Undersampled Light Field Rendering by a Plane Sweep

Yang Liu¹, George Chen², Nelson Max¹, Christian Hofsetz¹ and Peter McGuinness²

¹University of California, Davis
²STMicroelectronics
liu24@llnl.gov

Abstract
Images synthesized by light field rendering exhibit aliasing artifacts when the light field is undersampled; adding new light field samples improves the image quality and reduces aliasing but new samples are expensive to acquire. Light field rays are traditionally gathered directly from the source images, but new rays can also be inferred through geometry estimation. This paper describes a light field rendering approach based on this principle that estimates geometry from the set of source images using multi-baseline stereo reconstruction to supplement the existing light field rays to meet the minimum sampling requirement. The rendering and reconstruction steps are computed over a set of planes in the scene volume, and output images are synthesized by compositing results from these planes together. The planes are each processed independently and the number of planes can be adjusted to scale the amount of computation to achieve the desired frame rate. The reconstruction fidelity (and by extension image quality) is improved by a library of matching templates to support matches along discontinuities in the image or geometry (e.g. object profiles and concavities). Given a set of silhouette images, the visual hull can be constructed and applied to further improve reconstruction by removing outlier matches. The algorithm is efficiently implemented by a set of image filter operations on commodity graphics hardware and achieves image synthesis at interactive rates.

Keywords: light field rendering, image-based rendering., stereo reconstruction


1. Introduction

Photo-realistic rendering methods synthesize images by simulating physical interactions between light and matter. These methods produce images that are often visually indistinguishable from real photographs, but their computational requirements prohibit their practical application in many interactive systems. Image-based rendering (IBR), on the other hand, eschews complex lighting simulations and instead infers the information required for rendering from a set of source images. Thus, the quality of images created by IBR are fundamentally tied to the fidelity and availability of the source images used to create them. While IBR methods may not achieve comparable image quality to photo-realistic rendering, their computational cost is often cheap enough to allow interactive rendering. IBR can be broadly categorized into two types of methods: geometry-based and ray-based.

Geometry-based IBR uses geometric models to guide image synthesis. Projective texture-mapping (PTM) [1] is popular among such approaches. In PTM, new images are synthesized by drawing and texturing a polygonal model of the scene. Texture coordinates are computed by projecting polygon vertices into each source image and occlusion detection is applied to ensure that only polygons visible to a source image are textured by that image. Commodity graphics hardware architectures are designed to accelerate this image synthesis step and can draw and texture millions of polygons per second.

Ray-based methods typically assume little or no geometry information and derive information required for rendering entirely from the set of source images. Methods in this category include light field rendering [2], which synthesizes new images by looking up or interpolating the color of desired light rays from a ray database, commonly referred to...
as the light field function. This four-dimensional function is created from discrete ray samples, but is expensive to fully construct from a digital signal processing point of view since the dimensionality of this function implies a high minimum sampling rate that is difficult to meet. This presents many practical challenges to efficiently acquire, process and store the requisite rays for rendering.

The Shannon–Nyquist sampling theorem can be applied to understand the minimum sampling rate for reconstructing the light field function from a discrete set of samples and its sampling pattern. If the number of samples does not meet the minimum sampling rate, then the sampling pattern is said to be undersampled and aliasing will be introduced into the reconstructed function. This aliasing manifests as ghosting or noise artifacts, which are propagated to the output images generated by this function. One way to address this issue is band-limit the sampling function by filtering out the higher frequencies from its spectra to allow a lower minimum sampling rate. Unfortunately, this also filters out the detailed information and exchanges the aliasing problem with image blur. If a Lambertian model is assumed then many new ray samples can be inferred by analysing the scene geometry. The scene geometry can be estimated by applying multiple-baseline stereo reconstruction [3, 4] to the set of source images. If the geometry is known or can be reliably recovered, then images can be synthesized by PTM using relatively few source images. Each additional source image strengthens the stereo reconstruction, which in turn further increases the number of ray samples.

This paper describes an IBR method based upon this principle. There are two key tasks involved here: geometry estimation and image synthesis. The geometry estimation is restricted to a set of planes in the scene volume, which allows scalability and a convenient image-based representation for the geometry. New images are synthesized by drawing, texturing, and combining together these planes using the estimated geometry. This organization has two advantages: (1) the algorithm can be easily analysed under the light field framework to determine minimum sampling rates for rendering and (2) this approach allows two levels of parallel processing: pixel-level and plane-level. The graphics hardware can transparently leverage pixel-level parallelism for each plane, and the set of planes can be distributed across a set of graphics processing clusters. This paper describes a hardware implementation of this system and introduces new techniques to improve the geometry estimation using more sophisticated template matching metrics and by also incorporating the visual hull.

2. Related Work

Previous work in this research area includes Seitz’s voxel coloring algorithm [5]. Seitz estimates the scene geometry from a set of source images by evaluating and comparing the color consistency of voxels in the scene volume. Each voxel is projected into each source image, and their 2D projections are compared to calculate a matching score. Voxels with matching scores that exceed a predefined threshold value are labeled inconsistent and culled from the volume. After processing the entire scene space, the remaining voxels comprise the scene geometry, and can be directly applied as a model for image synthesis.

The method reported by Yang [6], is similar to the approach described in this paper. Yang also performs geometry estimation and light field rendering over a set of planes in the scene volume using programmable graphics hardware. However, his hardware implementation is only capable of applying rudimentary metrics, such as the Sum of Squared Distance (Section 5.1) to analyse the scene geometry. As a result, the images generated by Yang’s system exhibit many artifacts that arise from rendering using incorrectly estimated geometry. We address this deficiency by introducing and applying more complicated matching metrics to improve the quality of geometry estimation and rendering. Zitnick et al. demonstrated a system [7] that acquires video and computes the geometry estimation step off-line to subsequently render in real time. This system is not suitable for real-time teleconferencing, but allows implementation of even more sophisticated matching metrics to improve the final rendering quality.

3. Light Field Rendering

The light field parameterization characterizes each light ray by its intersections with two parallel planes, commonly labeled the camera plane with parameter \((s, t)\) and the focal plane with parameter \((u, v)\). Each ray can then be uniquely addressed by the quadruple \((s, t, u, v)\). The 4D light field function defined over this space assembles new images by querying the appropriate rays. Although the light field function is defined by this two-planes parameterization, the ray samples are normally represented internally by the set of source images; the light field function is usually kept implicit, and the two-plane parameterization is used only in the image synthesis.

3.1. Light field sampling

The continuous light field function can be completely reconstructed from a sufficient number of discrete samples. Chai’s analysis of this problem in the spectral domain [8], assuming constant depth planes and Lambertian reflectance, revealed that the spectral support of a light field signal is always bounded by the minimum and maximum depth of the scene, independent of its geometric complexity. A reconstruction filter and minimum sampling rate can then be derived by fitting spectral support replicas of the sampled light field signal into the smallest possible interval without any overlap. By incorporating and representing geometry as constant planes, spectral support replicas can be further compacted into smaller intervals to lower the minimum sampling...
rate. The relationship discovered between the number of images and amount of geometry required for full reconstruction is given by a joint image and geometry minimum sampling curve, which shows that geometry information can compensate insufficient image data and vice versa to satisfy the minimum sampling rate.

### 3.2. Dynamic focal plane

The focal plane in the two-plane parameterization can be adjusted to bring objects located at different distances from the camera plane into focus in the synthesized image [9]; objects that are close to or intersect with the focal plane appear with the best possible focus, while other objects away from the focal plane appear defocused. A composite image can then be assembled by rendering and identifying regions from a set of focal plane locations with the best focus.

### 4. Plane Sweeping

The plane sweeping algorithm compensates undersampled light fields by applying geometry estimation over each plane from a set of planes that occupy the scene volume. The planes are situated at the discrete locations that best approximate the depth complexity of the scene. Conceptually, this process can be described as a dynamic focal plane sweeping through the scene space, accumulating parts of the plane at each sweep location with the best ‘focus’ in the composite image. To perform this composition, a matching image must be computed and associated with the color image for the plane at each sweep location. The matching image is an intensity image with pixel values that reflect the level of ‘focus’ in the corresponding pixels of the color image. More precisely, each pixel of the matching image indicates the color consensus of the rays used to construct its analogous pixel from the color image. Not surprisingly, the match pixel value is also closely related to the depth hypothesis for its position on the focal plane. The final composite image is assembled by warping the match and color images into the desired view, and then combining the color images together using a Z-Buffer algorithm on the matching images.

#### 4.1. Single plane warping

Warping an image from one view into another is performed by a projective mapping transformation known as a homography. The parameters of this mapping includes a focal plane (also called the inducing plane) and two projection matrices associated respectively with the source and target view (Figure 1). Let $P_s$ and $P_t$ be the $3 \times 4$ projection matrices for the source and target view, respectively, and let $n$ be the normal vector to the inducing plane. The homography $H$ is given by:

$$H = P_t \left[ \begin{array}{cccc} P^{-1}_s \cdot I_{3 \times 3} & 0_{3 \times 3} \end{array} \right]$$

If the source and target views are separated only by a baseline that is perpendicular to $n$, then $H$ is reduced to a simple affine translation mapping, which can simplify computation for the image warp and resampling steps. This principle can be applied to the traditional light field two-plane configuration where the camera plane is parallel to the focal plane and the source views’ camera centers are positioned on the camera plane with their optical axes oriented parallel to the plane normal. In this case, only one homography between a reference source camera and the target view needs to be explicitly computed. All subsequent mappings can be determined by first translating to the reference view and then applying the precomputed homography.

#### 4.2. Image synthesis

Color images are computed by a weighted pixel-wise averaging of the source images. Pixel weights are assigned across the color image and are computed based on the relative position, angle, and image resolution of the source views relative to the target view. These weights are normalized to sum to unity at each pixel location. Matching images are similarly computed; pixels from the matching image are each assigned a matching score determined by comparing the color similarity of corresponding pixels (as well as the corresponding pixels from the local neighborhood) in the source images. The metrics and heuristics used to compute the matching scores are reviewed in Section 5.

The final output image is synthesized by assembling the color images together according to their matching images. This composition is performed in the target viewpoint by first warping all color and matching images into the target viewpoint.

© 2006 The Authors

Journal compilation © 2006 The Eurographics Association and Blackwell Publishing Ltd.
Each focal plane contains a matching image which indicates a match likelihood score for regions along the plane. The match for a pixel is determined by marching along its ray to find the plane location with the best matching score.

view using the homography mapping and then selecting the color pixel with the best matching pixel value (Figure 2) for each pixel of the target image. This pixel-selection step is implemented in hardware using the Z-Buffer by accumulating the matching images in the depth buffer. A confidence image can be created by reading back the contents of the depth buffer to reveal the quality of matches across the image. Finally, a depth image can optionally be computed by substituting the color images with depth labels unique to each plane sweep location.

A silhouette image is constructed by subtracting a background image from a source image and then thresholding the result into a binary image. The set of silhouette images derived from the source images together define the visual hull of the foreground object in the scene [10–12]. This hull is an initial conservative estimate of the object geometry and can be applied to constrain the computation of the color and matching images and generally improve the rendering quality.

An explicit polygonal representation of the visual hull can be constructed by extending the silhouettes from projection centers past their image planes into 3D cones (Figure 3), and then computing their common intersection volume using constructive solid geometry. The pixel projections of the object are contained within their respective silhouettes, so the visual hull is always consistent with the silhouette from each source view and is a maximum bound for the object. However, this is not necessarily an exact bound to the object volume since there may be possible concavities on the object surface that cannot be detected by silhouettes. With enough silhouette images, the visual hull converges to a bounding volume that is tighter than the object’s convex hull.

The visual hull can actually be applied to the rendering without constructing its polygonal geometry. This is accomplished by warping the silhouette images into the image plane of the target view and then computing their bitwise intersection, the result of which is applied to constrain both the color and matching image for each focal plane location. This masking operation is implemented in hardware by loading the stencil buffer with the intersection image before rendering the color and matching images.

The matching metrics we review in this section are defined over template windows and assume that the projections of a scene feature, such as a 3D point or surface, when observed from nearby viewpoints, should locally share the same distinct color and texture signature within a small

**Figure 2:** Each focal plane contains a matching image which indicates a match likelihood score for regions along the plane. The match for a pixel is determined by marching along its ray to find the plane location with the best matching score.

**Figure 3:** The visual hull is constructed by common intersection of image silhouette cones. This hull approximates the object geometry to remove unnecessary computation and to also avoid matching errors that arise from color ambiguity outside of the object volume.
neighborhood in the image. However, due to color ambiguity and view-dependent reflectance (non-Lambertian), this does not always occur in practice; color signatures may not be consistent among projections of the same feature or even unique among projections of different features. These errors manifest as both false positive and false negative matches in the matching image, which of course degrades the final rendering quality. Sophisticated matching metrics are more successful at suppressing false matches, but are also more expensive to compute than simpler ad hoc metrics. The cheaper matching metrics are computed by differencing, while the expensive metrics are based on correlation.

5.1. Differencing

Matching metrics based on differencing use the color distance between a pair of templates $T_1$ and $T_2$ to estimate their similarity. The most basic of such metrics is the sum of absolute difference (SAD) metric. Let $T_1$ and $T_2$ be two template windows. The SAD between $T_1$ and $T_2$ is given by:

$$f_{SAD} = \sum_{i,j} |T_1(i,j) - T_2(i,j)|$$  \hspace{1cm} (2)

If each $k \times k$ template is interpreted as a $k^2$-dimensional point, then the SAD metric computes the Manhattan distance between the two template points. Similarly, the sum of squared distance (SSD) computes the squared Euclidean distance between two templates:

$$f_{SSD} = \sum_{i,j} (T_1(i,j) - T_2(i,j))^2$$ \hspace{1cm} (3)

5.2. Correlation

The normalized cross-correlation (NCC) metric is given by:

$$f_{NCC} = \frac{\sum_{i,j} T_1(i,j)T_2(i,j)}{\sqrt{\sum_{i,j}(T_1(i,j))^2 \sum_{i,j}(T_2(i,j))^2}}$$ \hspace{1cm} (4)

If the $k \times k$ template is interpreted as a $k^2$-dimensional vector, then the NCC metric represents the normalized dot product, which computes the cosine angle between the two template vectors.

$$f_{NCC} = \frac{\mathbf{t}_1 \cdot \mathbf{t}_2}{||\mathbf{t}_1|| ||\mathbf{t}_2||} = \cos(q)$$ \hspace{1cm} (5)

NCC assumes constant color saturation, which may not be uniform across the source images. Therefore the templates should be mean-subtracted before computing the dot product. This resulting metric is called zero-mean normalized cross-correlation (ZNCC):

$$f_{ZNCC} = \frac{(\mathbf{t}_1 - \bar{\mathbf{t}}_1) \cdot (\mathbf{t}_2 - \bar{\mathbf{t}}_2)}{||\mathbf{t}_1 - \bar{\mathbf{t}}_1|| ||\mathbf{t}_2 - \bar{\mathbf{t}}_2||} = \cos(q)$$ \hspace{1cm} (6)

Although the ZNCC metric clearly requires the most computationally effort, it produces the best matching results. However, ZNCC is numerically unstable if at least one of the templates contain low color variance (i.e. $|t - \bar{t}| \approx 0$), in which case the results will become numerically unstable. A common strategy for addressing this problem is to add a small bias to the denominator of Equation 6 when computing the ZNCC.

5.3. Multiple image matching

Although the matching metrics introduced so far are comparing the similarity of two templates windows, these matching metrics are generalizable to assess the color similarity between an arbitrary number of templates. One way to define a multi-template matching metric is to construct a reference template $\mathbf{t}_{ref}$ from the set of templates (i.e. by arithmetic mean) and then compute the pair-wise matching scores between the reference template and each constituent template.

The intermediate matching scores can then be aggregated together:

$$\mathbf{t}_{ref} = \frac{1}{n} \sum_{i} \mathbf{t}_i$$

$$f_{SSD} = \sum_{i} f_{SSD}(\mathbf{t}_i, \mathbf{t}_{ref})$$ \hspace{1cm} (7)

Increasing the number of source images to compute the matching image generally improves the match quality. However, if the inclusion of a source image increases the longest baseline distance in the set of source views, then the results may actually be worse. Other factors such as relative differences in field of view may be detrimental to the matching quality. Therefore the selection of source camera views to participate in computing the matching image is very critical. Such a selection function depends on the sampling pattern of the source views and can usually be precomputed offline.

5.4. Template library

The size and shape of a matching template implies expectation of local continuity of scene elements and their 2D image projections—that corresponding pixels from different images are projections of the same point and their neighboring pixels in each respective image are also in correspondence. Using a template window implies that all points that project to pixels within this template are at similar depth. This assumption is false when the template window contains a surface discontinuity such as in the case of an object profile (Figure 4). In this situation, dissenting pixels outside of the object projection will worsen the overall matching score, even if a true match occurs. This is why square template windows typically fail to identify these discontinuities.

Determining the correct support of a match is very critical. Since a single template cannot capture all variations of
match support, we apply a library of templates to compute the matching image, and keep the best match at each location. The set of templates in this library reflect the types of image features that are likely to occur. For example, there are templates to match corners, edges, and other features at various positions and orientations in the projected images. The size of the templates is also important to establishing correct support. Large templates include more pixels to support they match, but might likewise also contain more discontinuities. Small templates typically do not contain many discontinuities, but also do not have enough variance to disambiguate between different hypothesis, which introduces many false positive matches that ultimately result in a noisy matching image. In practice, the template size should be set relative to the image scale of scene features; fine small geometric detail would be missed completely by relatively large templates. Templates in the library should also be set to roughly be the same size since otherwise the matching noise generated by the smaller templates may contaminate matching results produced by larger templates.

5.5. Matching errors

Matches are established by comparing color similarity, but color is not a unique quantity. Template matching relies upon local texture similarity, but local textures are also not unique. Therefore, the matching image is always susceptible to error. These errors manifest in the estimated geometry as outlier points, and ‘fattened’ surface boundaries and profiles. Two common sources of these matching errors are homogeneous areas and repeating patterns in the texture. In these situations, templates appear to be locally similar, and the correct match cannot be distinguished (Figure 5). These errors can be mitigated by decreasing the baseline distance between source images or by using more source images to establish the match—both of these strategies imply a higher sampling rate. Some statistical analysis can be applied to each matching image to identify and reject outliers, but this is difficult and expensive to integrate into a graphics hardware implementation. However, in the context of rendering undersampled light fields, matching errors are generally more tolerable than ghosting artifacts.

6. Hardware Implementation

The plane-sweeping algorithm is implemented in OpenGL on an ATI Radeon 9800 graphics card (Figure 6). The homography mapping is implemented in the transformation stage of the OpenGL pipeline by a vertex program, the color and matching images are computed in the rasterization stage by a fragment program, and the results are assembled and composited using the hardware Z-Buffer. The color and matching images based on the differencing metrics require a single rendering pass to compute. Matching images based on correlation metrics require several more passes. In this case, the intermediate data between each pass is stored into temporary image buffers [13], which are used again as input for subsequent rendering passes. In this feedback loop, the transformation stage is disabled such that no further warpings are applied to the images upon re-entry into the rendering pipeline.

The image warp is specified using the OpenGL projection and modelview matrix stacks. The warp is initialized for each focal plane by loading the homography matrix into the projection matrix stack, and the identity matrix into the modelview stack. The vertex and fragment shaders are invoked by drawing a local screen quadrilateral that spans the viewport. In the fragment program, the color image is internally represented using the RGB color channels, and the matching image is stored in the Alpha color channel (in the last
composition step, the matching image is moved into the depth channel).

6.1. Image matching filters

The SAD/SSD and NCC/ZNCC matching metrics, along with light field rendering can be expressed in terms of image convolution and basic image arithmetic operations. This is convenient since most of these image operations are efficiently implemented in hardware. For example, computing a SAD matching image from two source images involves two basic image operations: image subtraction and convolution with a template filter. The subtraction step computes the initial pixel-wise difference image, and the convolution step is necessary to aggregate these values into matching scores for local templates. Similarly, computing the NCC also involves several convolution steps. Convolving the resulting matching score image with a min (or max) filter using a fixed template shape effectively allows each template to shift around its raster location and seek the best match. This is useful for implementing different shifts of canonical template shapes in the template library. Image convolution is implemented more efficiently as two separable 1D convolutions that operate over the rows and columns of the image.

We implemented the SAD, SSD, NCC and ZNCC matching metrics as image filtering operations using the fragment programs and four $7 \times 7$ basic square templates rotated to $0^\circ$, $22.5^\circ$, $45^\circ$ and $67.5^\circ$ to cover possible orientations of corner and line features in the image. Applying the MIN/MAX filter to these four rotated square templates allows to increase the number of effective templates in the library to 196.†

†Cell positions within related and shifted templates may not coincide with actual raster positions; samples are obtained for cell positions using bilinear interpolation.

7. Results

Our experiments included both synthetic and real light fields. We acquired synthetic source images from the POV-Ray [14] raytracer, and real source images from six DragonFly [15] FireWire cameras linked together by an external timing trigger. Camera projection matrices are computed by in-house camera calibration software (and compared against the actual projection matrices used to generate the synthetic images to verify accuracy).

Figures 7, 8, and 9 demonstrate dense light field rendering using the plane sweep approach on synthetic images. Each of the output images were generated using a sampling grid of $16 \times 16$ source images, with four source pixels contributing to each output pixel. Matching score images are generated using the SSD matching metric. Good results are obtained at 12 frames per second using roughly 15–25 focal planes, depending on the depth complexity of the scene. Figure 10 demonstrates the background subtraction process in which silhouette images are obtained. Note that although the computed silhouette image over-estimates the true silhouette, rendering results are still acceptable since over-estimated regions can still be culled by other silhouette images.

7.1. Matching metrics

A comparison of matching metrics (Figure 11) shows that SSD and ZNCC are roughly equal in terms of rendering quality. The NCC matching metric is sensitive to homogeneous textures (i.e. the blue curtain), and incorrectly estimates depth along these areas. This contributes to noticeable blur in the rendered image. By subtracting out the mean from each template, ZNCC relies upon local color variance to infer matches, and generates a more accurate depth map along homogeneous areas. However, the ZNCC requires several more rendering passes to compute each matching image, which significantly lowers the frame rate.
7.2. Template library

Results show that matching is generally improved as more template windows are included in the library. Figure 12 compares results using (a) a single 7 \times 7 square template (b) a sliding window [3] (the square template positioned at each of its 49 cell locations) and (c) the template library (Figure 4) (which includes the four rotations of the square template, each positioned to the templates). By modeling edges and corners along the silhouette profile, the template library is able to reduce the thickening artifacts present in (a) and (b). Template windows that target anticipated features in the image generally improve the matching results. We compute 196 templates simultaneously by taking advantage of parallel resources on the graphics card. As these resources expand, additional templates may be included. Further results are shown in Figure 13.

8. Conclusions and Future Work

We have presented and demonstrated results from a new hybrid light field rendering framework that combines the benefits of many existing approaches. Our system applies geometry estimation to image synthesis, casting both of these tasks as image filters and operations that are efficiently implemented on graphics architecture, generating images at interactive rates. Since we estimate geometry dynamically, this system is suitable for processing streaming video input. A limitation of this method, common to all shape-from-stereo algorithms, is the matching errors due to color ambiguity. However, the errors due to incorrectly estimated depth are generally no worse than the ghosting artifacts when rendering directly from an undersampled light field. Further research directions and interests include: (1) improving the stereo matching by developing more accurate matching metrics and methods for establishing matching supports (i.e. Bayesian stereo matching [16]), (2) improved techniques for automatic image silhouette segmentation, (3) cached geometry estimations, (4) layered depth image representation for color and depth images and (5) optimizing the hardware implementation.

References


Figure 8: Balcony dataset.
Figure 9: Nature dataset.

Figure 10: Visual hull rendering using silhouettes extracted by background subtraction.
Figure 11: Comparison of matching metrics (20 focal planes).

Figure 12: Comparison of matching filters (20 focal planes, 15 fps).
Figure 13: Baby dataset.