Uniform multiresolution Light Fields

Ales V. Michtchenko

Department of Computer Science, Moscow State University

Moscow, Russia

Abstract

This paper presents two different techniques of virtual reality rendering: The mesh-intersecting light fields and the deformation-based light fields.

Both methods are shown to have a number of advantages over the existing light field rendering approaches.

Keywords: *Light field, Virtual reality, Mesh, Specular, Reflection.*

1. INTRODUCTION

1.1 Light-field as an array of view-rays

Image-based and Geometry-based rendering (Light Fields and Ray-tracing or Radiosity) are the two extreme approaches to scene representation. They are using extensively different resources (memory and CPU, correspondingly) and the most of research was concentrated on reducing the usage of CPU for geometry-based and memory for image-based rendering.

There was a number of works on reducing a Light-Field array dimension. The most popular approach is to consider Light Field not as a 3D array of images, but as a 5D array of rays. Since in a free space colour and intensity of a ray is constant, it is possible to reduce its index-dimension from 5D to 4D, that corresponds to a set of lines in a 3D space.

Light Field dimension reduction, based on a ray-representation can be used successfully for a real-time rendering ([1],[2],[3]), but it has a certain difficulties with uniform parameterisation of line-space.

In this paper we are proposing two ways of constructing the uniformly spaced Light Fields, with a small memory usage: The first approach (see section 2) is to connect a Light Field with a set of view-points, and the second approach (see section 3) is to render, using both geometry-based and image-based elements.

1.2 Continuity and uniformity of surface-intersecting light-fields.

1.2.1 Using different surfaces for view-lines parameterisation

Parameterisation of lines was usually performed by surrounding a scene with a simple surface (cube, polyhedron or sphere) and indexing each line by two point of its intersection with the surface.

The first surface of that kind was a pair of planes, containing a scene between them. This configuration, called the light slab [1] is still one of the most widely used, due to its simplicity and ease of intersection computation.

The major drawback of this configuration is that a camera is restricted to observe the scene only from outside a light slab. This can be used, for example, for a relief rendering, but unsuitable for sculptures. Another disadvantage of the light slab is that the density of rays ($1/\cos\{\vec{n}\wedge\vec{d}\}$) grows with $|\vec{d}|$ and reaches infinity for the side-view directions ($\vec{n}\perp\vec{d}$), where $\vec{n}$ is a norm to a light slab, $\vec{d}$ is a direction of view (a vector connecting a light slab point with the camera).

Later, the intersection of several light slabs was proposed in [2]. This allows camera to observe a scene from arbitrary position, but introduces a continuity problem: due to angles between faces of resulting polyhedron ($\vec{n_i}\wedge\vec{n_{i+1}}\neq 0$), the brightness of the picture will seem non-continuos on the edges.

This continuity problem was solved in [3] by introducing a smooth surface (sphere). Although rays are still biased towards visible boarders of a sphere ($1/\cos\{\vec{n}\wedge\vec{d}\}=\infty$, if $\vec{d}$ is a tangent to the sphere), density discontinuity is avoided.

1.2.2 Angular and image-plane non-uniformity of surface-intersecting light-fields and uniforming approaches.

It can be shown, however, that surface-intersecting light-fields fail to provide both angular uniformity of rays for arbitrary camera position and uniformity of intersection-

points of rays with arbitrary image plane. Therefore, the brightness of a picture will seem near-uniform only for some particular scene and surface configurations.

The best examples of angular uniformity are using sphere for inward looking from the vicinity of its centre and using cylinder for inward looking from vicinity of it's axis in a view-direction $\vec{d}$ almost perpendicular to it. These scene-surface configurations may be successfully used for virtual walk through a set of rooms and corridors (virtual interior design or virtual picture gallery), but limited to inward-looking directions.

The uniformity of intersection points on the image plane will be provided only for light-slab, parallel to the image plane. This brings camera back to outside the light slab.

A number of approaches to making a ray-distribution (and hence, the brightness of a picture) uniform may be proposed. It is possible, for each camera position $\vec{d}$, to render just a part (fraction $\approx \cos\{\vec{n} \wedge \vec{d}\}$) of rays to compensate the density growth. Another way is to assign a weight ($\cos\{\vec{n} \wedge \vec{d}\}$) to each ray. In both cases it is necessary to calculate a trigonometric function, and, if surface is not a plane, it is necessary to calculate its norm as well.

In the next 2 sections we will represent two different ways of more radical change to ray-space geometry:

In section 2 we will discretise 3D space of viewpoints accordingly to discretisation of line-space. Thus, we will improve uniformity at the cost of ability to use arbitrary camera-positions. Advantage of this approach is the absence of an intersection surface, so this parameterisation is independent on scene's objects locations.

In section 3 we will improve angular uniformity of the line-space by introducing intermediate parallel projection and simultaneous depth coding.

2. MESH-INTERSECTING LIGHT FIELDS

2.1 The dual-uniformity problem.

Let us turn down an idea of uniform angular parameterisation of rays for any arbitrary camera location and choose a discrete set of viewpoints.

2.1.1 An example of dual-uniform sets and a general formulation of the dual-uniformity problem

An example of uniform set of viewpoints together with a uniform set of rays, issued from each viewpoint (dual-uniform sets) is a spherical-uniform bunch of rays, issued from the each node of the cubic mesh.

However, since tangent is not a rational function, these rays are unlikely to hit another nodes, other from the node they are issued from. This makes the size of a light field to grow as a product of the number of viewpoints and the number of view-directions. To avoid that, it is necessary to introduce a certain fraction of non-uniformity into either set of rays or set of a view-points, so that each ray will intersect a number of nodes, which will grow linearly with the total number of nodes.

Strictly speaking, we have to find a set of points together with a corresponding set of lines, so that both sets will be reasonably uniform in corresponding spaces and average number of points lying on each line will be maximised. This could be named a formulation of dual-uniformity problem in the general case. It is very difficult to solve, but since arbitrary set of points in 3D is not easy to store and manipulate, such a general solution is not, indeed, needed.

2.1.2 Approximations of the dual-uniformity problem

Let us, for implementation reasons, restrict ourselves to a regular, periodic mesh. We will conduct all our reasoning for the case of a regular cubic mesh, and its 2D analogy - square mesh.

To find an approximate solution for a dual-uniformity problem, we can fix one of the sets and find an optimal solution for another.

Below, we will use a ray-displacement approach to this problem and formulate an approximate dual-uniformity problem as follows: For a given set of points, to select a lines, which will go through as much nodes as possible, providing almost equal angles between the neighbour lines.

2.2 Sequences of vicinities, providing near-uniformity and multiresolution.

2.2.1 Formulation of the dual-uniformity problem for periodic meshes

For the line in the periodic mesh, going through the nearest node is equivalent to going through the maximum number of nodes. Hence, the above mentioned task of intersection maximisation with restriction of near-uniformity boils down to defining a metric ρ in the mesh and drawing a set of angular equidistant lines through all the nearest neighbours.

It is natural to introduce a multiresolution at this stage and to formulate the dual-uniformity problem as follows: For a given mesh (regular cubic in our case) to define a metric ρ and to find a corresponding sequence of circles $R_1 \le R_2 \le \ldots$, so that: (1) for any circle in the sequence, the angles between radial lines, going through

the centre and nodes on the circle will be almost equal (uniformity) and (2) for any two subsequent circles $R_i \le R_{i+1}$, lines, going through the nodes on a smaller one, go through the nodes of bigger one as well (multiresolution). The pair $\{R_i; \rho\}$ will be called a solution to the dual-uniformity problem, if it satisfies conditions (1) and (2).

To provide a multiresolution (2), we will restrict ourselves to a geometric sequence and choose the smallest possible common ratio (two), so that $R_i = \Delta \cdot 2^{i+1}$, where Δ is a distance between neighbours in a mesh (the mesh's step).

To measure the angular uniformity of the view-rays (1), we are proposing two ratios: $\frac{\theta_{\max}}{\theta_{\min}}$ and $\frac{\overline{\theta_{\max}}}{\overline{\theta_{\min}}} = \frac{density_{\min}}{density_{\max}}$, where an operation $\overline{(...)}$ denotes an averaging over a small solid angle around a view-direction $\vec{d}$. It can be shown, that $\overline{(...)} = (...)$ in 2D case, because of the only one polar angle.

For a limit case i>>1 ($\Delta << R_i$ and $\theta = \sin\theta = \tan\theta$) two approximate solutions for the dual uniformity problem have been found:

$$\{R_i = \Delta \cdot 2^{i+1}; \rho = \rho_1 \equiv \sum_{i=1}^{dimension} |r_i^{/} - r_i^{//}|\} \quad \text{and}$$

$$\{R_i = \Delta \cdot 2^{i+1}; \rho = \rho_\infty \equiv \underset{i}{MAX} |r_i^{/} - r_i^{//}|\}.$$

It is possible to show, that in 2D case both solutions have the non-uniformity ratio $\frac{\theta_{\max}}{\theta_{\min}}$=2. As for a cubic mesh (3D case) $\frac{\theta_{\max}}{\theta_{\min}} = \sqrt{6}$ for $\{\Delta \cdot 2^{i+1}, \rho_1\}$. For $\{\Delta \cdot 2^{i+1}, \rho_\infty\}$ the corresponding formula has a more complicated form, but can be shown not to exceed 3.

The directions of maximum and minimum density are as follows: $\vec{d}(\overline{\theta_{\max}})$ for $\{\Delta \cdot 2^{i+1}, \rho_\infty\}$ and $\vec{d}(\overline{\theta_{\min}})$ for $\{\Delta \cdot 2^{i+1}, \rho_1\}$ are the coordinate axes $\pm\vec{e}_i$, i=x,y,z.

$\vec{d}(\overline{\theta_{\min}})$ for $\{\Delta \cdot 2^{i+1}, \rho_\infty\}$ and $\vec{d}(\overline{\theta_{\max}})$ for $\{\Delta \cdot 2^{i+1}, \rho_1\}$ are the octant bisectors $\frac{\pm\vec{e}_i \pm\vec{e}_j \pm\vec{e}_k}{3}$.

Expresions for $\frac{\overline{\theta_{\max}}}{\overline{\theta_{\min}}} = \frac{density_{\min}}{density_{\max}}$ are more cumbersome due to directional anisotropy, but it can be shown, that

$\frac{\overline{\theta_{\max}}}{\overline{\theta_{\min}}} = \frac{density_{\min}}{density_{\max}} < \frac{\theta_{\max}}{\theta_{\min}}$ for both solutions.

2.3 Advantages of mesh-intersecting light fields.

Besides an almost uniform distribution of lines along all view-directions, this parameterisation of light-field lines has the other two advantages: independence on scene location (and complexity as well) and multiresolution.

2.3.1 Independence on location and complexity of the scene

Since no object-surrounding surfaces were used, this method does not become more complex with addition of new objects to the scene.

Moreover, because of tight interconnection between viewpoints (nodes of mesh) and view-directions, it is easier to introduce a different colour and intensity values to different parts of the ray. To illustrate this, imagine, a view-line, intersecting an object surface at N different points (for example, the view-line $\overrightarrow{OZ}$ on fig.1 intersects a cylindrical object twice). There will be 2N corresponding colours for that line, and choosing among them depends on camera locations and on two possible camera orientations (positive and negative in respect to directional vector of a line).

The complexity of the data structure of surface-intersecting light fields grows with number N, since there is no natural connection between the camera location and the view-line. The mesh-intersecting light fields, on the contrary, connect view-lines and viewpoints by a simple periodic dependence (see 2.2), and, hence, it is easy to cluster viewpoints on each line by their location in respect to the object.

Therefore, handling multintersections does not require restructuring of the mesh-intersecting light fields and, hence, this method of virtual reality rendering is capable of robust handling not only a single object fly-around's, but an arbitrary walk through a complex environment, such as museums, botanical gardens, crowds of people.

2.3.2 Multiresolution

We introduced a multiresolution of light fields, which is very important in virtual video compression and communication. This enables communicating with virtual-reality rendering server via finite bandwidth

communication channels, and allows applying to light-fields all the multiresolution techniques. For example, for virtual reality helmet rendering, we can increase resolution in the direction of sight and decrease it in the side views (foveation, see [4]).

Multiresolution of view-directions can be used together with a multiresolution of view-positions (using multiresolution cubic mesh). This will introduce a trade-off between positional and directional discretising, which will be useful then applying to different virtual realities.

For example, for a virtual museum rendering, directional resolution is, probably, more important (since it is more important to see fine details of exhibits, than to see them from a lot of close viewpoints). For a computer games rendering, on the contrary, the density of available positions has to be sufficient to support a smooth dynamics of the game.

3. DEFORMATION-BASED LIGHT-FIELDS

This rendering technique is a compromise between geometry-based and image-based rendering, and, therefore, it reduces the dimension of a light field array (memory usage) at the cost of increasing intermediate computations (CPU load). It is based on exact intermediate (average) images, which are rendered using a perfectly uniform set of view-directions and contain all the precise illumination effects. These images are then deformed to achieve an approximation for an image from the given camera position.

3.1 Images clustering

Let us try to cluster images from different viewpoints and approximate them by an average image for the entire cluster.

3.1.1 Clustering as a minimisation

For a certain number of images (the certain volume in a camera-position space), clustering is more advantageous, if a square of a surface, containing this volume in itself, is minimised.

It can be shown, that, then minimising the surface/volume ratio, the close views will be more likely to belong to the same cluster. That means, for example, if rendering stereo views, dual images will belong to the same cluster, or, then rendering a virtual video, the subsequent shots will tend to remain in the same cluster. Since the change of cluster requires more computation, clustering with minimal surface/volume ratio is more advantageous.

3.1.2 Platonic solids case

Let us return to the notion of a light-slab and consider a platonic polyhedron, containing an object (scene). Rays, coming from the centre of a polyhedron through its edges, divide an outer space into N cones, where N is a number of polyhedron's faces (see fig.1 for one of cones).

We will use a dodecahedron, since it's faces (pentagons) have the biggest square/perimeter ratio and, hence, the corresponding cones have the biggest volume/surface ratio.

3.2 The cluster's average image as a parallel projection

Let us choose an average image for each cluster (for all the views from the inside of the corresponding cone) to be a parallel projection on the plane, containing the corresponding face of dodecahedron (see fig. 1).

3.2.1 Representativity of the parallel projection

We will assume, that there is no point that can be seen from some camera position inside the cone, but are not projected on the corresponding plane (representativity of an average image).

Otherwise, we can change this condition to the stronger one: object has a spherical topology and any line, perpendicular to any of projection planes, intersects the object's surface two or less times, so the backprojection is unique. Let us define the depth of a pixel (x, y) on the average image as a distance $h(x, y)$ between pixel and it's backprojection (see fig.1). We will discuss below, how a depth of the pixels can be used to deform an image from the average one into the one needed.

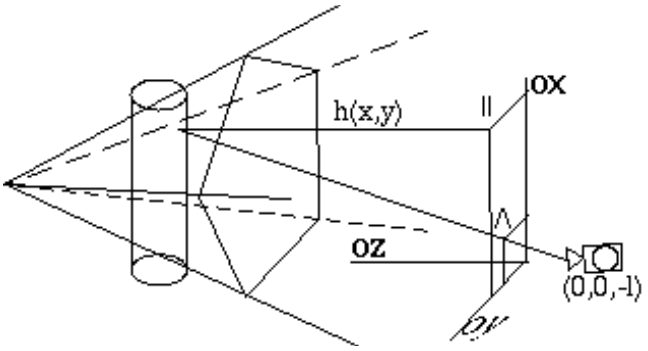

Figure 1: Parallel-to-perspective image deformation

3.2.2 The drawbacks of the parallel projections.

Parallel projections were used for light-field parameterisation (see, for example, [3]), because they provide a natural separation of 4D light-field index space into two 2D spaces: direction-of-projection space (position of a plane) and position-on-a-plane space.

Authors of [3] used this to avoid a non-uniformity of line-space, since, if chosen uniformly on a projection plane, parallel view-lines will remain equidistant in all the space.

However, this light field rendering on its own (without a depth-based deformation) has two serious drawbacks:

First, the difference between perspective projection (a view from a given camera position) and a parallel projection (an average image) grows then approaching an object and, also, then the depth of an object's surface $h(x, y)$ is significantly not a constant.

Second, all illumination effects, particularly resulting from a surface's deviation from Lambertian law (specular reflections, etc) will be placed unproperly.

3.2.3 Advantages of using the depth coding along with parallel projections

In the following chapters, we will introduce a simultaneous depth and light field coding for each projection plane. This will eliminate the two above-mentioned drawbacks and significantly reduce the overall size of the light-field array (instead of one 4D-array we will use only 12 2D-light-field arrays and 12 2D-depth arrays).

This approach is easier also from light-field acquiring point of view. If creating a light-field from the virtual reality rendering (ray-tracing or radiocity) it is necessary to render not a "continues" array of pictures, but just 12 average images, corresponding to 12 projection planes. If creating a light field from a set of real photographs, it is necessary to take 12 images of an object by 12 "infinitely" remote cameras. At the same time, it is necessary to use the binocular or trinocular views (see [6]) to estimate depth of an object's surface in respect to each plane.

3.3 The core of the average image deformation algorithm

Suppose now, that we have an average image with a depth, assigned to each pixel. Let camera belong to a certain cluster of views (let it be placed inside the corresponding cone). The rendering algorithm takes a corresponding parallel projection (2D-source image array) and copies it onto the screen (target array) by the following way: 1. Projecting a camera position point onto the projection plane (finding a focus of contraction - origin on a fig.1) and dividing a parallel projection image (an average image) into the four areas, that can not occlude each other (in a coordinate system, shown on fig.1, the camera position coordinates are $(0,0,-l)$), and hence, these areas are quadrants x>0,y>0; x<0,y>0; x<0,y<0 and x>0,y<0). 2. Scan each of the four areas to move pixels towards focus: $(x, y) \mapsto (x \cdot \frac{l}{l + h(x, y)}, y \cdot \frac{l}{l + h(x, y)})$, where l is the length or a perpendicular from a camera location onto the parallel projection plane, $h(x, y)$ is the depth of the pixel being moved (On a fig.1 pixel is moved from the position, marked "||" to the position, marked "/\").

This process is, in fact, a constructing a closer view of the object from a given infinitely remote view. This algorithm resolves all the problems, connected to perspective and occlusions. On the other hand, it neglects all the illumination effects. We will resolve these effects in the following sections: in 3.4 we will discuss how to reduce the brightness of the surface's element in accordance with the angle, at which it is viewed; in 3.5 problems with deviation from diffuse reflection will be discussed.

3.4 Changing the brightness for Lambertian part of reflection

Suppose, that reflection of an object's surface has no specular reflection component. For a given depth field $h(x, y)$ it is easy to calculate a norm field $\vec{n}(x, y) = \frac{\overrightarrow{\partial h/\partial x, \partial h/\partial y, -1}}{\sqrt{(\partial h/\partial x)^2 + (\partial h/\partial y)^2 + 1}}$ of the surface. This can be done within a rendering algorithm loop (see 3.3.) or as a pre-processing stage to it.

In a case considered, the amount of light $I_O(x, y, \vec{d})$, received in a certain camera position $\vec{d}$, fades only with decreasing of a solid angle, at which the surface element is viewed, hence $I_O \sim \cos\{(\overrightarrow{n(x, y)} \wedge \vec{d})\}$.

Therefore, the ratio of perspective projection intensity to a parallel projection intensity will be $\frac{I_\wedge}{I_\parallel}(x, y) = \frac{\cos\{(\overrightarrow{n(x, y)} \wedge \vec{d}_\wedge)}{\cos\{(\overrightarrow{n(x, y)} \wedge \vec{d}_\parallel)} = \frac{\overrightarrow{n(x, y)} \cdot \vec{d}_\wedge \cdot |\vec{d}_\parallel|}{\overrightarrow{n(x, y)} \cdot \vec{d}_\parallel \cdot |\vec{d}_\wedge|}$, where $\overrightarrow{d_\wedge} = \overrightarrow{(-x, -y, -h(x, y) - l)}$, $\overrightarrow{d_\parallel} = \overrightarrow{(0,0,-1)}$.

This ratio can be used as a fading multiplier for each pixel.

3.5 Changing the brightness for non-Lambertian part of reflection

Rendering of deviation from the Lambertian reflection is far more complicated. We will make the corresponding changes to the image in a three subsequent stages (to be able to stop refinement at some stage, if at a coarse resolution level or if the type of speck doesn't require

further processing): 1. Displacement of a specular reflection region due to the camera displacement. 2. Deformation of a bright region due to the change of surface's geometry at the old and at the new positions of the region. 3. Final brightness refinement by a reduced form of a ray-tracing.

In this paper we are concentrating on a specular reflection deviation. Other deviations, such as back-reflection (coarseness peak, see [5]) require simpler processing and are not discussed in this paper.

3.5.1 The first stage: The displacement of specular reflection regions due to camera displacement

If the surface is not purely Lambertian, then, besides multiplying the brightness of each pixel (x, y) by fading ratio $\frac{I_\wedge}{I_\parallel}(x, y)$, it is necessary to process the specular part of the brightness as well, namely: to calculate the centres $(x_\parallel, y_\parallel)$ and $(x_\wedge, y_\wedge)$ of the specular reflection regions for both parallel and perspective projections and, therefore, to move the specks on the average image plane.

Centres of the specks $(x_\parallel, y_\parallel)$ and $(x_\wedge, y_\wedge)$ can be calculated from direction to camera $\vec{d}$ and direction to a light source $\vec{s}$ within the norm calculating loop (see 3.4). Two physical conditions are used to mark the centres of the specks: (1): $(\vec{n}\cdot\vec{d})\cdot|\vec{s}|=(\vec{n}\cdot\vec{s})\cdot|\vec{d}|$ (angle of incidence is equal to angle of reflection) and (2): $([\vec{n}\times\vec{d}]\cdot\vec{s})=0$ (incident ray, reflected ray and a norm belong to a same plane - reflection plane). Pixels (x, y), satisfying these conditions for $\vec{d}=\vec{d_\parallel}=\overrightarrow{(0,0,-1)}$ will be marked as centres of parallel projection specks $(x_\parallel, y_\parallel)$; and pixels, satisfying these conditions for $\vec{d}=\vec{d_\wedge}=\overrightarrow{(-x,-y,-h(x,y)-l)}$ will be marked as centres of perspective projection specks $(x_\wedge, y_\wedge)$.

The speck displacement on the image plane is the operation of swapping the non-Lambertian intensities δI of pixels $(x_\parallel+\delta x, y_\parallel+\delta y)$ and $(x_\wedge+\delta x, y_\wedge+\delta y)$. Swapping may be performed gradually, for the expanding vicinity $\{|\delta x|+|\delta y|\}\to\infty$ and may be stopped, then the non-Lambertian δI intensities of the corresponding points will become the same.

The only difficulty is the separation of Lambertian I_0 and non-Lambertian δI intensities. For the first (displacement) approximation, we will assume, that the geometry of both $(x_\parallel, y_\parallel)$ and $(x_\wedge, y_\wedge)$ vicinities is nearly the same, and, hence, so is the Lambertian intensity.

For this approximation, it is possible to swap at overall intensities $I=I_0+\delta I$ of the above mentioned vicinities, and stop swapping, then their difference will fall below a certain threshold. The value of this threshold is, indeed, the error of the displacement approximation, but, if chosen too low, swapping may never stop due to the difference in Lambertian parts of reflection.

3.5.2 The second approximation: deformation of specks in accordance to surface geometry

The geometrical interpretation of the first approximation is the projection of the speck from the surface onto the image plane at the point $(x_\parallel, y_\parallel)$; moving this projection on the image plane to the point $(x_\wedge, y_\wedge)$; backprojection of the speck from the point $(x_\wedge, y_\wedge)$ back to the surface; and, finally, constructing the perspective view on it.

It is clear, that this sequence of steps do not lead to errors only if both the orientation and the curvature of surface are the same at the vicinities of points $(x_\parallel, y_\parallel)$ and $(x_\wedge, y_\wedge)$. Assuming, that the curvature difference between these two points is not sufficient to significantly change the form of backprojection, we will refine the backprojection according to the change in orientation and the form of the speck according to the change of curvature (see, also, [6]).

The backprojection refinement is, indeed, expansion of the speck, proportional to $1/\cos\vec{d_\parallel}\wedge\overrightarrow{n(x_\parallel, y_\parallel)}$ along the projection of $\vec{n}(x_\parallel, y_\parallel)$ on the image plane, and subsequent contraction, proportional to $\cos\vec{d_\parallel}\wedge\overrightarrow{n(x_\wedge, y_\wedge)}$ along the projection of $\vec{n}(x_\wedge, y_\wedge)$ on the image plane. Moving, again, means assigning the intensities:

$\delta I\ (x_\wedge+\exp^\parallel_X\cdot con^\wedge_X\cdot\delta x, y_\wedge+\exp^\parallel_Y\cdot con^\wedge_Y\cdot\delta y):=$

$\delta I\ (x_\wedge+\delta x, y_\wedge+\delta y)$, where $\overrightarrow{\exp^\parallel}$ and $\overrightarrow{con^\wedge}$ are the image plane projections of, correspondingly $\{1/\cos\vec{d_\parallel}\wedge\overrightarrow{n(x_\parallel, y_\parallel)}\}\vec{n}(x_\parallel, y_\parallel)$ and $\{\cos\vec{d_\parallel}\wedge\overrightarrow{n(x_\wedge, y_\wedge)}\}\vec{n}(x_\wedge, y_\wedge)$ vectors.

The second part of the refinement is the change of the speck form in accordance to the change of surface geometry.

Let us choose the local coordinate system, centred at the point $(x_\wedge, y_\wedge, h(x_\wedge, y_\wedge))$ on the surface as follows: $\overrightarrow{OZ} = \vec{n}(x_\wedge, y_\wedge)$, $\overrightarrow{OX}$ is the intersection of tangent plane and reflection plane, $\overrightarrow{OY}$ is perpendicular to the reflection plane.

Paper [6] shows, that curvatures $\frac{\partial \vec{n}}{\partial x}$ and $\frac{\partial \vec{n}}{\partial y}$ are connected to a non-Lambertian brightness δI of a pixel (x, y) as follows: $\delta I(x, y) = \delta I_{MAX} \frac{1}{\sigma^2}(\sigma^2 - 1 +$

$$+ \vec{d}\,\frac{\vec{r} + 2\cdot|\vec{r}|\cdot(x\cdot\frac{\partial \vec{n}}{\partial x} + y\cdot\frac{\partial \vec{n}}{\partial y})}{|\vec{d}|\cdot|\vec{r} + 2\cdot|\vec{r}|\,x\cdot\frac{\partial \vec{n}}{\partial x} + y\cdot\frac{\partial \vec{n}}{\partial y}|})$$

where $\vec{r} = (x - s_X, y, s_Z)$, $\vec{d}$ and $\vec{s}$ are the coordinates of the camera and the light source correspondingly.

This formula expresses $\delta I(x, y)$ as a function of surface curvatures. It can be used directly to calculate $\delta I(x, y)$ for a new curvature values or can be linearised for using as a multiplier (see [6] for details).

It is necessary to mention, that all these formulas use an approximation of small specks and are inapplicable if a surface geometry changes significantly across a specular reflectance region. There are two ways of dealing with big specks:

Using an integral equivalents for all the above mentioned formulas (we can use linear interpolation of curvatures to simplify the resulting integrals).

Direct calculation of brightness of each pixel (x, y) by the reduced form of a ray-tracing.

Although the second approach brings us from the image plane domain into the 3D-object space (see 3.5.4), it, in most cases, is still less computationally expensive. As we are dealing with only the non-Lambertian part of reflection δI, it is enough to trace the only two rays: specular reflected ray and the ray, reflected to the camera.

3.5.3 The order of rendering

Note, that, besides the above mentioned displacement and deformation, specks should suffer the same type of parallel-to-perspective contraction towards the focus (projection of a camera position on an image plane). Therefore, an overall sequence of rendering the non-Lambertian part of reflection δI is speck displacement, speck deformation and finally, the core algorithm applying: $(x_\parallel + \delta x, y_\parallel + \delta y) \mapsto (x_\wedge + \delta x, y_\wedge + \delta y)$

$\mapsto \frac{l}{l+h}(x_\wedge + \delta x, y_\wedge + \delta y)$.

All the three operations are of the swapping-intensity type and are performed on the image plane. Therefore, they are much faster than a direct ray-tracing.

The final, third stage of rendering, described in 3.5.4, provides the finest approximation for both Lambertian I_O and non-Lambertian δI parts of reflection. It involves a reduced form of a ray-tracing for an area, calculated at the previous stages.

3.5.4 The third approximation: the calculation of regions of specular reflection

At stages 1 and 2, we defined an area, containing all pixels, whose non-Lambertian reflection is increased or reduced then moving a camera from the infinity (parallel projection) to a rendering position (perspective projection). At the third stage, we are calculating the overall intensity $I_0 + \delta I$ for all pixels in this area.

Remembering where we stopped to expand the vicinity $\{|\delta x| + |\delta y|\} \to \infty$ of the centres $(x_\wedge, y_\wedge)$ and $(x_\parallel, y_\parallel)$ (see 3.5.2), we are reassigning the non-Lambertian intensity $\delta I(\delta x, \delta y)$ for all these pixels:

For each pixels $(x_\parallel + \delta x, y_\parallel + \delta y)$ and $(x_\wedge + \delta x, y_\wedge + \delta y)$ and their backprojections $(x_\parallel + \delta x, y_\parallel + \delta y, h(x_\parallel + \delta x, y_\parallel + \delta y))$ and $(x_\wedge + \delta x, y_\wedge + \delta y, h(x_\wedge + \delta x, y_\wedge + \delta y))$, we are calculating the specular reflected ray (vector) $\vec{r}$, using the two laws of specular reflection: (see 3.5.1): $(\vec{n}\cdot\vec{d})\cdot|\vec{s}| = (\vec{n}\cdot\vec{s})\cdot|\vec{d}|$ and $([\vec{n}\times\vec{r}]\cdot\vec{s}) = 0$.

An angle between $\vec{r}$ and $\vec{d}$ determines the amount of radiance, reflected to a camera direction $\vec{d}$.

Assuming the gaussian shape of the specular reflection peak $I = I_0 + \delta I = I_0 + \delta I_{MAX} \cdot \exp\{-(\vec{r} \wedge \vec{d})^2 / 2 \cdot \sigma^2\}$, we are calculating an overall intensity I for all pixels in the above-mentioned area.

The method, used in the third stage, can be applied to all deviations from Lambertian law. For example, we can use it to calculate the intensities δI, corresponding to a backreflection peak (as it was shown in [5], the amount of light, reflected back to the light source exceeds I_0 for coarse surfaces).

3.5.5 Multiresolution and specular reflection rendering

The each stage of specular reflection rendering can be put into a correspondence to a certain level of resolution, so to have a single multiresolution approach for deformation-based light fields.

However, a criterion of each stage to be sufficient is not developed yet. It depends on a lot of factors, such as surface geometry, shape and type of corresponding intensity peak. The development of such a criterion is a part of a future work.

4. CONCLUSIONS AND FUTURE WORK

4.1 Mesh-intersecting light fields

In this section we developed an approach to a virtual reality rendering, based on a close interconnection between a view directions and a view positions. It provides multiresolution and brightness uniformity of a resulting picture more than known surface-intersecting rendering.

It is based on a solution of the dual-uniformity problem, which was solved (approximately) for a cubic mesh.

One of the directions of the future work is to involve other view-position meshes (triangular, hexagonal, etc) to improve uniformity of the resulting view-direction set.

Other areas of investigation are to find an optimal metric ρ for cubic, triangular, etc, meshes and to try to solve the dual-uniformity problem in a most general formulation.

4.2 Deformation-based light fields

In this section we proposed a way of clustering images and a way of deformation a clusters average image into an arbitrary image from this cluster.

This method is based on a sequential rendering of perspective, occlusions, diffusive and specular reflections.

The future research in this direction will be an attempt to use different average images and different types of clustering, in order to optimise the quality and time of rendering.

5. REFERENCES

[1] Marc Levoy, Pet Hanrahan. *Light Field Rendering* Computer Science Department, Stanford University, SIGGRAPH'96 (New Orleans, LA, August 4-9, 1996)

[2] S.J.Gortler, R.Gizeszczuk, R.Szeliski, M.Gohen. *The lumigraph.* In proceedings of SIGGRAPH'96 (New Orleans, LA, August 4-9, 1996)

[3] Emilio Camahort, Apostolos Lerios, Donald Fussell. *Uniformly Sampled Light Fields*. Technical Report TR-98-9, Department of Computer Science, The University of Texas at Austin, Austin TX, March 1998.

[4] C. Yap, E.C. Chang, S.Mallat *Wavelet foveation*. IEEE PAMI January 1999

[5] Michael Oren, Shree K. Nayar *Generalization of Lambertian Model and Implications for Machine Vision.* CUCS-057-92, Department of Computer Science, Columbia University, New York, NY 10027, November 1992

[6] Ales Michtchenko *From-edges-to-regions approach and multilayered image processing*. Graphicon 1999, Moscow, August-September 1999.

Author:

Ales Michtchenko, PhD student of Moscow State University, Department of Applied Mathematics and Computer Science.
Address: Russia, Moscow 117292, Profsouznaya, 8-2-349
E-mail: ales.michtchenko@usa.net