly Processing of Generalized Lumigraphs," Eurographics 2001, 20(3):C165-C173, 2001.