# Non-Uniform Sampling of Image-Based Rendering Data with the Position-Interval-Error (PIE) Function

*Cha Zhang and Tsuhan Chen*
Dept. of Electrical and Computer Engineering, Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213, USA
*{czhang, tsuhan}@andrew.cmu.edu*

## ABSTRACT

In this paper we study the non-uniform sampling problem for image-based rendering (IBR). We first propose a general position-interval error (PIE) function that can help solve practical sampling problems. We then propose two approaches to capturing IBR data non-uniformly based on PIE, namely progressive capturing (PCAP) and rearranged capturing (RCAP). PCAP is applicable for static scenes. Based on the images captured so far, PCAP determines where to take the next image. RCAP, on the other hand, is for capturing both static and dynamic scenes. The goal is to intelligently arrange the positions of a limited number of cameras such that the final rendering quality is optimal. Experimental results demonstrate that the non-uniform sampling approaches outperform the traditional uniform methods.

## I. INTRODUCTION

Image-based rendering (IBR) has been a very popular research topic recently. By capturing a set of images or light rays in the space, the goal of IBR is to reproduce the scene correctly at arbitrary viewpoint. Compared with geometric models that dominate the traditional 3D rendering techniques, IBR provides the benefits that images are easier to obtain, simpler to handle and more realistic to render.

The image-based rendering process consists of two major stages – sampling and rendering. In the sampling stage, images or light rays are captured (sampled) from the scene and stored. In the rendering stage, novel views are reconstructed from the captured images. The literature of IBR has been mainly focused on different IBR representations and rendering methods [1]. For example, when dense image samples are available, new views can be rendered by interpolation [2][3][4][5]. 3D correspondence was used in several IBR approaches, such as in [6][7][8]. When some sort of geometry information is known, it can be incorporated into IBR as well [9][10][11][12]. On the other hand, little research has been performed on the sampling stage, which is as important as the rendering stage. In fact, it is the sampling stage that will essentially determine the final rendering quality.

For an ideal IBR capturing system, the following properties are disired:

- **Adaptive**: A good capturing system should be able to adapt to different scenes. It would be ideal if the camera arrangement can reflect where more images are needed.
- **Ease of setup, control and calibration**: If the capturing system is real, the easiness of setup, control and calibration is essential for its success.
- **Matching between the viewing space and the capturing space**: IBR is capable of generating virtual views where no images have ever been captured. However, due to many constraints such as camera resolutions and inaccuracy of the rendering geometry, we still hope that the capturing space can roughly cover where the viewing space will be.
- **High and consistent rendering quality**: High rendering quality is always desired. In addition, the rendering quality should be uniform across different views. Jittering of the rendering quality when the user moves around is a fairly annoying effect.
- **Low storage and short capturing time**: Although the rendering quality always improves when more images are taken, it is preferable to take fewer images due to storage or cost concerns. Either we achieve the specified quality with the minimum number of images, or we pursue the best quality with given number of images.
- **Robust**: Adaptive capturing algorithms that try to optimize the rendering quality may get trapped to local optima. Some conservative strategies can help increase the robustness of the system.

- **Use robust geometry**: Geometry is useful in improving the rendering quality. Whenever possible, we should reconstruct the geometry to help rendering. On the other hand, as the geometry will most likely determine the final rendering quality to some extent, the reconstruction method should be robust in a real system so that it does not hurt the rendering.

Most of the proposed IBR representations adopt uniform sampling for easy setup and control. Robustness may also be considered, but the other desired properties are largely ignored. In the few IBR sampling approaches [13][14][15][16][17][18] that have been proposed, most of them answer the following question: given a scene and a fixed uniform sampling pattern (such as light field [3] or concentric mosaics [5]), what is the minimum number of samples required to reconstruct the scene without aliasing. Many of them shared the same strategy: analyze the Fourier spectrum of the IBR representation and then sample it as a multi-dimensional signal. In [19], Lin and Shum performed sampling analysis on both light field and concentric mosaics with scale-space theory under the constant-depth assumption. The bounds are derived from the aspect of geometry and based on the goal that no "spurious detail" should be generated during the rendering (referred as the causality requirement).

In this paper, we propose to sample the IBR representations non-uniformly. It is well known in the signal processing community that non-uniform sampling could have better efficiency than uniform sampling, as uniform sampling is just a special case of non-uniform sampling. Moreover, it is the non-uniform sampling that can easily integrate all the properties as mentioned above into one system, as we will show later. Unfortunately, the methods applicable in uniform sampling in literature are not directly extendable to non-uniform sampling.

Non-uniform sampling has been massively studied in the signal processing literature [20][21]. For example, it is well established that a band-limited signal can be uniquely determined from non-uniform samples, provided that the average sampling rate exceeds the Nyquist rate [22]. Iterative methods may be used to reconstruct the band-limited signal from such non-uniform samples [23]. However, these theories are not easily applicable in practical problems such as IBR. First, IBR is a high-dimension signal. Non-uniform sampling theory for high-dimensional signal is not trivial [24]. Second, since the sampling process of IBR involves taking images of the scene, it is more natural to consider each captured image as a sample. Such a sampling process has never been studied in the literature. Third, the non-uniform sampling theory may tell us how many samples are enough, but it does not tell where to take these samples. Finally, in IBR we often cannot afford to capture enough images for a perfect reconstruction. When the number of samples is limited, it is not clear from theory how these samples should be taken.

Recently we proposed to use stochastic sampling and the sampling density function to analyze IBR non-uniform sampling [25]. The idea is to assume that we know the optimal stochastic sampling strategy on the scene surface. Using the Monte Carlo method [26], we may obtain the optimal sampling scheme for the IBR representation. Unfortunately, this approach requires too much knowledge about the scene, such as the scene geometry and the scene surface's optimal sampling density function. The application of such analysis may be limited.

Fleishman *et al.* [27] proposed an automatic camera placement algorithm for IBR. A mesh model of the scene is known. The goal is to place the cameras optimally such that the captured images can form the best texture map for the mesh model. They found that such problem can be regarded as a 3D art gallery problem, which is NP-hard [28]. They then proposed an approximation solution for the problem by testing a large set of camera positions and selecting the ones with higher gain rank. Here the gain was defined based on the portion of the image that can be used for the texture map.

Another related work is by Schirmacher *et al.* [29]. They proposed an adaptive acquisition scheme for a light field setup. Assuming the scene geometry is know, they captured the scene recursively through subdivision on the camera plane based on some rendering quality measurement through image warping. Although they showed that the estimated rendering error decreases monotonically as the adaptive acquisition goes on, they did not compare the rendering image quality between adaptive capturing and uniform capturing, which is essential for showing the advantage of adaptive acquisition. We think that this might be due to the very limited viewing range for a light field setup.

In this paper, we study the non-uniform sampling of IBR under a new mathematical framework. Our contributions can be summarized as follows:

- We propose a new tool to solve non-uniform sampling problems, namely the position-interval error (PIE) function. It can be used for the sampling of IBR and potentially many other applications.
- We propose progressive capturing (PCAP) for IBR. PCAP always captures more images around the region where the estimated rendering quality is the worst. When the number of captured images is limited, we showed through experiments that non-uniform sampling is better than uniform sampling.
- We propose rearranged capturing (RCAP), which is targeted for capturing dynamic scenes with multiple cameras. The cameras are rearranged in such a way that the estimated rendering quality is equal everywhere (or biased as the user prefers).

The paper is organized as follows. In Section II we present the PIE function as a general tool for sampling analysis. Section III briefly discusses how we can estimate the PIE value of an image pair for IBR. Section IV and V describe the progressive capturing and the rearranged capturing, respectively. Conclusions are given in Section VI.

## II. THE POSITION-INTERVAL-ERROR FUNCTION

In this section, we propose a new mathematical formulization for sampling. Compared with the approaches proposed in the signal processing literature, our framework is more engineering-oriented. That is, we want to apply our formulization in practical sampling problems.

### A. Problem formulation

We start our discussion with a scalar variable function $\mathbf{y} = f(x)$. Here $x \in R$ is a scalar variable defined on the real axis, which can be the camera position on a circle (e.g., concentric mosaics [5]). $\mathbf{y}$ can be a variable, a vector, a matrix or anything else that is useful. In the scenario of IBR sampling, $\mathbf{y}$ is an image captured at position $x$. Without loss of generality, we assume the function is continuous. That is:

$$\Delta \mathbf{y} \to 0, \text{ when } \Delta x \to 0 \tag{1}$$

This assumption is often true in practical applications. For example, when two images are captured at very close positions, what they captured will be very similar.

The general sampling problem can be considered as the following: knowing there is a function $\mathbf{y} = f(x)$, find out a set of sample positions $X = \{x_i, i \in I\}$, where *I* is a countable index set, such that the reconstruction of $\mathbf{y}$ meats certain error requirement. Since infinitely many functions can have the same value on set $X = \{x_i, i \in I\}$, the above problem is meaningful only by adding constraints on $f(x)$. In the traditional sampling theory, a common constraint is that $f(x)$ is band-limited [20][21]. That is, the Fourier transform of $f(x)$ is non-zero only on a finite support. In our framework, we choose not to impose such a constraint, as the definition of Fourier transform may be impossible on a general function $\mathbf{y} = f(x)$, where $\mathbf{y}$ can be anything such as an image.

### B. The position-interval error (PIE) function

For any function $\mathbf{y} = f(x)$, we first define a reconstruction error function *e* as:

$$e(x | X, M_R) \tag{2}$$

where *x* is a variable on which $f(x)$ is valid; *X* is the sample set; $M_R$ is the reconstruction method. The error function *e* can be specified by the user. For example, we may let:

$$e(x | X, M_R) = \left| f(x) - \hat{f}(x | X, M_R) \right|^2 \tag{3}$$

where $\hat{f}(x | X, M_R)$ is the reconstructed value at *x* with sample set *X* and reconstruction method $M_R$. Unfortunately, Equation (3) relies on the value of $f(x)$, which is not known in practice.

In many practical applications, the reconstruction method $M_R$ is predefined before sampling. For example, in IBR reconstruction, we often use a geometry assisted rendering algorithm that interpolates nearby light rays [30]. Although other rendering algorithms are possible, we choose it for good balance of rendering quality and speed. Furthermore, many practical methods reconstruct the function value at *x* only based on a small subset of samples in *X* that is close to *x*. Therefore we may rewrite the error function independent of $M_R$ and most other samples in *X*:

$$e(x | X, M_R) = e(x | S_{x,X} \subset X), \text{ for given } M_R. \tag{4}$$

where $S_{x,X}$ is the subset of samples in *X* that is used to reconstruct $f(x)$ at *x*.

We are particularly interested in reconstruction methods that only use the two closest samples in *X* to interpolate the function value. In other words, we may write:

$$e(x | X, M_R) = e(x | \{x_1, x_2\} \subset X, x_1 \le x < x_2) \tag{5}$$

where $x_1$ and $x_2$ are the two closest samples to *x* in *X*. The *average* quality of reconstruction in the range of $[x_1, x_2]$ can be written as:

$$E(x_1, x_2) = \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} e\left(x \middle| \{x_1, x_2\} \subset X, x_1 \le x < x_2\right) dx \tag{6}$$

For illustration purpose, let $x = (x_1 + x_2)/2$ and $\Delta x = x_2 - x_1$, we may reparameterize the function $E(x_1, x_2)$ as $E(x, \Delta x)$. This function measures the average reconstruction quality at the interval $[x - \Delta x/2, x + \Delta x/2]$ if we take two neighboring samples at $x - \Delta x/2$ and $x + \Delta x/2$ and use them to do the reconstruction in that interval (Figure 1 (a)). We name it the *position-interval error (PIE) function*, as shown in Figure 1 (b). Notice again that each point on the PIE function does not correspond to a sample of $f(x)$. Instead, it represents the average reconstruction error between *two* samples. Later we will use the PIE function to help solve the non-uniform sampling problem.

As shown in Figure 1 (b), the PIE function is a surface defined on the 2D plane $(x, \Delta x)$. Since it is related to the signal $f(x)$, the reconstruction method $M_R$ and error function *e*, general properties about PIE rarely exist. We next make several assumptions about the PIE function that are generally true:

***Assumption 1:*** *$E(x, \Delta x) \to 0$, when $\Delta x \to 0$.*

If we take two samples that are very close to each other and use them to reconstruct the function values in between, the reconstruction error will be very small (given the reconstruction method makes sense). This is a direct result from the continuity assumption made for $f(x)$ (Equation (1)).

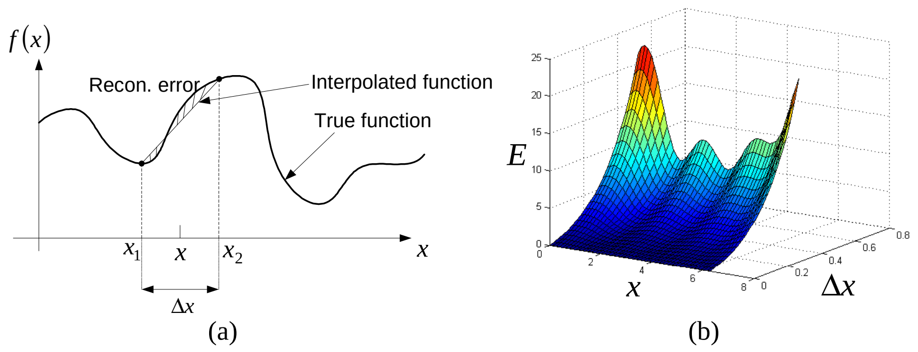


**Figure 1 An example PIE function. (a) The original function, its samples and reconstruction. (b) The PIE function.**

***Assumption 2:*** *$E(x, \Delta x)$ is continuous as a function of $(x, \Delta x)$.*

***Assumption 3:*** *There exists an interval $T$, such that when $\Delta x < T$, $E(x, \Delta x)$ monotonically non-decrease with respect to $\Delta x$.*

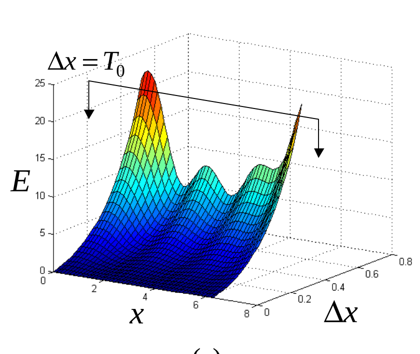
Assumption 3 means that if the interval between two samples is less than a certain threshold, reducing it will always improve the reconstruction quality. This assumption actually imposes a smoothness constraint on $f(x)$, which is similar to the band-limitness assumption used in the literature. For the rest of the paper, we assume $\Delta x < T$ is satisfied.

## C. Sampling with the PIE function

Both uniform and non-uniform sampling can be studied by the PIE function. For example, if we decide to uniformly sample the signal with a certain period $T_0$, all the possible average reconstruction errors between two neighboring samples are on a curve representing the value of the PIE function at a constant $\Delta x$:

$$E_{uniform}(x) = E(x, \Delta x)\big|_{\Delta x = T_0} \tag{7}$$

An example of the above curve is shown in Figure 2. Notice that at different *x*, $E_{uniform}(x)$ is varying, which indicates that the reconstruction errors at different positions are unstable. A real implementation of such a sampling will corresponds to a set of discrete points on the curve. Notice that on the $E_{uniform}(x)$ curve, each point corresponds to the average error between two neighboring samples on the original $f(x)$. Furthermore, as we do uniform sampling on $f(x)$, the points on the $E_{uniform}(x)$ curve are also evenly spaced by $T_0$.



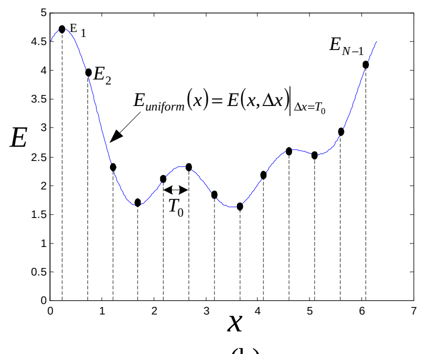


**Figure 2 Uniform sampling. (a) A vertical cut of the PIE surface at a certain $\Delta x$. (b) The error curve for uniform sampling and a concrete implementation.**

Let the point values on the $E_{uniform}(x)$ curve be $E_1, E_2, \cdots, E_{N-1}$, we may get the overall reconstruction error of the implementation as:

$$E = \sum_{i=1}^{N-1}(E_i \cdot \Delta x) = \Delta x \cdot \sum_{i=1}^{N-1} E_i \tag{8}$$

With Equation (8), we know that different implementation of the sampling at the same sampling rate may still result in different reconstruction errors. To minimize the overall reconstruction error, we need to shift the points on the $E_{uniform}(x)$ curve around together (thus also the samples on $f(x)$) until we get the minimum $\sum_{i=1}^{N-1} E_i$.

In non-uniform sampling, we are interested in algorithms that can maintain a constant average error throughout the possible *x* values[1]. In practice, such algorithms are of high interest. For example, in image-based rendering, we want the reconstructed views to have a uniform quality, such that when the user moves around he or she does not feel uncomfortable due to quality jittery. This "equal-error" sampling is equivalent to a horizontal cut of the PIE surface at a certain error $E_0$. Figure 3 shows the result of such a cut. Obviously, at different *x*, $\Delta x$ has to have different values in order to keep the reconstruction quality constant. This corresponds to a non-uniform sampling of the signal.

We may use the PIE function to solve two type of non-uniform sampling problem in general:

***Problem Type 1:*** *Given an error function $E_0(x)$, determine the minimum number of samples required and where to put them such that the final reconstruction error is less than $E_0(x)$.*

***Solution:*** Since $E_0(x)$ is a function of *x*, we need to first cut the PIE function with this function. That is, solve:

$$E(x,\Delta x) = E_0(x) \tag{9}$$

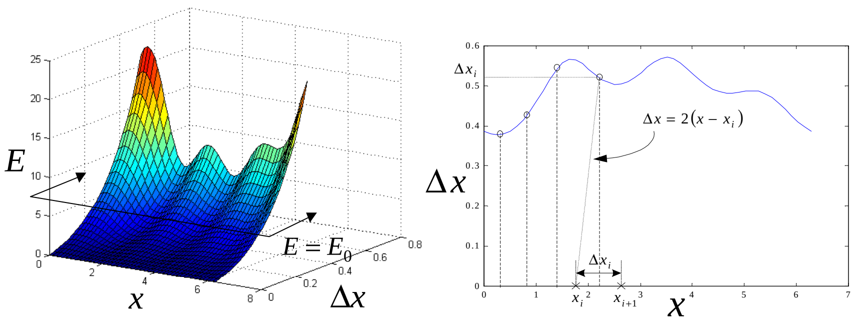
and get a curve on the $(x,\Delta x)$ plane. Notice that the equal error sampling in Figure 3 is simply a special case when $E_0(x) = E_0$ as a constant. The function $E_0(x)$ may simply be a user specified preference function. Let the cut curve be $c(x,\Delta x) = 0$. Next, let the optimal sample points of $f(x)$ be $x_1, x_2, \cdots, x_N$, where *N* is also unknown. They can be solved as the following problem:

$$\text{Find } x_1, x_2, \cdots, x_N \text{ and } N\text{, s. t. } c\left(\frac{x_i + x_{i+1}}{2}, x_{i+1} - x_i\right) = 0, \text{for } 1 \le i < N \tag{10}$$

Given $c(x,\Delta x) = 0$, the above problem can be easily solved in a recursive manner. For example, let $x_i$ be the current sample position on $f(x)$, we may just find $x_{i+1}$ by solving $c((x_i + x_{i+1})/2, x_{i+1} - x_i) = 0$, which can be done by finding the crossing point of $c(x,\Delta x) = 0$ and line $\Delta x = 2(x - x_i)$ to get $\Delta x_i$ and obtaining $x_{i+1} = x_i + \Delta x_i$, as is shown in Figure

[1] Non-uniform sampling can also be used to minimize the overall reconstruction error. To solve such a problem, we need to first change the error notation of the PIE function to be the *overall* reconstruction error between two neighboring samples. The solution of the problem corresponds to a set of samples on the modified PIE surface with the same slope $\frac{\partial E(x,\Delta x)}{\partial \Delta x}$. Iterative methods may be used to get the set of samples.

3. The initial point $x_1$ may be set as the minimum value of *x*, which is known. And the number of samples required is a nature result of the above recursive process.



(a) (b)

**Figure 3 Equal-error sampling – a horizontal cut of the PIE surface at a certain error** $E_0$**.**

***Problem Type 2:*** *Given a maximum number of samples N we may take and a relative error function* $E_r(x)$*, find the sample positions such that the PIE function satisfies* $E(x,\Delta x) = \alpha E_r(x)$*, and* $\alpha$ *is minimized.*

***Solution:*** This is a more difficult problem. We propose to solve it iteratively. Let the value of $\alpha$ at iteration *k* be $\alpha_k$, which means the current error function is $\alpha_k E_r(x)$. We may obtain $N_k$ through the method for solving problem type 1. The $\alpha$ value at the next step can be:

$$\alpha_{k+1} = \alpha_k + \beta(N_k - N) \tag{11}$$

where $\beta$ is a positive number that controls the step size of $\alpha$ during the iteration. Notice that when $E_r(x)$ is a constant, we get back to the equal-error sampling.

The above non-uniform sampling problems are solvable provided that the PIE function is known. However, in practice since the original signal $f(x)$ is unknown, the PIE function is also unknown. In Section IV and V, we propose two practical approaches for non-uniform sampling and apply them to image-based rendering. A common fourth assumption is made as follows:

***Assumption 4:*** *Although the PIE function of the sampled signal is unknown, it is possible to estimate the value of the PIE function at certain point given two neighboring samples.*

As we can only obtain certain PIE function values through sampling the signal, the solution to optimal or sub-optimal non-uniform sampling is often through *learn and adjust*. We first learn the PIE function values at several $(x,\Delta x)$ points by taking a set of samples from $f(x)$ (and applying Assumption 4). Based on the learned values, we may next decide how to adjust our sampling positions so that the PIE function values are more uniform. Section IV and V will show how such strategy can be used to obtain sub-optimal solutions for non-uniform sampling in real applications. Estimating the PIE function value for a pair of neighboring samples is discussed in Section III.

### D. Extension of the PIE function to higher dimensional space

We may extend the PIE function to higher dimensional space. Consider function $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{x} \in R^d$ is now a *d*-dimensional vector. Let the sample set be $\mathbf{X} = \{\mathbf{x}_i, i \in I\}$, where *I* is a countable index set. Define extended PIE (EPIE) function as $E(\mathbf{x}, r)$, where *r* is the distance from $\mathbf{x}$ to its closest sample point in $\mathbf{X}$. Here we assume that the reconstruction error will be mainly determined by the distance *r*.

We may make the same assumptions for the EPIE function as the PIE function. For example, we have:

***Assumption 1:*** $E(\mathbf{x}, r) \to 0$*, when* $r \to 0$.

***Assumption 2:*** $E(\mathbf{x}, r)$ *is continuous as a function of* $(\mathbf{x}, r)$.

***Assumption 3:*** *There exists a distance R, such that when* $r < R$, $E(\mathbf{x}, r)$ *monotonically increases with respect to r.*

Also, for solving practical problems, we may assume:

***Assumption 4:*** *Although the EPIE function of the sampled signal is unknown, it is possible to estimate the value of the EPIE function at certain point given its neighboring samples.*

## III. ESTIMATE THE PIE FUNCTION VALUES FOR IBR

In practical applications, we often do not have the PIE function at hand when we sample the signal. Nevertheless, we assume that the PIE function value may be estimated at certain points given pairs of neighboring sample points from the original signal. Through the *learn and adjust* strategy, we may develop algorithms that achieves optimal non-uniform sampling. Finding a good estimation of the PIE function values is difficult but often critical for such non-uniform sampling analysis. In this section, we use IBR as an example to show that such estimation is possible.

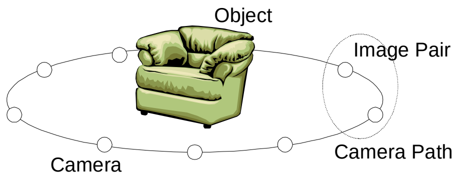


**Figure 4 Inward-looking concentric mosaics.**

Consider a simple scenario where we capture a scene through inward-looking concentric mosaics, as is shown in Figure 4. We may easily define a PIE function for it. Consider any pair of neighboring images captured on the camera path. Applying the geometry assisted rendering algorithm in [30], any novel view between these two images will be rendered only by them, assuming each light ray in the novel view is interpolated with its two closest neighboring captured light rays. The PIE function can be defined as the average rendering quality (error) between all the possible image pairs.

We propose to use the color consistency criterion to estimate the PIE function values for a given image pair. The idea has been proposed in [30] without the introduction of PIE function. We briefly describe the approach below. For a Lambertian surface, color consistency criterion claims that light rays from the same surface point should have the same color. It has found many applications in geometry reconstruction problems, such as the voxel coloring algorithm [31] and various stereo algorithms [32][33]. In IBR, although the scenes are often non-Lambertian, color consistency is still applicable locally. Many IBR rendering algorithm obtain virtual light rays through interpolation. If the light rays used to interpolate the same virtual light ray have very different colors, we can expect that the rendering quality might be bad.

Assume that we know a certain volumetric model to represent the scene geometry. For each image pair, we may simply project all the voxels to two images and verify their color consistency. Here color consistency can be measured by the difference between the colors of the two projections. The more the overall color difference, the worse the estimated rendering quality between these two images, the larger the estimated PIE function value we give to the image pair. Please refer to [30] for detailed algorithms. Also, the consistency measure in [34] can be applied for estimating the extended PIE function values.

## IV. PROGRESSIVE CAPTURING

### A. Problem setup

In progressive capturing (PCAP), we consider capturing a static scene for image-based rendering (IBR) with one or several cameras. This is a very common task in IBR. As the scene is static, we may move the cameras around on a surface or a curve, e.g., a spherical surface or a circle, to compensate the fact that we may not have enough cameras at the same time. Due to storage or cost concerns, we may want to use the least number of images to capture the scene, given a certain rendering quality requirement.

At the first stage, we consider a scenario similar to concentric mosaics [5], as in Figure 4. Cameras are placed on a path that forms a circle, pointing to the object center, and indexed by their angular positions. While in traditional concentric mosaics the cameras are spaced evenly, we allow the cameras to be any where on the circle. From the signal processing point of view, we employ a non-uniform sampling instead of the traditional uniform one in [5]. Since uniform sampling often causes large variation of the quality of the reconstructed signal (as shown in Figure 2 (b)), the goal of our non-uniform sampling is to bring more uniform rendering quality.

If we know the PIE function of such a problem, it is obvious that the PCAP problem becomes a good example of the type 1 problem introduced in Section II-C. Unfortunately, in practice we do not know the PIE surface. What we may have are some estimated PIE function values for certain image pairs by applying the algorithm in Section III. Therefore, PCAP needs a *learn and adjust* solution (Section II-C).

## B. The solution

We have proposed a solution for PCAP in [30]. Therefore, in this paper we only briefly describe the idea and focus on how we can explain the approach with the PIE function.

Our proposed PCAP approach starts by capturing a set of images uniformly spaced. We guarantee that the distance between two neighboring captured images satisfy the Assumption 3 of the PIE function. We are able to estimate the PIE function value for each image pair, as described in Section III. The next camera position is chosen to be the middle point of the pair with maximum estimated PIE function value (named a *split* of the image pair). This process runs iteratively until the rendering quality requirement is meat or we have achieved the maximum number of images allowed. Notice that we may easily add user preference to the system by adding weights to the estimated PIE function values in different region. Robustness can be fulfilled by splitting an image pair when its interval is much more than all the other intervals.

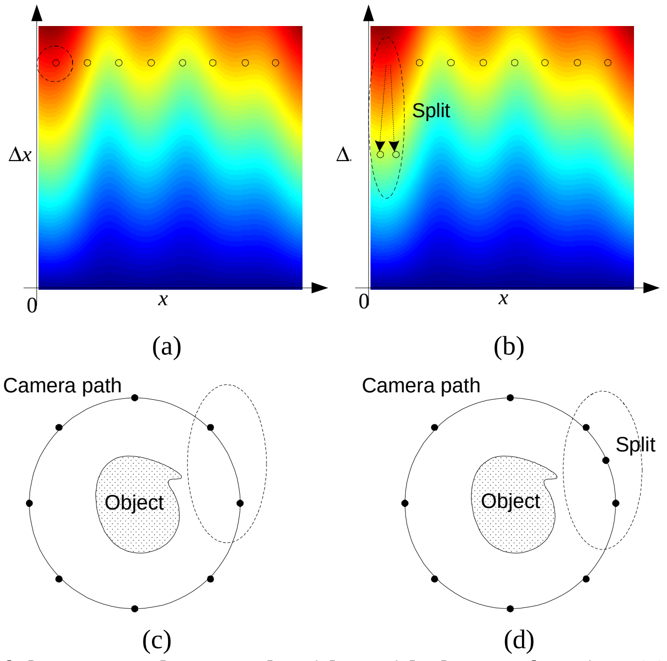


**Figure 5 The explanation of the proposed PCAP algorithm with the PIE function. (a) Before split. (b) After split. (c) Before split on the camera path. (d) After split on the camera path.**

The explanation of the proposed algorithm with the PIE function is shown in Figure 5. In the top two figures, the horizontal axis is $x$ and the vertical axis is $\Delta x$. The colors of the two figures represent different values of the PIE function. At the same $x$, a point at the bottom (smaller $\Delta x$) always has a smaller value than a point on the top (larger $\Delta x$). Across different $x$, the same color represents the same PIE function value. We show in Figure 5 (c) a uniform sampling on the camera path, which corresponds to Figure 5 (a) where a set of points on the PIE surface have the same $\Delta x$ (each image pair is a point in this figure). Obviously, these points have different PIE function values. The left-most point has the biggest value, so we decide to split it. After the split, we get Figure 5 (d) and (b). A new sample image is taken and the left-most point on the PIE surface is *replaced* by two points that has half of its $\Delta x$. Obviously, after the split, the sampled image pairs have a more uniform PIE function values. We iterate the above process until the stopping criterion is met. Notice here we have implicitly introduced a fifth assumption about the PIE function:

***Assumption 5:*** *The PIE function is smooth enough along x such that:*

$$\left|E(x,\Delta x)-E\left(x,\frac{\Delta x}{2}\right)\right|<\left|E\left(x,\frac{\Delta x}{2}\right)-E\left(x\pm\frac{\Delta x}{4},\frac{\Delta x}{2}\right)\right| \tag{12}$$

We found that such assumption is often satisfied in our practical experiments.

## C. Experimental results and extensions

We show three example scenes where PCAP outperforms uniformly captured IBR (UIBR). The three scenes are *Reflective cone* (*RC*), *Hemisphere bowl* (*HB*) and *Punctured sphere* (*PS*), as are shown in Figure 6 (a), (b) and (c), respectively. All the scenes are rendered with POV-Ray raytracer [35]. Synthetic scenes are used because we may have full control over the cameras in the experiments. The visual hull algorithm [37] was adopted for geometry reconstruction when necessary, which is due to the robust geometry concern in Section I. A real system for PCAP has been implemented in [36].

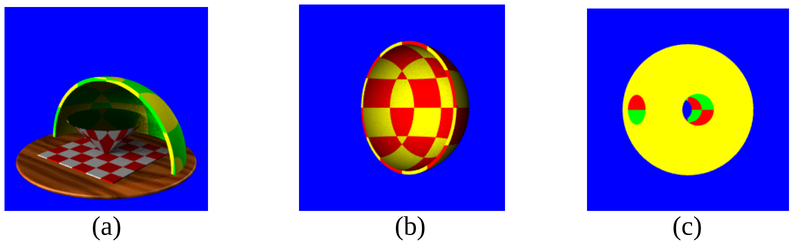
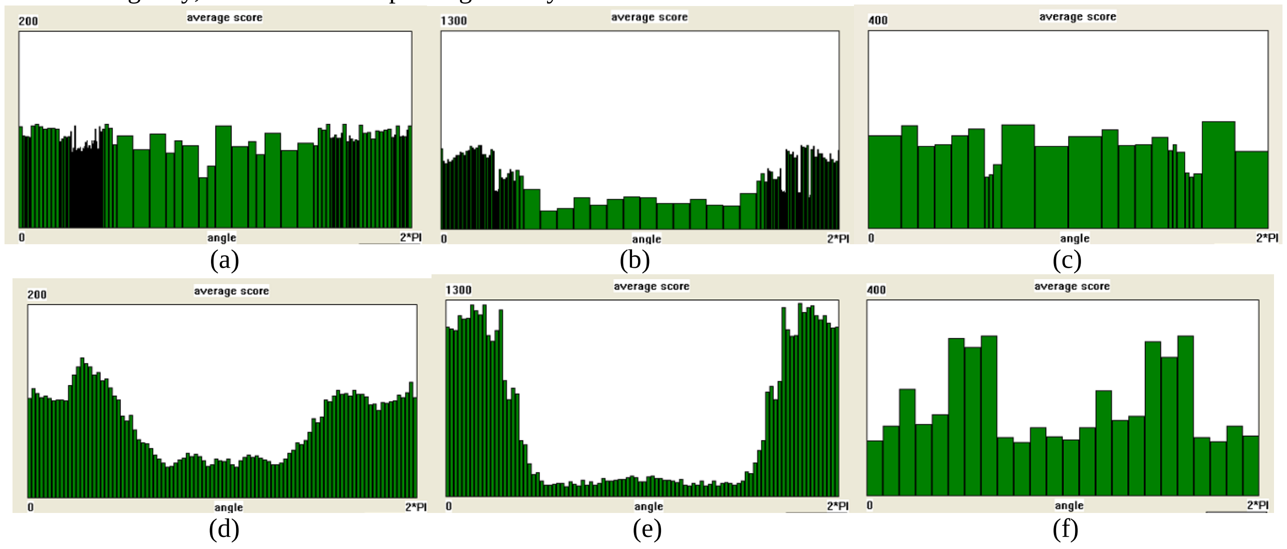

(a) (b) (c)

**Figure 6 The three test scenes. (a) Reflective cone. (b) Hemisphere bowl. (c) Punctured sphere.**

In the *RC* scene, we assume that the geometry model of the scene is known. A reflective cone is positioned at the center of the scene, which requires more samples than usual objects. This requirement stems from the conclusion in [17] that non-Lambertian scenes need more samples than Lambertian ones. The *HB* scene is the difference between two concentric hemispheres at different radii. Its geometry model is reconstructed with the visual hull algorithm [37]. Notice that no matter how many images we take for the scene, the reconstructed geometry cannot be correct, since the scene has a concave shape. In the IBR literature, rendering a scene with inaccurate geometry model is very normal. This example shows that more samples should be placed at the side where the geometry is wrong. In the third scene *PS*, a sphere is punctured twice along different directions. The purpose of this example is to show that, PCAP can determine the next views intelligently, which in turn helps the geometry reconstruction of the scene.



(a) (b) (c)
(d) (e) (f)

**Figure 7 Resultant estimated PIE values after PCAP and UIBR. (a) Scene *RC*, PCAP. (b) Scene *HB*, PCAP. (c) Scene *PS*, PCAP. (d) Scene *RC*, UIBR. (e) Scene *HB*, UIBR. (f) Scene *PS*, UIBR.**

The experimental results are shown in Figure 7. We take 96, 96 and 24 images for the scene *RC*, *HB* and *PS*, respectively. In the figures, the height of each green bar represents the estimated PIE value after PCAP or UIBR. The boundaries of the green bars represent the locations of the cameras on the circle. Obviously, the distributions of the heights of bars are more uniform after PCAP than UIBR. This shows the effectiveness of PCAP. Please refer to [30] for more detailed experimental results report.

The current system constrains all the cameras to lie on a circle. More generally, we may allow them to lie on a 2D surface, such as a sphere or a plane. This extends our PCAP approach to use the EPIE function. For camera placement on a 2D surface, the corresponding EPIE function is 3D (in $E(\mathbf{x}, r)$, $\mathbf{x}$ is a 2D vector). Although the EPIE function has higher dimension than the PIE function, in practice such an extension does not introduce too much trouble. Work has already been done in [29][34]. In both papers, the light field [3] setup was used. Nevertheless, they did not show large improvement from UIBR to PCAP. We consider the reason as that in light field the viewer's viewpoint range is rather constrained. Therefore, we plan to perform PCAP on a spherical surface in our future work.

## V. REARRANGED CAPTURING

### A. Problem setup

Consider a scenario where a couple of cameras are used to capture a static or dynamic scene. Since the scene can be dynamic, we need all the cameras to be present at the same time. Unfortunately, the required number of cameras for a regular scene is often huge and may not be affordable. We propose to use a small set of cameras and make them movable to compensate the problem caused by limited amount of cameras. Assume the cameras can move along some surface or trajectory, and for a dynamic scene, the movements of the scene object are relatively slow compared with the camera movements. The goal is to find the optimal camera positions for the existing cameras such that the rendering quality can be optimal.

Again we assume all the cameras are on a circle, looking at the object center. This is similar to the current PCAP setup, as is shown in Figure 4. The difference is that PCAP add cameras one by one, assuming that the scene is static, while in RCAP, all the cameras stay on the circle and arrange themselves to optimally capture the scene. Therefore the scene in RCAP can be dynamic. The PIE function can still be used to help solve RCAP.

As the maximum number of cameras / images is known, RCAP is a type 2 non-uniform sampling problem (Section II-C). Since the PIE surface is unknown, such problem is very difficult to solve. As in PCAP, we know that the only way we may get knowledge about the PIE function is through taking sample images. This way we can get sample points on the PIE function. Next we propose a simple solution to perform RCAP.

### B. The solution – inconsistency force based method

We propose a very intuitive solution here. Let us introduce the concept of *inconsistency force*, a virtual force caused by the color inconsistency. Consider a certain camera on the camera path, as is shown in Figure 8. It belongs to two neighboring image pairs: the left pair $p_{left}$ and the right pair $p_{right}$. For each image pair, we may calculate its inconsistency score through the method in Section III. That is, we may find the average errors $E(p_{left})$ and $E(p_{right})$. Define the inconsistency forces on the current camera as:

$$f_{left} = k \cdot E(p_{left}); f_{right} = k \cdot E(p_{right}) \tag{13}$$

where $f_{left}$ is the force from the left and $f_{right}$ is the force from the right. The camera will stay at the same position if the two forces have equal strength. Otherwise, the camera will be pulled to move along the stronger force, which corresponds to the side with higher estimated reconstruction error. Assume that we start with an equally spaced distribution of cameras. The cameras keep moving until at a certain point all the cameras receives equal forces from left and right. We refer this state as a *stable* state.

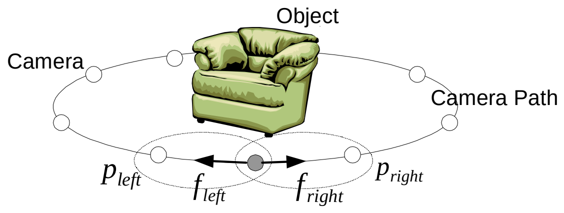


**Figure 8 Inconsistency force on a certain camera.**

We may again explain the above approach with the PIE surface. For illustration purpose, let us consider the movement of one camera, while all the neighboring cameras stay at their original position. As shown in Figure 9 (c), three cameras are circled with a dotted ellipse and the center one is the one under consideration. This corresponds to two points on the PIE function (also circled). As the left image pair has a higher PIE function value, the current camera moves leftwards. This is equivalent to reducing the $\Delta x$ of the left image pair and increasing the $\Delta x$ of the right image pair. After the movement, the result is shown in Figure 9 (d) and (b). Obviously, around the considered region we have a more uniform PIE function value or reconstruction quality.

The step size of the movement needs to be fine-tuned. In our preliminary experiments, we simply define the movement as:

$$\begin{cases} mv_{left} = \beta \Delta x_{left} \cdot (f_{left} - f_{right}) / \max(f_{diff}), & \text{if } f_{left} \geq f_{right} \\ mv_{right} = \beta \Delta x_{right} \cdot (f_{right} - f_{left}) / \max(f_{diff}), & \text{if } f_{left} < f_{right} \end{cases} \tag{14}$$

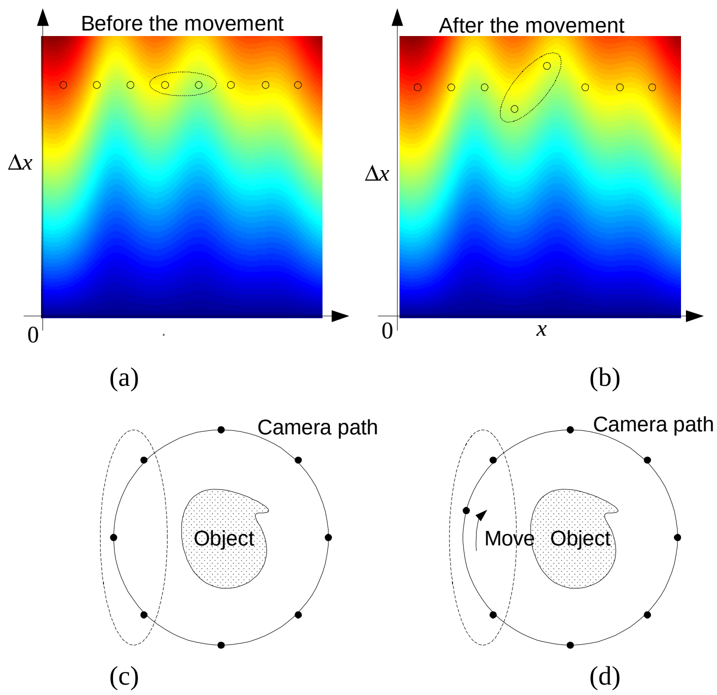


**Figure 9 The explanation of the proposed virtual force based RCAP algorithm with the PIE function. (a) Before the movement of a certain camera. (b) After the movement. (c) Before the movement on the camera path. (d) After the movement on the camera path.**

where $mv_{left}$ and $mv_{right}$ are the amount of motion towards left or right; $\Delta x_{left}$ and $\Delta x_{right}$ are the intervals of the left and right image pair; $\max\left(f_{diff}\right)$ is the maximum difference of the two forces on the same among all. The scalar $\beta$ is chosen heuristically in the current implementation. We let $\beta \le 0.5$ so that we may guarantee that all the cameras are still in order after the movement.

## C. Preliminary results

We have some preliminary results on the proposed inconsistency force based method. We apply the algorithm on the *Punctured sphere* (*PS*) scene. 24 cameras are placed on the camera circle. After 30 iterations, we obtain the inconsistency score map as in Figure 10 (a). Compared with that of the PCAP result in Figure 10 (b), the cameras are spaced more naturally. Subjective measure on their rendering quality is comparable.

The above algorithm basically considers each camera as an independent unit and they control their motion by themselves. The final result is (hopefully) a global (sub)optimum. Similar ideas may be applicable for other distributed applications such as sensor networks. To overcome possible problems caused by such distributed approaches, certain constraints may be enforced. For example, there might be extreme cases where all the cameras move towards the same position, punishment should be performed if two neighboring cameras become too far from each other.

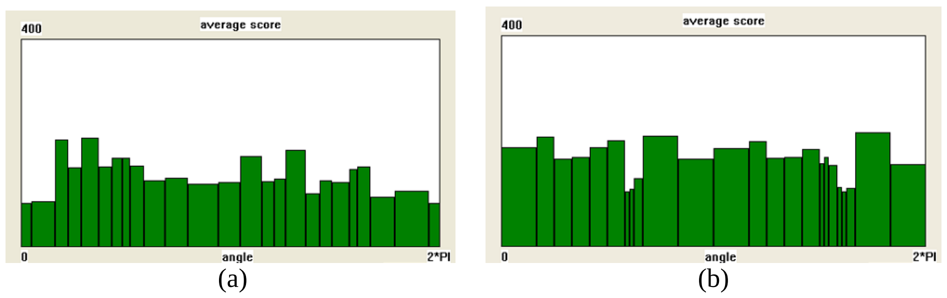


**Figure 10 Inconsistency score map of force based RCAP and PCAP for the scene *PS*. (a) Force based RCAP (b) PCAP.**

## VI. CONCLUSIONS

We studied the non-uniform sampling of IBR under the framework of the PIE function and showed that the PIE function is very general and has potential applications in many other problems. We gave two examples, namely the progressive capturing and the rearranged capturing. In both examples non-uniform sampling outperforms the traditional uniform sampling. We are exploring the possibilities of other non-uniform sampling schemes with the PIE function.

## REFERENCE

[1] S. B. Kang. "A survey of image-based rendering techniques", *VideoMetrics*, SPIE Vol. 3641, pages 2-16, 1999.

[2] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system", *Computer Graphics (SIGGRAPH'95)*, pp. 39-46, Aug. 1995.

[3] M. Levoy and P. Hanrahan, "Light field rendering", *Computer Graphics (SIGGRAPH'96)*, pp. 31, Aug. 1996.

[4] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, "The Lumigraph", *Computer Graphics (SIGGRAPH'96)*, pp. 43-54, Aug. 1996.

[5] H.Y. Shum and L.-W. He, "Rendering with concentric mosaics", *Computer Graphics (SIGGRAPH'99)*, pp.299-306, Aug. 1999.

[6] S. Chen and L. Williams, "View interpolation for image synthesis", *Computer Graphics (SIGGRAPH'93)*, pp. 279-288, Aug. 1993.

[7] S. M. Seitz and C.M. Dyer, "View morphing", *Computer Graphics (SIGGRAPH'96)*, pp. 21-30, Aug. 1996.

[8] M. Pollefeys, "Self-calibration and metric 3D reconstruction from uncalibrated image sequences", *Ph.D. Thesis*, ESAT-PSI, K. U. Leuven, 1999.

[9] L. McMillan, "An image-based approach to three-dimensional computer graphics", *Technical Report*, UNC Computer Science TR97-013, 1999.

[10] J. Shade, S. Gortler, L. W. He, and R. Szeliski, "Layered depth images", *Computer Graphics (SIGGRAPH'98)*, pp. 231-242, July 1998.

[11] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach", *Computer Graphics (SIGGRAPH'96)*, pp. 11-20, Aug. 1996.

[12] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured Lumigraph rendering", *Computer Graphics (SIGGRAPH'01)*, pp 425-432, Aug. 2001.

[13] J.X. Chai, X. Tong, S.C. Chan and H. Y. Shum, "Plenoptic sampling", *Computer Graphics (SIGGRAPH'00)*, pp.307-318, July 2000.

[14] S. C. Chan and H. Y. Shum, "A Spectral Analysis for Light Field Rendering", *ICIP 2000*.

[15] D. Marchand-Maillet and M. Vetterli, "Sampling Theory for Image-Based Rendering", *Master thesis*, EPFL, Apr. 2001.

[16] Z. C. Lin and H. Y. Shum, "On the Number of Samples Needed in Light Field Rendering with Constant-Depth Assumption", *CVPR 2000*.

[17] C. Zhang and T. Chen, "Spectral Analysis for Image-Based Rendering Data", accepted by *IEEE Trans. on CSVT Special Issue on Image-based Modeling, Rendering and Animation*.

[18] C. Zhang and T. Chen, "Generalized Plenoptic Sampling", *Carnegie Mellon Technical Report*: AMP01-06.

[19] Z. C. Lin and H. Y. Shum, "On the Number of Samples Needed in Light Field Rendering with Constant-Depth Assumption", *CVPR 2000*.

[20] Aldroubi and K. Gröchenig, "Nonuniform Sampling and Reconstruction in Shift-Invariant Spaces", *SIAM Review*, pp. 585-620, Vol. 43, No. 4, 2001.

[21] M. Unser, "Sampling – 50 years after Shannon", *Proceedings of the IEEE*, pp. 569-587, Vol. 88 No. 4, Apr. 2000.

[22] K. Yao and J. B. Thomas, "On Some Stability and Interpolatory Properties of Nonuniform Sampling Expansions", *IEEE Trans. Circ. Theory*, pp. 404-408, Vol. CT-14, No. 4, Dec. 1967.

[23] C. Cenker, H. G. Feichtinger and M. Herrmann, "Iterative Algorithms in Irregular Sampling: A First Comparison of Methods", *ICCCP-91*, March 1991, Phoenix, Arizona, USA.

[24] M. J. D. Powell, "The Theory of Radial Basis Function Approximation in 1990", *Advances in Numerical Analysis – II: Wavelets, Subdivision Algorithms and Radial Functions*, pp. 105-210, W. A. Light, Ed., Oxford Univ. Press, 1992.

[25] C. Zhang and T. Chen, "Surface Plenoptic Function: A Tool for the Sampling Analysis of Image-Based Rendering", *ICASSP 2003*.

[26] J. H. Halton, "A Retrospective and Prospective Survey of the Monte Carlo Method", *SIAM Review*, pp. 1-63, Vol. 12, No. 1, Jan. 1970.

[27] S. Fleishman and D. Cohen-Or and D. Lischinski, "Automatic Camera Placement for Image-Based Modeling", *Computer Graphics Forum*, Vol. 19, No. 2, Jun. 2000.

[28] J. O'Rourke, *Art Gallery Theorems and Algorithms*, The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[29] H. Schirmacher, W. Heidrich and H. P. Seidel, "Adaptive Acquisition of Lumigraphs from Synthetic Scenes", *EUROGRAPHICS'99*, Vol. 18, No. 3, 1999.

[30] C. Zhang and T. Chen, "Active Scene Capturing for Image-Based Rendering", submitted to *ICCV 2003*.

[31] S. M. Seitz and C. R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring", *CVPR 1997*, pp. 1067-1073.

[32] I. Cox, S. Hingoraini, S. Rao, "A maximum likelihood stereo algorithm", *Computer Vision and Image Understanding*, Vol. 63, No. 3, pp. 542-567, May, 1996.

[33] S. Roy and I. J. Cox, "A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem", *Int. Conf. on Computer Vision (ICCV'98)*, pp. 492-499, Jan. 1998.

[34] C. Zhang and T. Chen, "Active Scene Capturing for Image-Based Rendering with a Light Field Setup", *Carnegie Mellon Technical Report*: AMP03-02.

[35] http://www.povray.org/

[36] C. Zhang and T. Chen, "A System for Active Image-Based Rendering", *ICME 2003*.

[37] A. Laurentini, "The Visual Hull Concept for Silhouette Based Image Understanding", *IEEE PAMI*, pp. 150-162, Vol. 16, No. 2, 1994.