AN EFFICIENT APPROACH

TO

LAYERED-DEPTH IMAGE BASED RENDERING

RAVINDER NAMBOORI

NATIONAL UNIVERSITY OF SINGAPORE

2003

AN EFFICIENT APPROACH

TO

LAYERED-DEPTH IMAGE BASED RENDERING

RAVINDER NAMBOORI

(B.Comp (Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2003

ACKNOWLEDGEMENTS

I would like to sincerely thank A/P Teh Hung Chuan and Dr. Huang Zhiyong, my project advisors, for their continual support and guidance throughout my research. Their assistance, patience, warmth and constant encouragement have been invaluable to this research. My thanks to Dr. Chang Ee Chien and Mr. Low Kok Lim for their helpful suggestions.

I am extremely grateful to Mr. Chong Peng Kong for his time and help with the lab apparatus. This project wouldn't have been possible without his willingness to help at any moment and his readiness to ensure that all is well with my work.

My special thanks to Mr. Sushil Chauhan, for his help in better formulating the sampling arc functions.

Ravinder Namboori

Oct 2003

TABLE OF CONTENTS

SUMMARY

There exist a lot of computer graphics techniques to synthesize 3-D environments, of which, Image Based Rendering (IBR) techniques are becoming increasingly popular. In this thesis we concentrate on improving one such IBR technique, viz. Layered Depth Images (LDI). This technique, like many other IBR techniques, works on a set of pre-acquired imagery to model the world, and often, problems have been encountered in determining how exactly to decide on this pre-acquired set of sample images. As the quality of the synthetic view is governed by the initial stages of sampling, addressing this problem can enhance the result achieved by the eventual rendering engine.

This research presents a new approach to rendering an LDI, by adaptively sampling the raw data based on the determined set of sample parameters. This approach eliminates the redundancy caused by over-sampling, and removes the hole artefact caused by under-sampling. In addition, the rendering speed of the LDI is improved by the pre-computed visibility graph and patch lookup table.

Subject Descriptors:

G.1.2	Approximation of surfaces and contours
I.3.3	Picture/Image Generation
I.4.1	Digitization and Image Capture
I.4.8	Scene Analysis

LIST OF FIGURES

INTRODUCTION

1.1 Documentation Layout

For the purpose of easy readability, the content has been divided into seven chapters. This chapter, Chapter 1, is an introduction to the research as a whole, an introduction to the various phases of the research, as well as the nature of this project. We shall highlight the problem statement and the overall system framework in this chapter.

Chapter 2 covers an overview of the related work in the area to date. Included in this chapter, is a brief description of the various researches and techniques in the area of Image Based Rendering and Layered Depth Images in particular, sampling methods and automatic camera placement techniques.

Chapter 3 highlights the proposed improvement to the Layered Depth Image system by adaptively sampling the reference images and pre-computing the *patch lookup table*. Also discussed in this chapter are the derivations and assumptions leading to the essential steps involved in the system framework.

Chapter 4 is an elaboration of the implementation of the system and the sampling issues involved in the research. This section takes a methodological approach to exemplify the steps involved in demonstrating the proposed method of improving the Layered Depth Image system.

Chapter 5 discusses the results achieved by the implementation of the proposed method. In this chapter, we go through the various examples used and the outputs we got using our system, and contrast the result with those achieved by an earlier framework, which does not include the proposed improvements.

Chapter 6 concludes this thesis discussing the lessons learnt from this research and reinstating the goals achieved and the solution proposed and implemented.

Chapter 7 addresses the future prospects of research in this area, and wraps up the report with a final word.

1.2 Image Based Rendering and the Sampling Problem

The traditional approach to synthesize realistic images of virtual environments involve modeling the environments using a collection of 3-D geometrical entities with their associated material properties, and a set of light sources. Then, rendering techniques such as radiosity and ray tracing are used to generate the images at given viewpoints. The realism of such rendered images is limited by the accuracy in the description of the primitive material and illumination properties and hand coded or mathematically derived graphical models. Also, real-time rendering using this technique relies heavily on the complexity of the scene geometry and the hardware configuration.

Computer Vision, on the other hand can be considered as an inverse process of computer graphics, which recovers 3-D scene geometry from 2-D images. Extracting 3-D geometry of a scene usually requires solving difficult problems such as stereovision, depth from

shading, or using expensive rangefinders. From the 3-D geometry recovered, approximated 3-D models are constructed, from which new images can be synthesized. However, these reconstruction techniques are usually computationally expensive and the reconstructed models suffer from the lack of accuracy.

Image Based Rendering is an emerging new field, which counters these limitations. In this technique, new images and 3-D worlds can be modeled without the knowledge of the geometry of the scene involved. Realism is achieved by the fact that the basic entities of the 3-D environment are no longer polygons or geometries, but are pre-acquired images. Fig 1.1 depicts the process of Image Based Rendering. As can be seen, it has emerged from both the fields of Computer Graphics and Computer Vision and yet bypasses the complicated and limiting stage of defining the scene's geometry. It also shows that the tedious 3D shape modeling can be avoided and little or no knowledge of 3D shape of the scene is required.

In addition, what is highlighted in Fig 1.1 is the new step of sampling, which dictates how different the Modeled Synthetic World is going to be, in comparison to the Real world. As the quality of the synthetic view is now governed by the reference images at our disposal and not any 3D geometry, the initial stages of sampling becomes of paramount importance. The *sampling problem* is to determine where the scene needs to be sampled from and how many such samples are required to adequately sample the scene. It is important that a robust solution be formulated for the problem of sampling and determining the exact set of reference imagery required in rendering the 3D world.

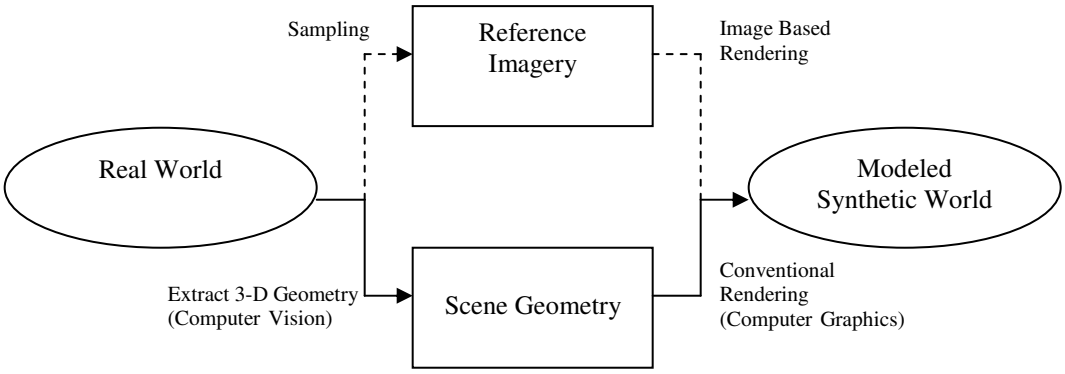


Figure 1.1: Model of Image Based Rendering

Tackling the sampling problem isn't as straightforward as over-sampling, as that would result not only in redundancy of sampled data, but also an increased amount of time to re-render the synthetic view. On the contrary an attempt to under-sample even if followed by stages of splatting, compromises on the realism of the modeled world, often leaving holes and visual artifacts, or portions of synthetically splatted patches.

A lot has been researched in the field of Image Based Rendering since its emergence a few years back. Essentially Image Based Rendering is about creating new photo-realistic images of complex scenes through interpolation techniques or other computations based on input data from photographs, drawings and rendered virtual scenes. There are various techniques to model a 3-D World using pre-acquired imagery, viz. Layered Depth Images, LumiGraph/Light Field, Panorama, View Morphing etc. All these techniques differ slightly in striking a balance between the computation involved in generating new views and the size of the sampled database. Irrespective of the approach, the stage of sampling is indispensable to its framework. The availability of geometry as in the case of methods like

Layered Depth Images provides ample opportunity to precisely select and limit the reference Imagery. As for the other approaches, splatting and other techniques to compensate for under sampling have been seen as a possible alternative. We shall overview these techniques in Chapter 2, under Overview of Related Work.

1.3 Problem Statement and Research Scope

This Research is focused on one of the areas of the huge field of Image Based Rendering, viz. Layered Depth Images. The aim of this research work is to enhance the realism and hasten the generation of the views achieved by the standard way of Layered Depth Image based Rendering by adaptively sampling the reference images and pre-computing a *patch lookup table*. The idea is to introduce a filtering stage after densely over-sampling the real world. The filtering stage, like the sampling stage, being a part of the pre-rendering phase, helps out the rendering engine by pulling out additional computations, and saving up precious time while rendering. This method would not only improve the quality of the synthetic images generated, in terms of getting rid of holes and occlusion artefacts, but will also enable quick generation of images, owing to the tabulation of the required sampled imagery that is acquired.

The major challenge is to effectively compute the required reference viewpoints from the dense sample to eliminate possible loss or redundancy of data. To create a compelling sense of virtual presence, the following goals must be achieved:

- Users can interactively navigate through the 3-D environment, without hardware acceleration.

- The photo-realism of the environment ought not to be compromised in terms of holes or other synthetic occlusions.
- The speed of the rendering pipeline should be unaffected by the fact that the sampled imagery is bigger than a sparsely sampled image set.

The sampled images are assumed to be taken under white light and with an ideal pinhole camera (no lens distortion).

The contributions of this research include:

- Implementation of a Layered Depth Image framework that enables rendering of complex 3-D environments, catering for absence of holes or visual artefacts in the modeled world.
- An efficient approach to tabulate the pre-acquired set of imagery, to ensure fast reference view selection and rendering of the synthetic views.
- A method which retains the realism of the 3-D environment, through dense samples of the real world, and yet achieves a rendering engine which is as fast as a sparse sampled LDI system.

1.4 The System Framework

The original Layered Depth Image system is essentially classified into 3 main phases. Scene Sampling, Scene Geometry and Photometry Extraction and Scene Resampling.

The System framework of this Layered Depth Image based rendering approach is depicted

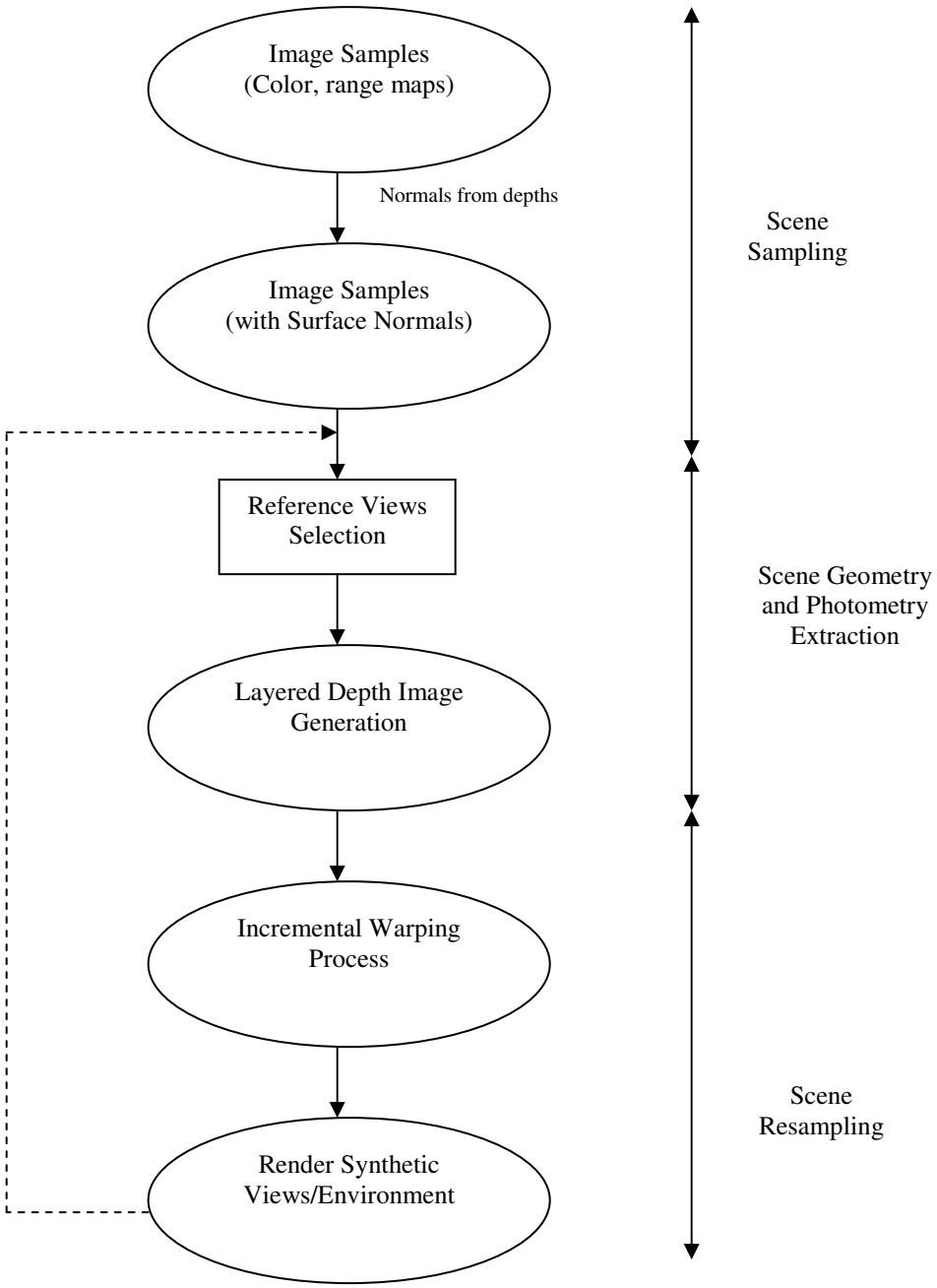


Figure 1.2: Framework of the Layered Depth Image based rendering System

in Fig 1.2.

We work on the first 3 stages of this framework, the so-called pre-computation phase, and

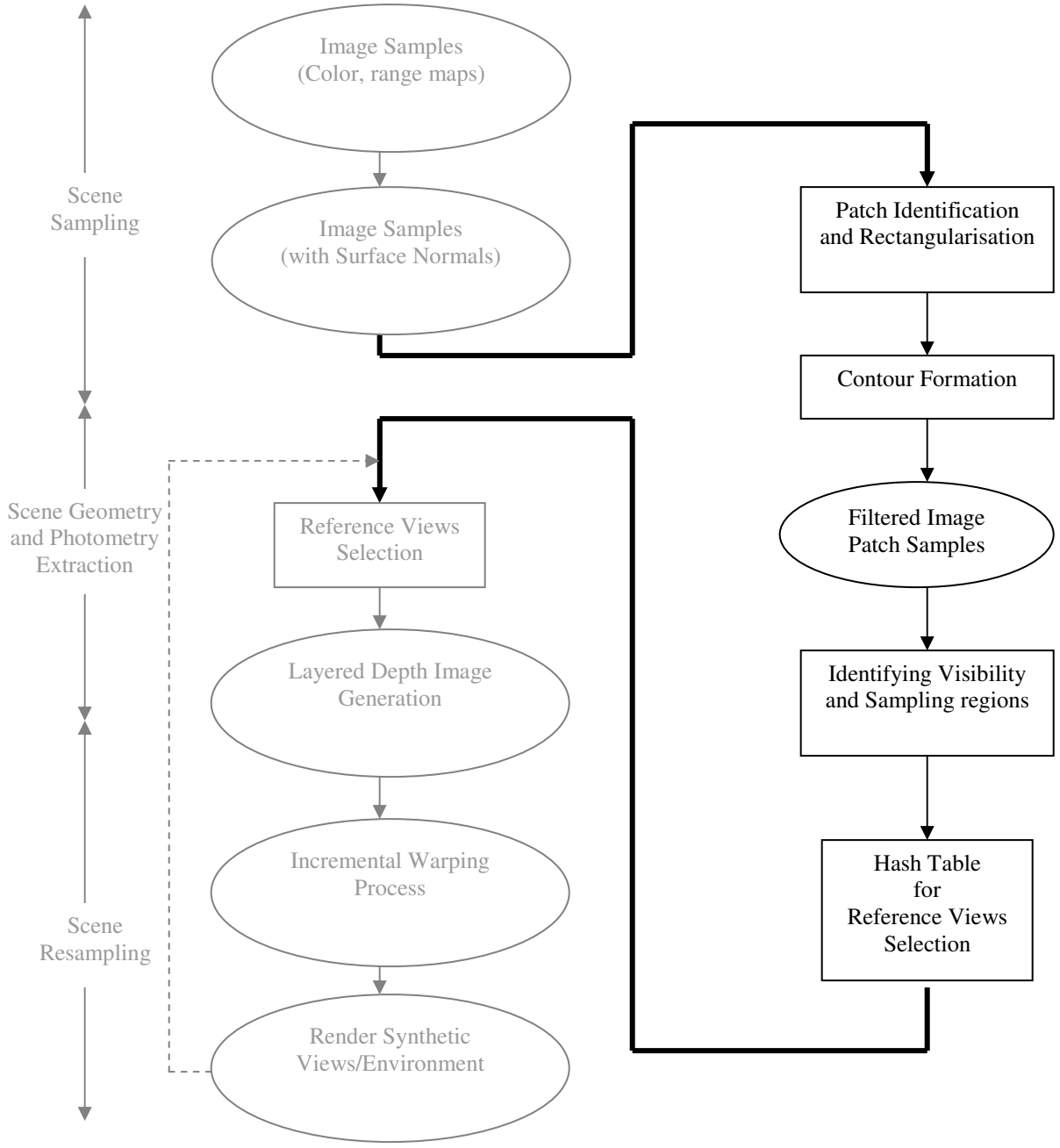


Figure 1.3: Our System Framework

modify the framework as depicted in Fig 1.3. We briefly go over these new stages of the framework in the sections to come.

1.4.1 Patch Identification and Rectangularisation

The surface normals at every point are calculated based on the range information. (for more information on the calculation of surface normals, please refer to chapter 4, section 4.3.2). The color & range maps, along with the normals constitute our sampled point cloud. From this point cloud, this step attempts to identify the uniform *patches*, surfaces that are not uneven and that fit in the camera's field of view, sets of points defined by certain patch constraints. Based on these constraints, the whole point cloud is divided into smaller uniform regions called patches. These patches, owing to the constraints thus applied, have a close to constant third dimension. The patches are then rectangularised, a recursive process which applies a greedy algorithm to extract the largest rectangle in the patch. At the end of this stage, we have categorized the point cloud into rectangular patches, ones that can be summarized as a line in two dimensions, when viewed from the top. This process is explained and discussed in detail in chapter 3.

1.4.2 Contour Formation

The output of the previous stage, viz. the rectangle patches, is fed into this part of the pipeline, in an attempt to identify unique *2D-contours* along the vertical axis. The aim of this stage is to identify those parts in the vertical space, which when viewed from the top, look as if it were in a single plane. This stage subsequently summarizes these parts of the object as a 2D-contour associated to a particular vertical range.

1.4.3 Identifying Visibility and Sampling Regions

In this step, we find the *visibility* and *sampling regions* for each of the contours thus found. *Visibility region* for a particular edge in a 2D-contour is defined as that region from

which the whole edge is visible, if there is no occlusion. A *sampling region* for a particular edge is defined as the region where it's appropriate to sample that particular edge, ensuring that all of the data visible on the edge is captured. A sampling region is determined by formulae dependent on the size of the edge, the camera calibrations and the sampling camera trajectory.

1.4.4 Patch lookup table for Reference view selection

This last step, despite being out of the scene-sampling phase, and being a part of re-rendering, is worthwhile mentioning at this point, because of the organization of the data in the prior phases. Given the structured organization of the sampled points of the original data, the selection of reference views to render while generating a new synthetic view becomes straightforward.

During rendering, the necessity to look for the closest reference viewpoints or the reference views which cover such and such occluded region is overcome by the fact that these considerations have been addressed during the pre-computation phase of determining the required set of sampled imagery. Hence, the reference view selection becomes a fast and straightforward procedure of looking up the hash table of patches thus created, for relevant reference data, as the viewer moves around the 3D space.

OVERVIEW OF RELATED WORK

Image based rendering techniques have been classified into four distinct categories: pixel based, block based, reconstruction based and mosaicing [Kang, 1997]. These categories are not necessarily mutually exclusive. Also, there exists a different categorization, i.e. Rendering from Interpolation of Dense Samples, Panorama based Rendering, Morphing and Depth based Rendering. These techniques vary largely in the knowledge of the geometry of the scene and the number of samples of the scene. We shall restrict our domain to the Depth based Rendering model, to be more specific, Layered Depth Image Based Rendering.

The depth based rendering model exploits the additional data available in terms of the 2D image samples being images with depths. These so called depth images, in addition to having the color values at a particular pixel, also contain the depth information at that location. Synthetic images for new viewpoints are created by a re-projection of the depth pixels in the reference depth images [Lee, 1998]. Layered Depth Image Based Rendering is an extension to the depth based rendering model, which performs warping from an intermediate representation called a Layered Depth Image (LDI) [Shade et al., 1998]. An LDI is a view of the scene from a single input camera view, but with multiple pixels along each line of sight. An LDI is constructed by warping n depth images into a common camera view.

This chapter surveys the various techniques employed to get around the sampling problem

for the LDI based rendering method, discussed in the previous chapter. While some of these techniques look at remedying the damage caused by the problem like splatting the holes during rendering, some others attempt to find the best next view to sample, assuming the first sample was ideal. All these techniques aim to exploit the geometrical knowledge to improve the photo-realism of the synthetically generated scene.

2.1 Splatting

Splatting is a technique, which aims to remedy the effects of the sampling problem. The Layered Depth Image, which is created from uniformly sampled images, is splat into the output image by estimating the projected area of the warped pixels [Shade et al., 1998]. This estimation is computed differentially based on the distance between the sampled surface point and the LDI camera, the field of view of the camera, the dimensions of the LDI and the angle between the surface normal at the sampled surface point and the line of sight to the LDI camera.

As splatting is a post-sampling step, care has to be taken that it doesn't slow down the rendering engine. In this view, a lookup table is generated. Before rendering each new image, the new output camera information is used to pre-compute the lookup table.

2.2 Multi-Resolution Sampling

Multi-Resolution sampling attempts to get around the problem of over-sampling or under-sampling for various camera distances, by sampling sets of images for different resolutions. While splatting and meshing are proposed to deal with the disocclusion artifacts, they are seemingly adequate only for post-rendering warping in which the

resolution of the current view does not deviate much from the resolution of the reference image.

In cases where an LDI is created from reference images not at similar distances from the object under consideration, insufficient sampling rate of the LDI might cause the synthetic view to look blurrier than it looks in the reference image closer to the object. On the contrary excessive sampling rate of the LDI might slow down the rendering pipeline.

The LDI Tree method [Chang et al., 1999], employed a hierarchical partition scheme with the concept of LDI, which preserves the sampling rate of the reference images by adaptively selecting an LDI from the LDI cluster for each pixel. In another approach, an L-System was implemented, which could store images of varying resolutions at different nodes of the L-System for effective tree modeling. [Lluch et al., 2004].

2.3 Sampling all Visible Surfaces

As the name suggests, in this technique of sampling all visible surfaces, an attempt is made to record a series of images that, collectively, capture all visible surfaces of the object. This technique revolves around the selection of a good heuristic method to find a good set of viewpoints for a given geometric model. The goal is to have sampled images from the computed viewpoints such that every visible surface is shown at least once. One such heuristic is to segment the object to exemplify hierarchical visibility [Stuerzlinger, 1998]. The scene is assumed to be a set of surface polygons organized in a hierarchy. The

hierarchical visibility method subdivides the scene hierarchy depending on the relative visibility of objects.

Yet another heuristic is to cover all possible surfaces, masking reference images as each surface is considered [Fleshman et al., 1999]. In this approach, the set of scene polygons visible from a viewing zone is approximated and then a greedy algorithm is employed to select a small number of camera positions that together cover every polygon in the geometric model. Towards this goal, the boundary of the walking zone is first tessellated. Scene polygons are subsequently subdivided to reduce the likelihood of the visibility problems. Visibility and quality of the subdivided sections of the polygons determine the worth of any reference image.

2.4 Best Next View Sampling

The best next view problem is that of selecting the next view for the sampling system to take, given some already acquired views of the object. Two criterions are often considered in solving this problem. The visibility criterion attempts to maximize the number of surfaces not seen thus far, by adding the next image to the sampled set, while the quality criterion aims to improve the quality of the surfaces sampled. The quality criterion prioritizes an image, which samples a decent number of surfaces, covering most areas of these surfaces, over an image, which samples a lot of surfaces but obliquely.

Several methods are available in this form of sampling, most of them differing in the way they establish the two criterion mentioned. In one particular approach, a volumetric representation, termed the voxelmap, is generated at each cycle of best next view

computation [Massios, Fisher, 1998]. The voxels thus scanned are marked empty, seen, unseen or as an occlusion plane depending on the visibility from the new view. The seen voxels carry a quality property, which is estimated by the aggregate normals of all the points sampled in a particular voxel.

In yet another approach, each range image sampled thus far is approximated by a triangular mesh [Garcia, 1998]. The resolution of the triangular mesh determines the minimum distance of that can be distinguished during the exploration process. The edges in these triangular meshes are marked as exterior, occlusion or interior depending on whether they bound a region or whether they are susceptible to occlude surfaces of the scene or whether they are formed by an overlap of two exterior edges. The quality criterion is satisfied by a voting mechanism. Each occlusion edge has an associated normal histogram and a tangent histogram. Every cell of the histogram keeps the sum of all the associated normals. That cell is looked up for which has received the maximum number of votes in either histogram.

2.5 Sampling Issue for other Rendering Techniques

Though drifting slightly from the area of concentration, interesting techniques have been researched in two other forms of Image based rendering.

In Point based rendering [Grossman, Dally, 1998], an attempt is made to ignore the issue of adequate sampling during rendering. The problem is dealt with, during the phase of sampling, by suggesting that to minimize the number of samples that adequately sample an object, the distance between the adjacent samples on the surface of the object should be

as large as possible but less than the pixel side length at the target resolution assuming unit magnification. An equilateral triangle mesh is used for the purpose.

For Lumigraph/Light Field Image based rendering techniques [Gortler et al., 1996], a spectral analysis of light field signals combined with the sampling theorem is used to derive the analytical functions that determine the minimum sampling rate [Chai et al., 2000]. The minimum sampling rate is obtained by compacting the replicas of the spectral support of the sampled light field within the smallest interval. As it is known that the spectral support of a light field signal is bounded by the minimum and maximum depths only, no matter how complicated the spectral support might be because of the depth variations in the scene, a reconstruction filter with an optimal constant depth can be designed to achieve anti aliased rendering.

THE PROPOSED IMPROVEMENT TO THE LDI SYSTEM

It is established that a Layered Depth Image is a view of the scene from a single input camera view, but with multiple pixels along each line of sight. It is a comprehensive image data structure, built to take into account various artifacts like occlusions and holes, by storing not just the first layer of pixels but also a few layers along each line of sight. Unfortunately, this comprehensive LDI framework's ability to render photo realistic views, devoid of holes and other visual artefacts, is highly dependent on the nature of the reference images sampled, to be precise, the number of reference images sampled and the position from which they are sampled.

The *sampling problem* is to determine where the scene needs to be sampled from and how many such samples are required to adequately sample the scene. It is worthwhile to note that under-sampling results in visual artefacts. On the contrary, over-sampling helps get around the problem of visual artefacts, but at the cost of the rendering speed. There is no one fixed scheme to adequately sample the LDI, as the occlusion artefacts and holes are largely dependent on the scene's geometry. Hence we adopt the approach to adaptively sample the scene based on the scene's available geometrical data.

In this chapter we shall discuss a method to adaptively sample a layered depth image, based on the geometrical information at our disposal. We shall elaborate how the LDI system is improved by this change in the sampling phase and the *patch lookup table* generated during the pre-rendering phase. In the chapters to follow, we shall go over a

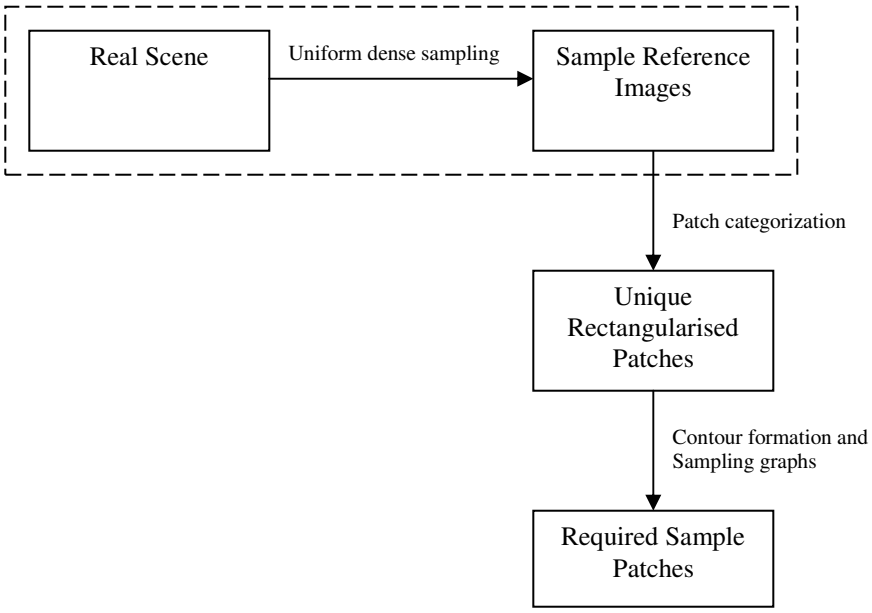


Figure 3.1: Adaptive Sampling Pipeline

sample implementation and the results observed using this model.

3.1 Brief Overview

Uniform sampling is the simplest alternative preferred by most LDI engines to-date, and the inadequacy of the sampling problem is solved by methods like splatting discussed in the previous chapter. The density of uniform sampling affects the quality of the output. Sparse uniform sampling results in visual artefacts while dense uniform sampling ends up with a slow re-rendering pipeline.

We approach the adaptive sampling method by starting with a highly dense uniform sample set and adaptively filtering redundant data, retaining only the adequate information. Figure 3.1 depicts the adaptive sampling pipeline.

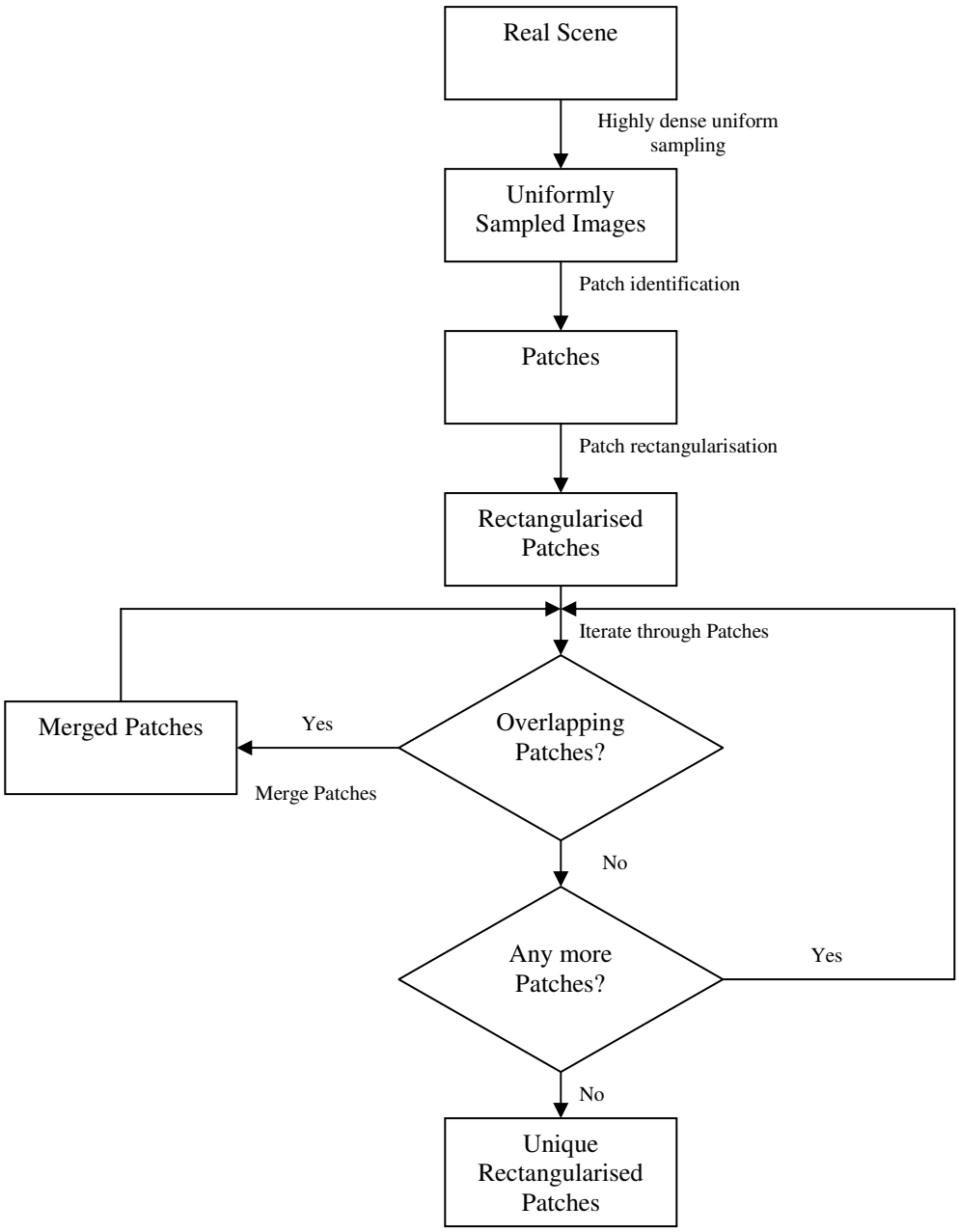


Figure 3.2: Patch Categorization

The step of *patch categorization* first categorizes the uniformly sampled images into uniquely defined patches. These patches help in guiding the sampling. The next step forms contours with these patches, and identifies the sampling regions required. Eventually, the exact reference points to sample from are deciphered, and the unnecessary data is

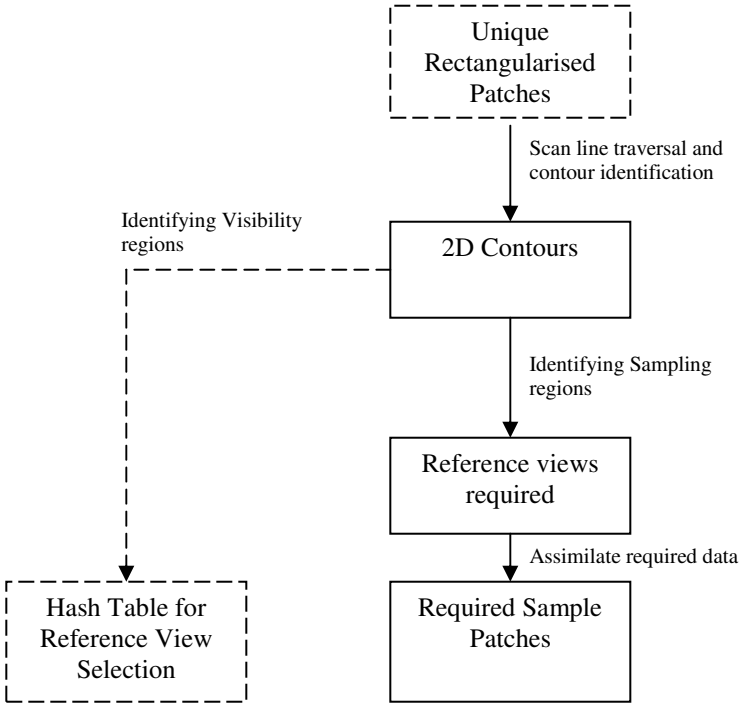


Figure 3.3: Contour Formation, Sampling and Visibility Graphs

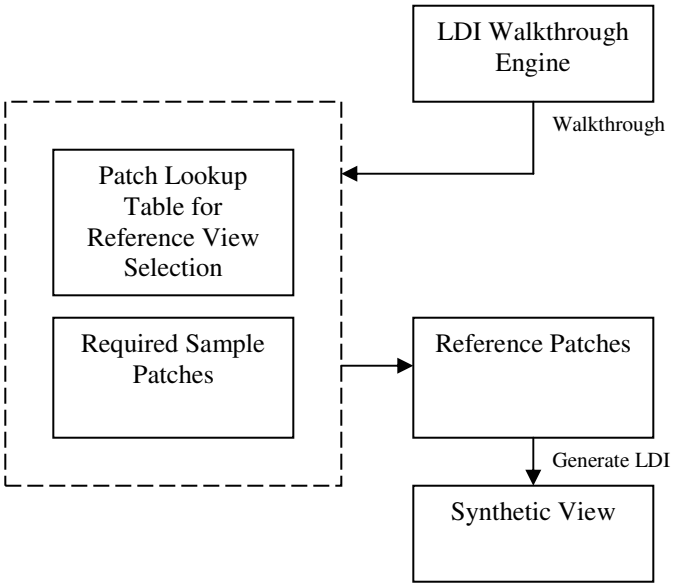


Figure 3.4: Re-rendering Engine

disposed. Figure 3.2 elaborates on the step of Patch Categorization and Figure 3.3 depicts the stage of formation of contours and the identification of sampling and visibility regions there on. Figure 3.4 highlights the effect of the adaptive sampling method on the final re-rendering engine. In the sections to follow, we shall go through each of these steps in detail, defining and discussing the theories and considerations.

3.2 Patch Categorization

This step marks one of the most critical steps in the method of adaptive sampling, as it's in this step that we start from just a point cloud to a representation, which though isn't as detailed as a triangle mesh, is still informative enough for us to understand the geometry of the scene and proceed to adaptively sample the object. It is necessary to clarify at this point that the patches thus formed are only for guiding the stage of sampling. The eventual rendering is still from the originally captured data.

3.2.1 Patch

Before going any further with the procedure of patch categorization, we explain the concept of a patch, in the context of data sampling. A patch is regarded as any uniform surface on the object (a surface without uneven bumps), which can wholly fit into the field of view of the camera under consideration.

The purpose of defining a patch is to be able to summarize the geometry of the object in a plane in 2 dimensions, so as to ensure that we get a rough sketch of the uniform sections of the object. It's worthwhile to note that a crude way of expressing adaptive sampling is to say that more samples are needed for areas of the object which are not too uniform

(occluded by parts of the same surface or different surfaces), and lesser samples for those sections of the object which are fairly smooth. Hence the need to be able to clearly distinguish these various sections of the object.

3.2.2 Patch Constraints

Having gone through a layman's definition of a patch, and its purpose, in this subsection we attempt to formally define a patch. A patch is formally defined as all neighboring points in the point cloud which satisfy the following constraints: (for more information on the point cloud and the attributes of the points there-in, please refer to the next chapter, section 4.3, titled Other Issues)

a) The normals between any two neighboring points in the spherical co-ordinate system don't differ by a preset δn_φ and δn_θ.
b) The normals between the extreme two points of the patch, in the spherical co-ordinate system don't differ by more than a preset Δn_φ and Δn_θ.
c) The Z values of any two neighboring points don't differ by more than a preset δ_z. This ensures that areas on two objects which have smooth transition, but are placed apart in the viewing direction don't end up being called a patch
d) The size of a patch both horizontally and vertically never exceeds the maximum size that the field of view, θ, of the camera permits at that depth. This is calculated as follows:

Taking a top view and denoting the size of the patch in any scan line with an edge, let the first and last points of this edge be S_{max}, S_{min}. Suppose the orthogonal bisector of the

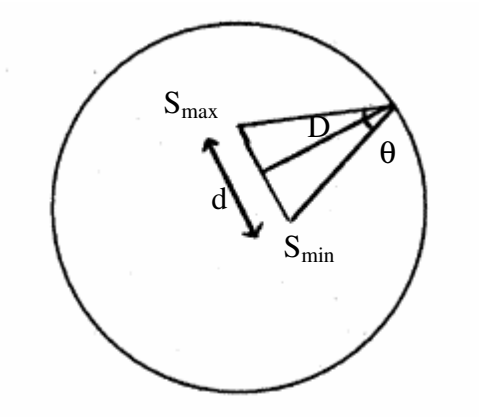


Figure 3.5: Patch Size Constraint (top view)

edge intersects the *sampling circle* at a point, and let the distance from the midpoint of the edge to the circle be denoted as D. (The sampling circle will be explicitly defined in section 3.3.2)

The size d of the patch is:

$$d = 2D \tan(\theta/2) \quad \textbf{(3.1)}$$

The same criteria applies for the vertical extent, with the corresponding angle, φ. Given the point cloud and given these constraints, the patches more-or-less as defined in section 3.2.1 are obtained and the whole point cloud is categorized.

3.2.3 Rectangularisation of Patches

We stated at the end of the previous subsection that we have categorized the point cloud into patches, which are "more-or-less" as defined earlier. The reason why these patches are still not exactly the way we defined is because, though the patches are uniform and if seen from the top or bottom look like they occupy only 2-dimensions, they still have a

non-uniform shape in the 2-dimensional plane. We hence attempt to break down these patches obtained, into patches of shape that can be easily summarized in one dimension, as a line. We choose a rectangle for our convenience and for the fact that it can be reduced to a line along a scan line.

In order to break the patches gotten so far, into rectangles, we follow a step called the rectangularisation of patches. In this step, for each of the patches thus identified, we find the biggest rectangle that can fit into it, and cut that portion out. We follow the same procedure for the remaining area in the patch, until we are left with areas, which are smaller than a preset area. All the portions cut out from the patch in the process are rectangularised components of the patch, and hence the term rectangularised patches.

It can hence be seen that a single patch obtained by patch categorization may later end up as a few rectangularised patches.

3.2.4 Patch Merging

So far we have defined and discussed about how to theoretically get patches from the point cloud. Practically, storing the entire sampled point cloud in the program's heap may not be feasible, owing to the highly dense sampling. A simpler and more practical approach is discussed in this subsection.

We consider one reference image at a time, and find the patches from its point cloud. As the size of individual images is much smaller, the heap constraint is no longer applicable.

As patches are being formed, they are checked for overlaps with patches found from previous reference images.

Merging the patches is done by isolating the various vertical cross-sections, and concentrating on the patches that overlap. If we are to find two sections of a patch that need to be merged, we can be assured that they would have an overlap, owing to the high density of the samples. Care should be taken to make sure that the patch, as it's being merged, still satisfies the patch constraints, over the border and as a whole.

Redundant patch areas can be found and dropped, when patches are found which overlap by an extent more than the approximation involved in the step of rectangularisation. The section of the patch that is not redundant (which is now definitely smaller than the maximum size a patch permits) can be merged with some other patch if need be.

It's worthwhile to mention the effect of the sequence of the reference images, on the eventual set of patches generated. The patch merging process is applied in a linear fashion, with the patches being created in the current reference image, compared with the patches thus formed, to check for an overlap. This makes the efficiency of the merging process reliant on which reference image is considered next. The reduction in the data redundancy is independent of this decision; however the number of patches generated, is not. A recursive approach to make patch-merging independent of the sequence of the reference images is computationally expensive. Also, this is unnecessary in situations where there is a clearly defined sampling trajectory on which the reference images were captured. In our system implementation, the reference images are considered in the same order that they

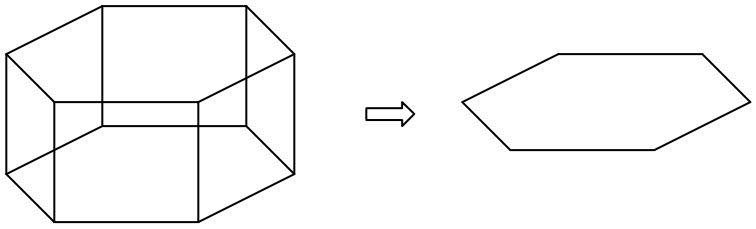


Figure 3.6: From a group of Rectangle Patches to a 2-D contour

were captured during the sampling camera traversal on the sampling trajectory. This ensures maximum overlap between consecutive reference images, given that our sampling trajectory was a circle.

3.3 Contour Formation, Sampling and Visibility Graphs

A contour, literally, stands for a 2-dimensional shape expressed as a line representation. Our definition isn't far from this meaning of contours. Having broken down the point cloud into rectangularised patches, we are now in a position to form a skeleton of the entire object.

3.3.1 Contour Formation

In the previous section, we discussed how we categorized the point cloud into a set of rectangle patches. In this section, we attempt to make a skeleton out of these uniform sections of the object.

Each patch obtained thus far, is labeled with a number. A scan line traversal is now performed to see the patches that are encountered at each scan line. At the end of this scan

```
procedure FormContours (patch [])
        for k←0 to patch.size-1          //find patch demarcations in the vertical direction
                demarcations.add(patch[k].min_y)
                demarcations.add(patch[k].max_y)
        end for
        sort(demarcations) //ascending order
        for j←0 to demarcations.size-1
                for k←0 to patch.size-1
                        if patch[k].min_y = demarcations[j] //new  patch starts at this demarcation
                                temp.add(patch[k])
                        else if patch[k].max_y = demarcations[j] //old patch ends at demarcation
                                temp.remove(patch[k])
                        end if
                end for

                if j = 0
                        prev_demarcation = -1000
                else
                        prev_demarcation = demarcation[j-1]
                end if

                /form a contour with the patches in temp, and applicable for y from prev_demarcation to the current
                createContour (temp, prev_demarcation, demarcations[j])
        end for
end procedure
```

Algorithm 3.1: Contour Formation

line traversal, we have a set of patches traversed for each scan line. We group all the scan lines with the same patch traversals together. Given these groups of patches, we try to make one contour to represent each of these groups. The contour is basically the representation of what we would see when viewed from the top. Each patch, since close-to-constant in the 3rd dimension, and a rectangle in 2 dimensions, can be represented as one line, as it would be when seen from the same plane in which it is present. Algorithm 3.1 depicts the pseudo code that summarizes the contour formation algorithm applied to the rectangularised patches.

We hence have distinctly formed contours, representing various vertical segments of the

object. Figure 3.6 depicts the formation of a contour from a simple group of rectangle patches.

3.3.2 Sampling Arc

Having broken down the point cloud into a few 2D contours, the initial problem of sampling now boils down to adequately sampling all the edges in each of these contours. In this context we shall define the concept of a sampling arc. For any contour, we attempt to sample the edges from the circumference of a circle, lying on the plane of the contour, with its center at the object's origin and a radius, which defines how close we can get to the object during camera walkthrough. We call this the *sampling circle*. We can have multiple concentric sampling circles for various resolutions.

For any edge, a *sampling arc* is defined as the arc of the sampling circle, such that from any point on that arc, the edge under consideration has maximum visibility. To understand the sampling arc better, we need to take a brief look at the concept of cameras and views.

Any camera has a view plane on to which any point seen from the camera is projected. When we see through the camera, or take images with the camera, what we see is a projection of the scene on to the camera's view plane. The number of pixels on the view plane does not necessarily have a one-to-one mapping with the number of actual points in the world coordinate system.

The sampling arc of any edge can now be defined as the arc of the sampling circle, defined by all those points on the circle's circumference, from which the geometric

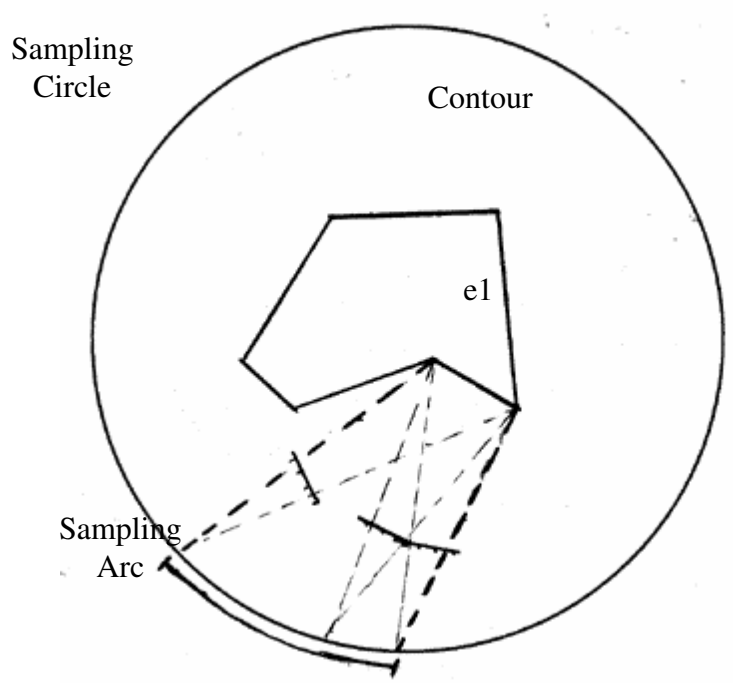


Figure 3.7: Sampling Arc

content of the pixels seen, as compared to what is seen from an orthogonal view of the edge, is unchanged. Figure 3.7 depicts a contour, the object's sampling circle and the sampling arc of a particular edge labeled "e1" of the contour.

When an edge is sampled from a point on the sampling circle outside the sampling arc, we end up sampling fewer pixels than can be seen from points on the sampling arc. We call this phenomenon *oblique sampling*.

3.3.3 Determining the Sampling Arc

Having seen the definition and necessity of a sampling arc, we shall in this subsection see how to find the sampling arc, given an edge and the sampling circle.

Suppose we choose a Cartesian st-coordinate system for the sampling circle with the

center of the circle placed at its origin and the t-axis parallel to the edge. Before actually deriving the sampling arc, let us look at how the *edge depth* z, varies with respect to the camera motion, as depicted in Figure 3.8. We define the edge depth, as the distance between the midpoint of the edge and the camera placed on the sampling circle. The *prime sample point* (s_o,t_o), is defined as the point of intersection of the edge's orthogonal bisector and the sampling circle. For any given edge, its *prime depth,* z_o, is defined as the edge depth when the camera is at the prime sampling point. The ***edge depth function*** F_z, which is the edge depth as a function of the sampling point, can now be formulated as:

$$z' = F_z(s', t') = \begin{cases} \Delta s / \cos\left(\tan^{-1}\left((z_o - \Delta t)/\Delta s\right)\right), & \Delta s \neq 0 \text{ or } \Delta t \neq 0 \\ z_o, & \Delta s = 0 \text{ and } \Delta t = 0 \end{cases}$$

(where $\Delta s = s' - s_o$, and $\Delta t = t' - t_o$)

(3.2)

Now we are well equipped to derive the *sampling arc*, given any edge and the sampling circle. We have discussed before, while defining the sampling arc, that there is no one-to-one mapping between the actual points on the object and the pixels seen on the image plane. It is clear that the maximum visibility for the surface associated with the edge is obtained when seen from the prime sample point. However, as we would see further, this maximal visibility extends to a certain span on either side of the prime sample point, on the circumference of the sampling circle, giving us the sampling arc, where the best possible view of the edge can still be maintained.

Our aim is to find a sampling arc, such that the same pixel resolution as seen from the prime sampling point can be obtained for all points within the sampling arc. We observe

from Figure 3.7 that as we move away from the prime sample point, the number of points on the edge projected to a pixel on the view plane will reduce. Suppose, when viewed from the prime sample point, we establish clearly the largest segments on the edge which map to at most two pixels on the view plane, we would end up dividing the edge into several overlapping segments. Given that the length on the image plane, that these segments correspond to, keeps reducing as we move away from the prime sample point, we reach a stage where at least one of these segments corresponds to less than two pixels on the image plane. It is clear that, this is the point that defines the end point of the sampling arc. From Figure 3.8, we infer that the righter more the segment on the edge, the more its reduction of size on the image plane, upon moving left. Also, given that the right most and the left most segments of the edge, are the smallest, it is evident that one of these would be the first of the segments to correspond to less than two pixels on the image plane. We call these the *critical segments* of the edge.

Hence, the problem of finding the sampling arc boils down to finding the points on the sampling circle where the two critical segments occupy 2 pixels. Before going further, we need to acquaint ourselves with two camera dependent parameters. The pixel size Δp is the length of one pixel. Its corresponding edge segment is Δp_w. The camera distance f, is the distance between the camera's lens and the view plane.

We now term the ***pixel occupancy function***, F_W, as the number of pixels occupied by the critical segments. Given that an edge has a right and left critical segment, we have a right and a left pixel occupancy function, F_{WR}, F_{WL}. The sampling arc is thus determined by the

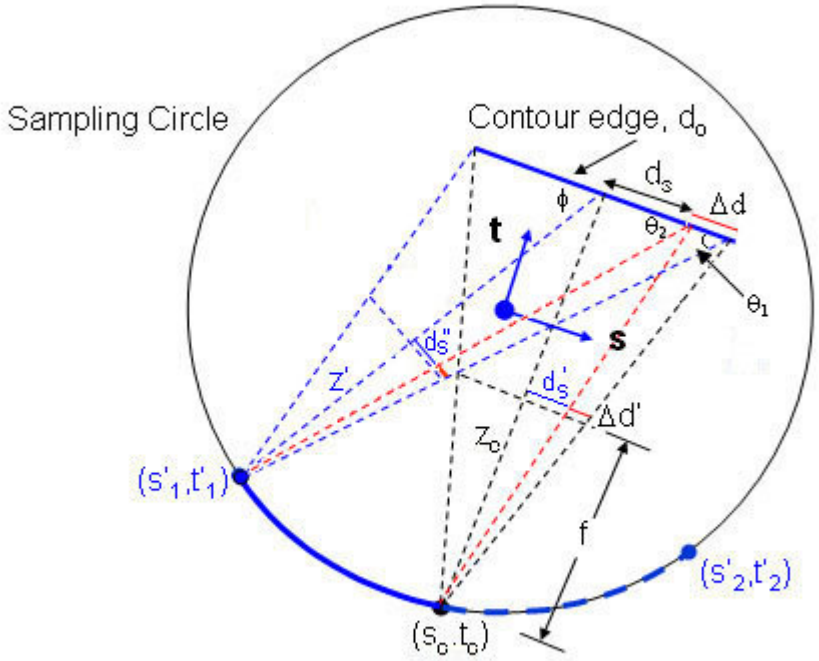


Figure 3.8: Sampling Arc derivation

set (s',t') such that F_{WR} (s', t') > 1 and F_{WL} (s', t') > 1. Solving for this, the left end point of the sampling arc is determined by the equation:

$$C_4t^4 + C_3t^3 + C_2t^2 + C_1t + C_0 = 0 \quad \textbf{(3.3)}$$

Where C_4 ,… C_0 depends on edge end points, R and f.

$C_4 = M^2 + K^2 + P_1^2 - 2KP_1$

$C_3 = 2P_1P_2 - 2MN - 2MK\ s_0 - 2KP_2 - 2MP_1s_0$

$C_2 = 2KP_1t_0^4 - 2KP_3 + 4KP_1s_0^2 + 2P_1P_3 + P_2^2 - 2K^2t_0^2 + N^2 + 2KN\ s_0 - M^2t_0^2 + 2NP_1s_0 - 2MP_2s_0$

$C_1 = 2MNt_0^2 + 2KM\ s_0t_0^2 + 2P_2P_3 + 4KP_2s_0^2 + 2KP_2t_0^4 + 2NP_2s_0 - 2MP_3s_0$

$C_0 = 2NP_3s_0 - N^2t_02 - 2KN\ s_0t_0^2 + K^2t_0^4 + P_3^2 + 4KP_3s_0^2 + 2KP_3t_0^4$

$K = (d_S - 2s_0)\ (2s_0f - fd_S - Z_0\ \Delta d')$

$M = 4fd_St_0 + 4fd_SZ_0 - 8f\ s_0t_0 - 4fZ_0s_0 + 2Z_0\ (Z_0+t_0)\ \Delta d'$

$N = 4fd_St_0^2 + 2fd_SZ_0^2 + 4fd_SZ_0t_0 - 4fZ_0t_0s_0 - 8ft_0^2s_0 + Z_0\ \Delta d'\ (t_0^2 + (Z_0+ t_0))$

$P_1 = -4ft_0\ (Z_0 + t_0)$

$P_2 = 8ft_0^3 - 2fZ_0^3 + 8ft_0Z_0$

$P_3 = (t_0^2 + (Z_0 + t_0)^2)\ (f\ Z_0^2 - 2ft_0^2)$

$Z_0 = ((t_1 + t_2)/2 - \sqrt{(R^2 - (s_1 + s_2)/2)})$

$\Delta d = d_0/10$

$d' = ((f/Z_0)\ \Delta d)$

$ds = ((d_0/2\ ) - ((Z_0/f\ )d'))$

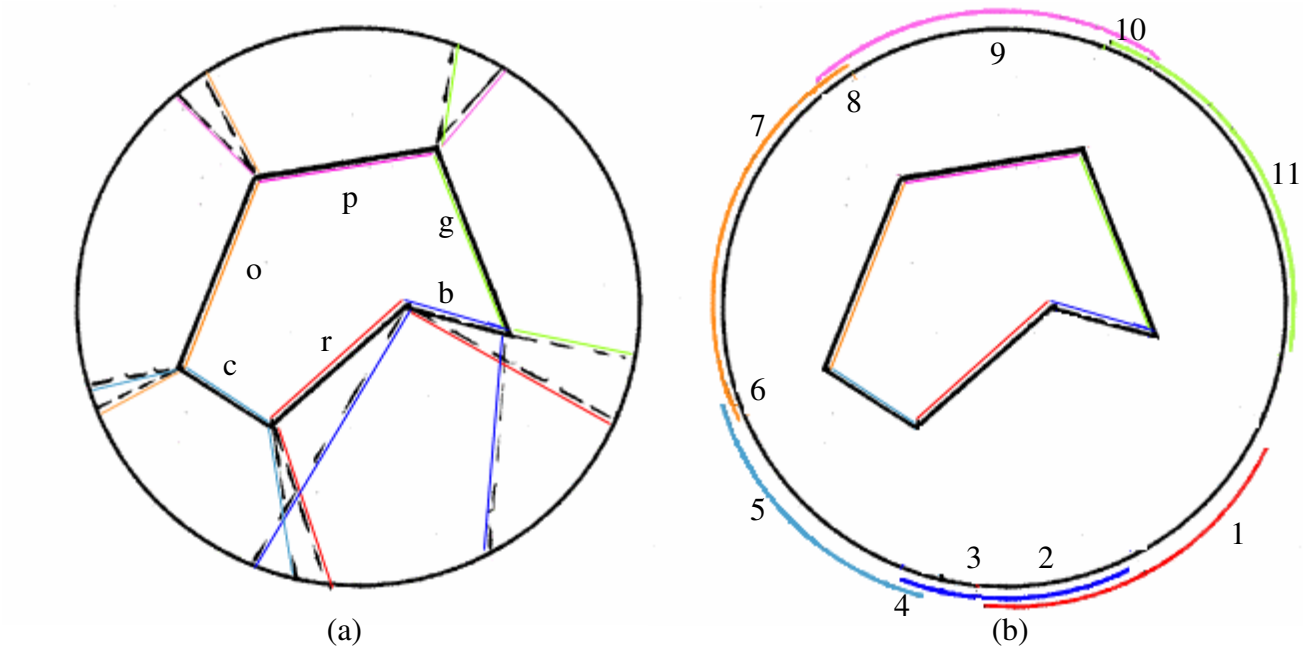


Figure 3.9: Sampling Graph: (a) Graph of the sampling arcs of all the constituent edges of the contour (b) Numbered arc segments formed, owing to overlaps of the sampling arcs

The reason why this is a four dimensional equation is because we haven't defined the visible side of the edge in our input and hence have sampling arcs on both sides, front and back. We then select the correct sampling arc from this. A detailed derivation can be found in Appendix A.

3.3.4 Sampling Graph

Having established how to find out the sampling arc, given any edge, we shall now look at how we use these sampling arcs to sketch the *sampling graph* and hence find the points to sample from. For all of the edges in the contour, we find the sampling arcs, as explained in the previous subsection, and transform them from their st-coordinate system to the world coordinate system. Figure 3.9 (a) depicts the sampling graph sketched by the sampling arcs obtained for all the edges.

Arc Segment	Associated Edges
1	r
2	r,b
3	b
4	b,c
5	c
6	c,o
7	o
8	o,p
9	p
10	p,g
11	g

Table 3.1: Tabulation of Arc Segments and their associated edges

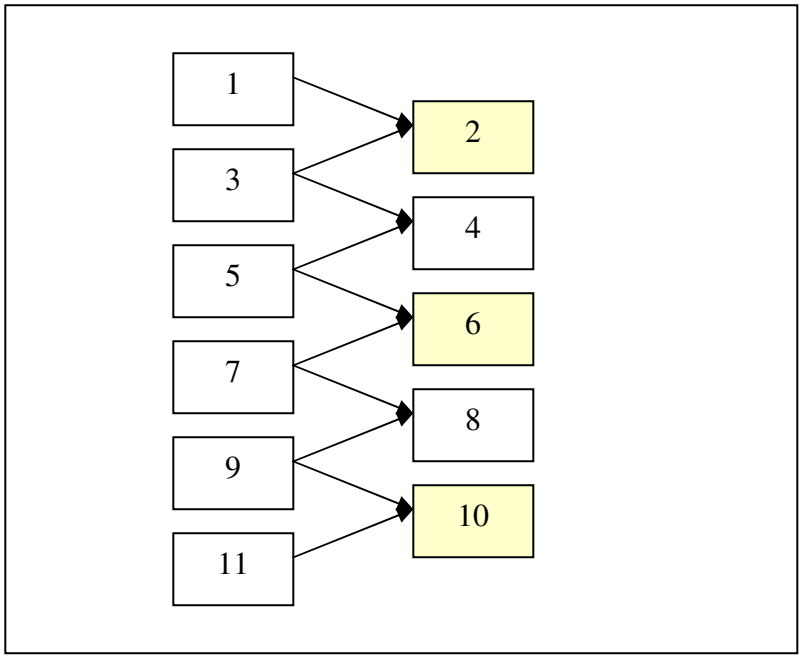


Figure 3.10: Directed Graph of arc segments

We then number the various arc segments formed on the circumference of the sampling circle, as depicted in Figure 3.9 (b). Next, we tabulate every arc segment against its associated edges in the contour, as in table 3.1. A directed graph is sketched with each of

```
procedure MakeSampleSet (criticalarcs [])
        for k←0 to criticalarcs.size-1 //for all critical arcs
          while criticalarcs[k].edges.size>0 //as long as some edge is yet to be sampled
            for j←criticalarcs[k].beginpt to criticalarcs[k].endpt
              for i←0 to 180
                for a←0 to criticalarcs[k].edges.size-1
                  calculate(pixel_occupancy for criticalarcs[k].edges[a])
                  //z', d/2 and θ3 values change with angle i and edge a
                  if pixel_occupancy satisfactory
                        edgescovered.add(criticalarcs[k].edges[a])
                        //this edge can be sampled from here
                  end if
                end for
                if edgescovered.length > max    //if this is the most optimum solution thus far
                  solution.remove(edgescovered)
                  solution.add(j, i, edgescovered) //point and angle to sample the edge set
                  max = edgescovered.size
                end if
              end for
            end for
            criticalarcs[k].remove(solution.edgescovered)  //these edges are sampled
          end while
        end for
        return solution
end procedure
```

Algorithm 3.2

these arc segments as nodes. Node-A directed to Node-B in the graph indicates that the edges associated with the arc segment represented by Node-A, is a subset of the edges associated with the arc segment represented by Node-B. Figure 3.10 depicts the directed graph plotted for the example in Figure 3.9.

The set of *critical arc segments* is the set of minimum number of arc segments required to cover all the edges of the contour. This is determined by considering the set of arc segments represented by the leaf nodes in the directed graph. We perform a greedy algorithm on this set of arc segments, by selecting one arc segment at a time, to maximize

the number of edges covered thus far. The result of this step is the set of critical arc segments, which in the case of the example depicted in Figure 3.10, is {2, 6, and 10}.

We now have a set of critical arc segments and the list of edges that can be sampled from each of the critical arc segments. The set of adequate samples for the contour are obtained by sampling from selected points on these critical arc segments, trying to maximize the number of edges sampled in a single sample, ensuring that all the edges of the contour are covered. This is illustrated by the pseudo code depicted by Algorithm 3.2.

3.3.5 Visibility Region and Visibility Graph

This subsection describes the generation of the patch lookup table for selection of reference views during camera walkthrough. We have seen that the whole sampled object is categorized into uniquely defined surfaces, and we have associated each of those surfaces with a particular camera point to sample from, or in other terms, a reference image. It is now straightforward that given a point in the camera walkthrough, the set of reference images needed to generate the required synthetic view can be obtained, if the surfaces that can be seen from that point are known. This is essentially what the reference view selection algorithm does. We shall elaborate this later in the chapter, but for the moment it has been established that it is valuable to know the surfaces that can be seen from any given point in the camera walkthrough.

Towards this aim, we define *visibility region* of an edge as the region in the plane of the edge, where at least some part of the edge is visible. As an edge can be seen from any

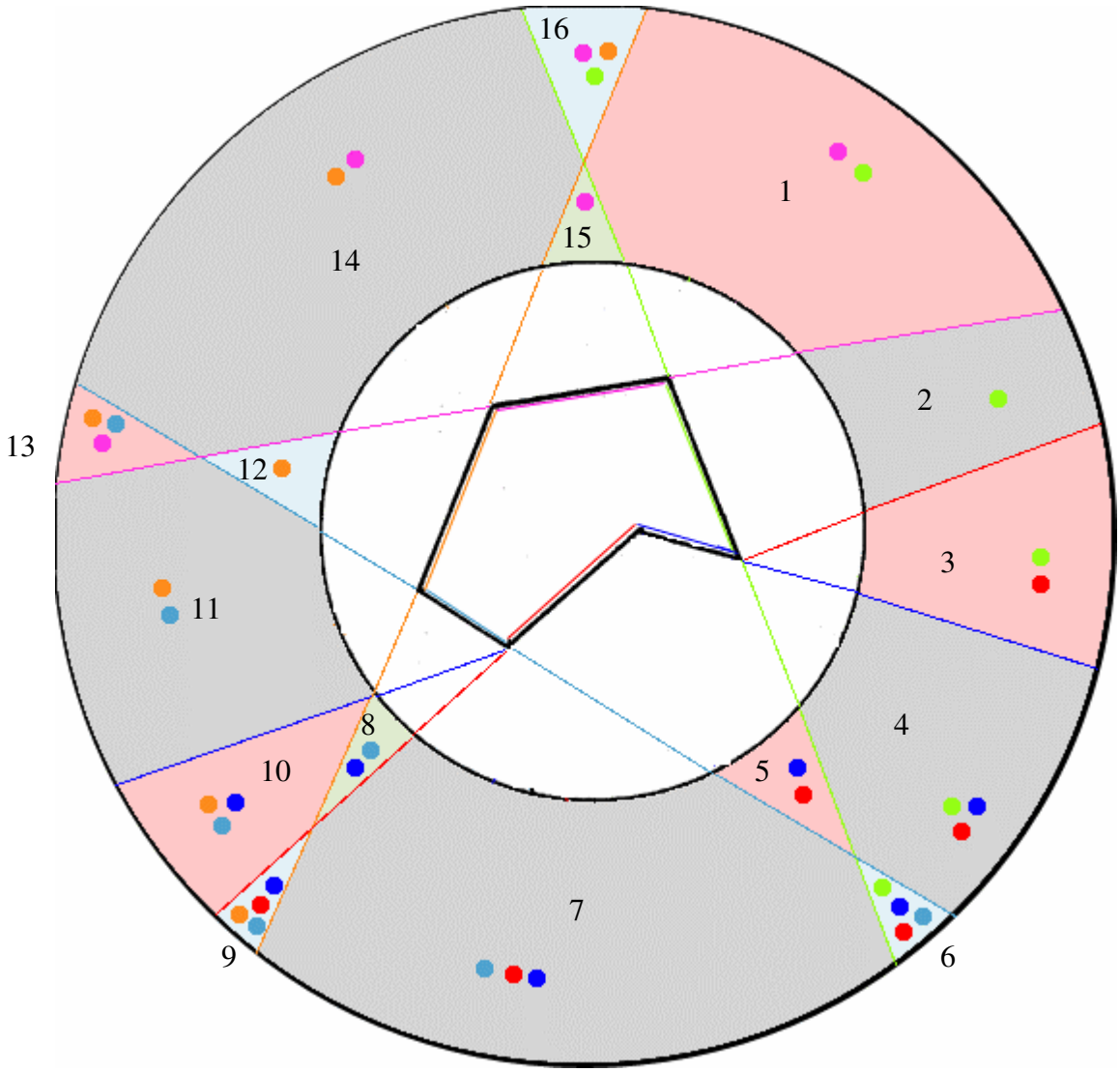


Figure 3.11: Visibility Graph: Graph sketching the visibility regions of all the constituent edges of the contour, highlighting the overlap of the various visibility regions thus formed

region in front of it, the visibility region for an edge is as depicted in Figure 3.11. A *visibility graph* for a contour is defined as the graph sketching the visibility regions of all the constituent edges of the contour, highlighting the overlap of various visibility regions.

We see from Figure 3.11 that a number of regions are formed, given any contour, with each region having a clearly defined set of visible edges. It must be noted that a region

radii index	angle index (o)	angles (i)	edge
1000, 2000	(40,165)	(45,150)	p
1000, 2000	(210,320)	(215,315)	b

Table 3.2: Patch lookup table (y = 10 to 18)

formed by the overlap of *visibility regions* of two edges, defines the region from which both the edges are visible. It might be worthwhile to note that, while sketching the *visibility graph*, we need not worry about the visibility regions inside the sampling circle, as by the definition of the sampling circle, this sets the boundary for the camera walkthrough.

Table 3.2, depicts a sample patch lookup table generated for the visibility graph. This table can be used to lookup the patches that can be seen from any point in the camera walkthrough (y', r', θ'), where y' is the scan line and (r',θ') are the spherical coordinates at that plane. The visibility arcs on the inner and outer bounding circles, define a visibility region for an edge. All of the regions depicted in Figure 3.11 were formed by the overlapping of these regions. In spherical coordinates, if the visibility arcs for edge e are defined as (r,θ_{i1}), (r,θ_{i2}), (R,θ_{o1}), (R,θ_{o2}), a point (r',θ'), during the camera walkthrough would be able to see all edges, whose index satisfy the constraints,

$r < r' < R$,
θ' lies between θ_{o1} and θ_{o2} , and
of θ_{o1} and θ_{o2}, let θ_{oc} be closer to θ', then
    if $|\theta_{o1} - \theta_{o2}| <= 180$, $h' >= h$
    if $|\theta_{o1} - \theta_{o2}| >= 180$,
        either $int(\theta'/90) <> int(\theta_{o1}/90)$ and $int(\theta'/90) <> int(\theta_{o2}/90)$
        or $h' < h$

    where h is $R.\sin(\tan^{-1}(r.\sin(|\theta_{ic} - \theta_{oc}|)/(R-r.\cos(|\theta_{ic} - \theta_{oc}|))))$
    and h' is $R.\sin(\tan^{-1}(r'.\sin(|\theta' - \theta_{oc}|)/(R-r'.\cos(|\theta' - \theta_{oc}|))))$

3.4 Rendering Engine

We have seen thus far, the proposed improvements and how it works. In this section we shall look at the rendering stage of the pipeline and how our sampling method improves the synthetically generated views.

The stage of rendering is marked by a camera walkthrough of a camera in a 3D space around/in a particular object/scene sampled prior to this stage. This step is broadly a composition of identifying the camera's position in the 3D space, identifying the reference images required to render the synthetic view at this point and finally generating the synthetic view from the reference images thus found. We shall briefly go through each of these sections constituting the re-rendering phase of the pipeline, throwing light on the effect of the new method of sampling.

3.4.1 Reference View Selection

Given the camera/user's co-ordinates, we need to generate a synthetic view that can be seen from that location. In generating the synthetic view, we rely on the data collected in the sampling stage. The process of Reference View Selection is to select the best set of reference images to adequately generate the current view.

Methods thus far include computations like closest reference images to the co-ordinates under consideration, approximation of geometrically adequate surfaces required and so on, which need to compromise rendering speed to allow for computationally expensive calculations during a real time walkthrough.

In our method, as the visibility graphs had been calculated from the 2D contours, during pre-computation, the process of Reference View Selection now boils down to a mere look up of the patch lookup table thus formed, based on the camera's coordinates, to elicit the patches visible from those co-ordinates.

3.4.2 LDI Generation

This is one of the very critical sections of the LDI rendering engine, as this is the routine, which handles the generation of our LDI, as the name suggests. The core of this routine lies in re-projecting the pixel from the reference patches to the LDI. There are a few points to be considered, though the basic concept involved is pretty straightforward. Our reference patches have a camera matrix (C_1, C_2 …), associated with them, which indicate the camera viewing cum perspective transformations. This matrix basically transforms a point in the world coordinates to a point in the screen coordinates. Similarly our Layered Depth Image stores its camera details in its own camera matrix (C_L), which essentially transforms any world point to the LDI's screen.

This implies that a pixel in a reference patch is transformed into its corresponding point in the world coordinates, by multiplying it with the inverse of the reference patch's camera, C_1. This point in the world coordinates when further multiplied by the LDI's camera matrix, C_L, would transform the point to the LDI's screen. This is exactly what our re-project routine does.

$$C_L * C_1^{-1} * [x_1, y_1, z_1, w_1]^T = [x_L, y_L, z_L, w_L]^T \qquad \textbf{(3.4)}$$

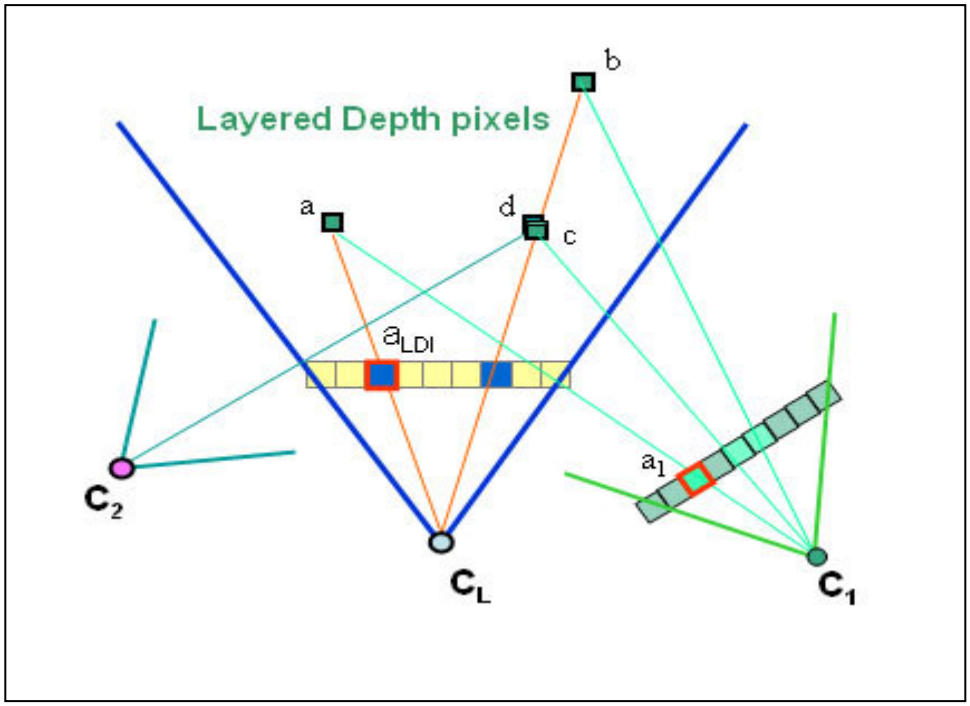


Figure 3.12: LDI re-projection and interpolation

The pixel on the input reference patch's image-plane taken by camera C_1, is x_1, y_1, z_1 and the coordinates of that pixel re-projected onto the LDI camera C_L's view-plane is x_L, y_L, z_L. Let us now throw light on certain considerations that ought to be taken care of during this re-projection. The re-projection basically gives the x, y and z values of the pixel on the LDI cameras view-plane. The x and y specify the pixel position on the view-plane. The z is the value we make use of, to find out an appropriate layer to associate this pixel with. Our values of z in the previous layers help us determine the layer number for this pixel. If we find a pixel getting re-projected to an x, y position that already has one or more pixels in some layers, we verify the z of this new pixel with those in the layers and create a new layer in an appropriate position for this new pixel. Now what happens if two pixels re-project to the same x and y and their z's map to the same layer number? In such

cases, we interpolate the two pixels and place the interpolated pixel in the position defined by that x, y and layer number.

Let us consider the example depicted by Figure 3.12 for a better understanding of this routine. Pixels a, b, c belong to reference patch taken by camera C_1. Of course these pixels are in the world coordinate system, and they were in different pixel positions on C_1's image plane. On re-projecting these world points to the LDI camera C_L, we find that a maps to an x and y position which has no pixels stored in it yet. So a goes to layer 1 of this x, y positions buffer. Similarly b goes to layer 1 of a different x, y positions buffer. Now when we re-project c, we find out that it maps to the same x, y position as b. We also notice that c's z is less than b's z, which implies that c should be in a layer ahead of b. Hence b is pushed to layer 2, and c is put in layer 1. Now consider another pixel d, taken from a reference image taken by another camera C_2. Now this pixel maps to the same x, y position as b and c, and the z of d is less than b but is almost same as c. In such a case what we do is that we interpolate the values of c and d and place it in the layer in which it originally was, i.e. layer 1.

3.4.3 Incremental Warping

Warping is the final stage of the LDI rendering engine. The Incremental warping is the routine, which handles the LDI system, between any two successive LDI generations. Before discussing what this routine does, let us look at the scenario without an incremental warping stage.

Basically we generate an LDI with our LDI generate method, and display the pixels in the current LDI for the walkthrough. What happens when we move our camera a little? The LDI might still be the same in terms of the LDI number and the reference images set, but yet the position and the layers of the various pixels would have changed. This implies that the LDI we have, despite having the same input, is no longer valid, and needs a re-projection for the current view. And like we have discussed, LDI generation is a computationally expensive routine, and calling it every time our walkthrough camera moves would make the rendering process quite slow. Hence, the need for an incremental warping computation.

The key to this incremental warping computation is the realization that the re-projection equation, when factorized, is nothing but a sequence of additions/increments to a base case. Essentially redundant calculations are avoided and additions are preferred over multiplications.

From **(3.4)**

$$(C_L) * C_1^{-1} * [x_1, y_1, z_1, w_1]^T = [x_L, y_L, z_L, w_L]^T$$

Let $T_{1,L} = (C_L) * C_1^{-1}$, so (3.3) becomes

$$T_{1,L} * [x_1, y_1, z_1, w_1]^T = [x_L, y_L, z_L, w_L]^T \qquad \textbf{(3.5)}$$

We know that for a particular x and y, we have a lot of z values depending on the number of layers in that layered depth pixel. For all these pixels, except the first one, the redundant calculation of $T_{1,L} * (x, y, z,w)$ can be avoided.

(3.5) can be hence written as,

$$T_{1,L} * [x_1, y_1, 0, w_1]^T + z_1 * T_{1,L} * [0, 0, 1, 0]^T = [x_L, y_L, z_L, w_L]^T$$

$$\rightarrow \quad \text{start} + z_1.\text{depth} = [x_L, y_L, z_L, w_L]^T \qquad \textbf{(3.6)}$$

Eqn **(3.6)** can be used for all the layers of a particular layered depth pixel, by using just a z_1*depth increment over the start calculated for the first layer, instead of re-projecting every time. Also, the computation expense of start can be minimized. We know that we would be following a particular scan line order to warp each of the layered depth pixels. Assuming we go from left to right, top to bottom, our layered depth pixels would be ones with a constant y and with successive x's for each row.

So,

$$T_{1,L} * [x_1+1, y_1, 0, w_1]^T = T_{1,L} * [x_1, y_1, 0, w_1]^T + T_{1,L} * [1, 0, 0, 0]^T$$

$$\text{(start for next layered depth pixel)} = \text{start} + \text{xiner} \qquad \textbf{(3.7)}$$

Eqns **(3.6)** and **(3.7)** are the core of the incremental warping computation. Depth and Xiner being constant throughout, all we need to do is calculate start once for every row by a matrix multiplication. Then calculate the re-projected coordinates for each of the pixels in that (x, y) position, by just adding the pixels z multiplied by depth to the start calculated. For the next layered depth pixel, start can be calculated by adding Xiner to the previous start. Of course, if we move to the next scan line, we need to calculate the start again.

One last thing yet to be discussed is the scan line ordering. We assumed during the

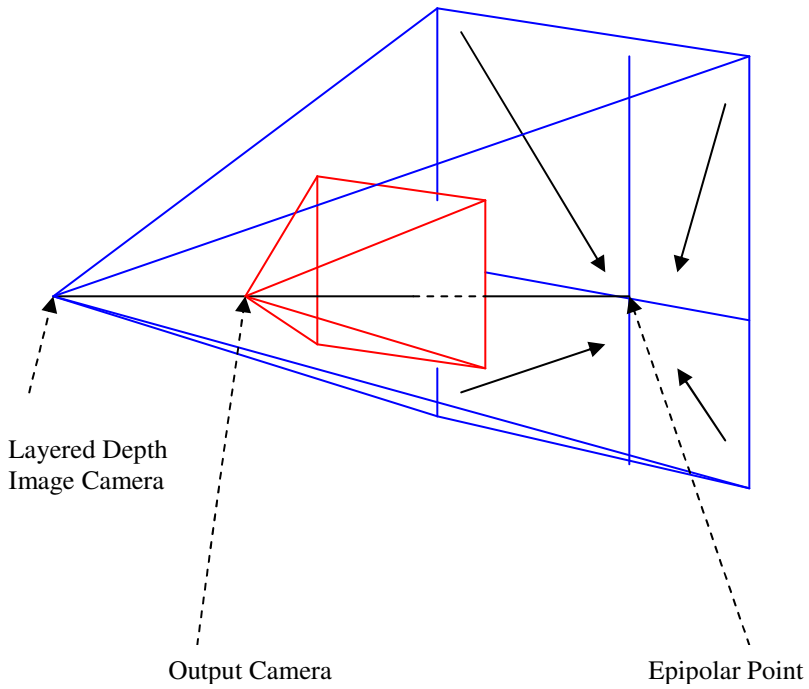


Figure 3.13: McMillan's ordering

previous discussion that we move from left to right and top to bottom. But that's just an instance of the many possible cases. For calculating and displaying the pixels in the new LDI, we use the McMillan's ordering algorithm on our old LDI [McMillan, 1995]. Essentially what this algorithm says is that, depending on the sign of w of the camera, the ordering of the pixels is determined either towards the epipolar point or away from it. The epipolar point is defined as a projection of the output camera's location in the input cameras view-plane, i.e. the intersection of the line joining the two camera locations with the input camera's image plane. The epipolar point might divide the view-plane into at most 4 quadrants. When w is negative the ordering is to move away from the epipolar point, starting from the pixel next to the epipolar point to all the way to the corner of the

quadrant and the view-plane, in a column wise, row based approach, in each quadrant. This is depicted in Figure 3.13.

3.4.4 Splatting and Hole Filling

Unlike in the previous methods, this stage, which is generally vital to the rendering pipeline, is absolutely unnecessary in our case. The stage of splatting is generally used to fill up all the holes that might and would be generated owing to the lack of adequate sample data. The misery shown in the amount of sample data owes to the fact that, more the sample data, slower the LDI Generation.

In our case, since the pre-computation has dealt with the problem of holes and occlusions by adaptively sampling the object based on its geometry, while re-rendering, we can be assured that we wouldn't be faced with that problem again.

3.4.5 Summary of the Rendering Engine

We have discussed in the previous subsections how the LDI is rendered and synthetic views are generated during the walkthrough. We noticed that our proposed method significantly changed the step of Reference View Selection and eliminated the Splatting. However, the lack of a significant difference in the remaining steps doesn't entirely mean that the output is not affected. Owing to the fact that the reference samples/patches found are exactly adequate, and that there's hardly any redundancy in the patches, the LDI Generation, which is in fact a re-projection of the reference images, is much faster and hence we have a faster walkthrough with much better quality.

3.5 Comparison

In the previous chapter, we had briefly gone through the related work in sampling for image-based rendering. In this chapter, we have thus far proposed and discussed our method. In this section, we shall compare and contrast our method with the related work so far.

3.5.1 Original LDI

In the original LDI method [Shade et al., 1998], the basic framework for rendering with Layered Depth Images was introduced. Though particular care was taken to render realistic synthetic views, it was largely a compromise between speed and realism. The following paragraph discusses this work and illustrates its shortcomings.

In this work, the reference images were depth images taken from a range scanner. A uniform sampling approach, with a fixed interval was used to accumulate the sample reference set. The re-rendering phase was marked by the creation of LDIs with a set of "proximal" reference images, from amongst the initially sampled set of images. The visual artefacts owing to inadequate/occluded data or a difference in resolution between the synthetic view to be generated and the sampling rate of the reference images were approximately covered by a splatting technique.

The problem with this approach was that the reference view selection was not complete. There was no absolute definition of a proximal set of reference images. The LDIs hence created weren't substantial, in quality. The step of splatting did cover up most of the holes

but with possibly incorrect data. This step also accounted for much of the render time computational time.

Our method excels this work by defining the completeness of the reference set of images required for rendering. The speed of selection and the elimination of the step of splatting ensure a real time rendering. The availability of adequate information and the lack of splatting improve the quality of the synthetic views.

3.5.2 LDI Tree

Of the two problems of holes and gaps resulting because of inadequate sampling, the LDI Tree implementation [Chang et al., 1999] solved the issue of gaps. This implementation used a hierarchical partition scheme, which preserves the sampling rate of the reference images by adaptively selecting an LDI from the LDI cluster for each pixel. The various reference images and sections of the LDI were categorized into an LDI Tree, which could be looked up to the depths dictated by the current sampling rate. Essentially what this means is that, samples of a particular scene at different resolutions are collected, and depending on the resolution required during camera walkthrough, the appropriate sample is chosen.

However, since nothing much was done to prevent holes, the sets of samples at each resolution were inadequate just like in the original LDI.

We tackle the issue of gaps by sampling from not just one sampling circle, but a set of concentric sampling circles. The issue of gaps and hence the sampling rate of the synthetic

view need not be taken care of, as it is implicitly solved in our method. The visibility regions are mapped to the patches sampled, and this information is stored in the hash tables. At run time, it is irrelevant which sampling rate is currently required, as the current position clearly defines a visibility region and hence the patches required. Hence we see that the patches of the correct sampling rate are seamlessly used in the rendering pipeline. Also, since there's no explicit difference in the approach for different sampling rates, the theory of adequacy of the samples at a particular sampling rate implicitly applies throughout.

3.5.3 Best Next View

This approach aims to select the next view for the sampling system, given some already acquired views of the object. Two criterions often considered in solving this problem are the visibility criterion, which attempts to maximize the number of surfaces not seen thus far, and the quality criterion, which aims to improve the quality of the surfaces sampled.

A volumetric representation, termed as the voxelmap, is generated at each cycle of best next view computation in one such work [Massios, Fisher, 1998]. The voxels thus scanned are marked empty, seen or unseen depending on the visibility from the new view. The seen voxels carry a quality property, which is estimated by the aggregate normals of all the points sampled in a particular voxel.

This method relies heavily on the set of steps taken previously, to an effect, working like the greedy algorithm. A wrong choice by the heuristic at one stage would imply an inefficient solution. Also, since the quality criterion encourages the inclusion of reference

images until a particular threshold is reached, there's no check on the redundancy whilst sampling. Also, the selection of an image owing to some visibility criterion only means that it has the most number of surfaces not seen thus far. It doesn't however dictate that the surfaces seen thus far are not present in this image, which all the more adds up to the issue of redundancy.

On the contrary, in our method, unlike a view centric approach, we consider the whole picture before deciphering which samples to use for rendering. This eliminates the problem of incorrect intermediate steps of a greedy algorithm. Also, since the overlap between the patches is minimal and most of it is eliminated during patch merging, the issue of redundancy is almost non-existent. The process of finding the critical sampling arcs ensures that the sample data collected for the stage of rendering is minimal.

DESCRIPTION OF SYSTEM IMPLEMENTATION

We have discussed the proposed improvements to the LDI system in the previous chapter. Before going through the results and improvements seen with this new method, we shall, in this chapter, go over the various components used for the system implementation. We shall also briefly describe some of the issues dealt with, during the system implementation.

4.1 Hardware Components

One of the important hardware components required was the Range Scanner, a camera that samples not just the color at each point, but also the depths. The scanner used in our case was the Minolta Vivid 900, which samples a color map and a range map at the same time for any given viewpoint.

Given the bulky nature of the Range scanner, it was found more meaningful to move around the object whilst sampling, than the camera itself. Under the assumption that uniform lighting conditions prevailed and that no specular reflectance was observed on the surface of the object being sampled, this was a valid alternative. Towards this effect, a mount was built and placed over the turntable to hold the object to be sampled. This apparatus has two degrees of freedom; rotation about a vertical axis with precision of ±0.2 degrees and vertical translation with a precision of ±0.2 cm. Figure 4.1 illustrates this apparatus and the setup used for sampling.

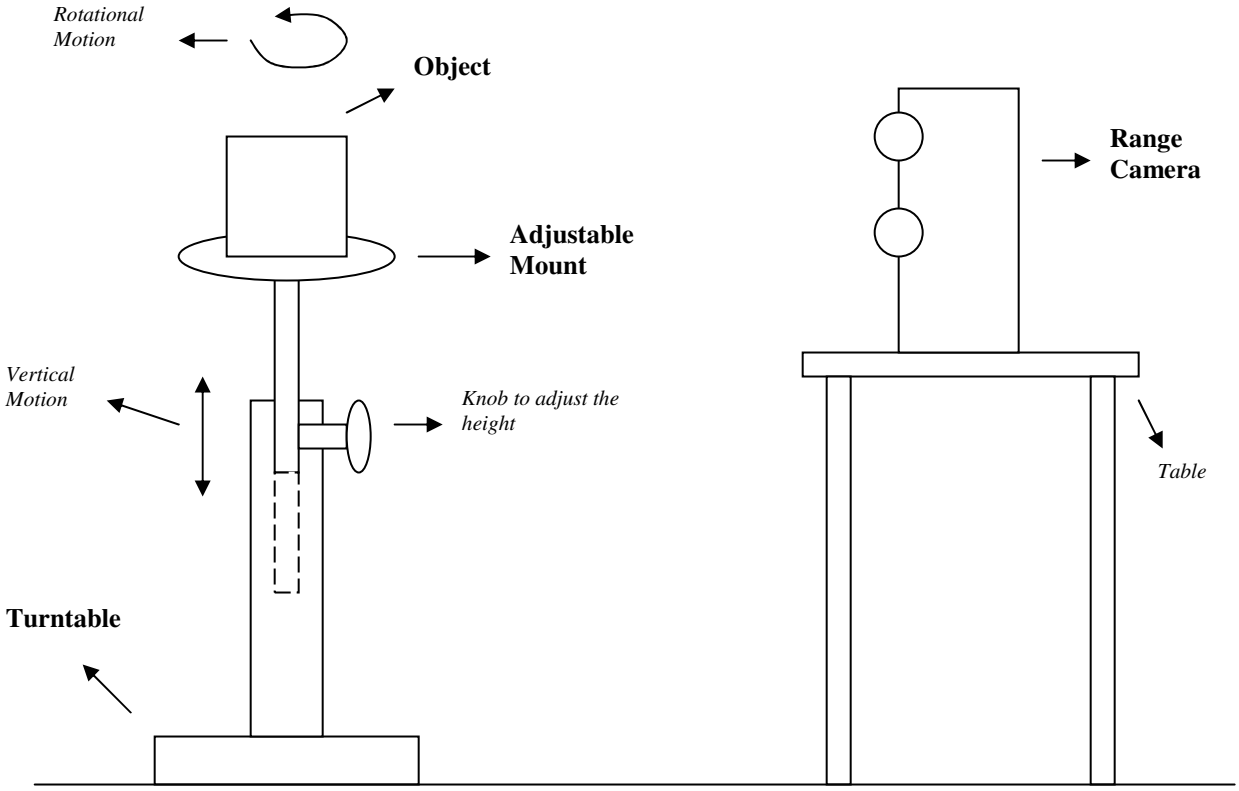
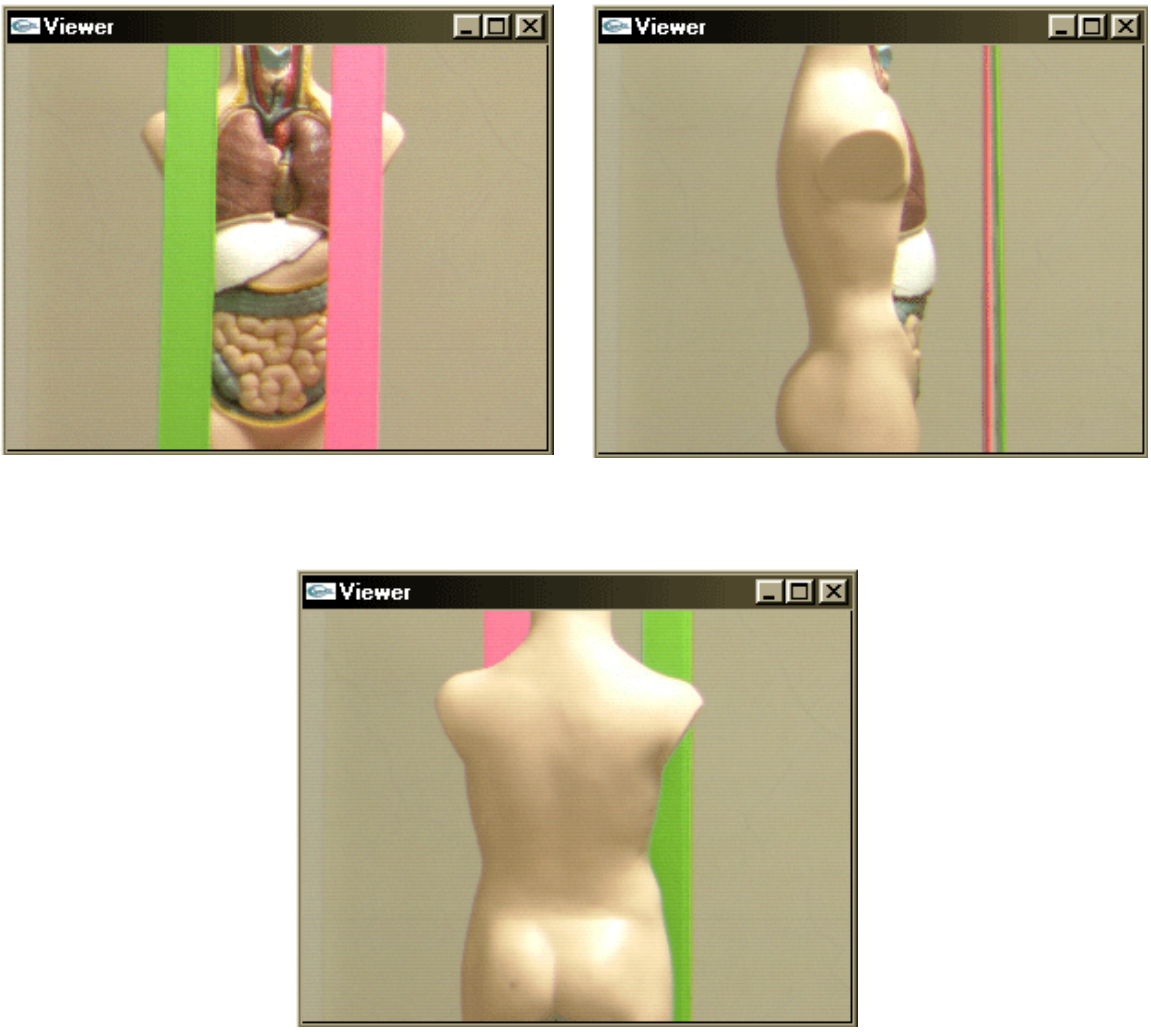


Figure 4.1: Set-up for sampling

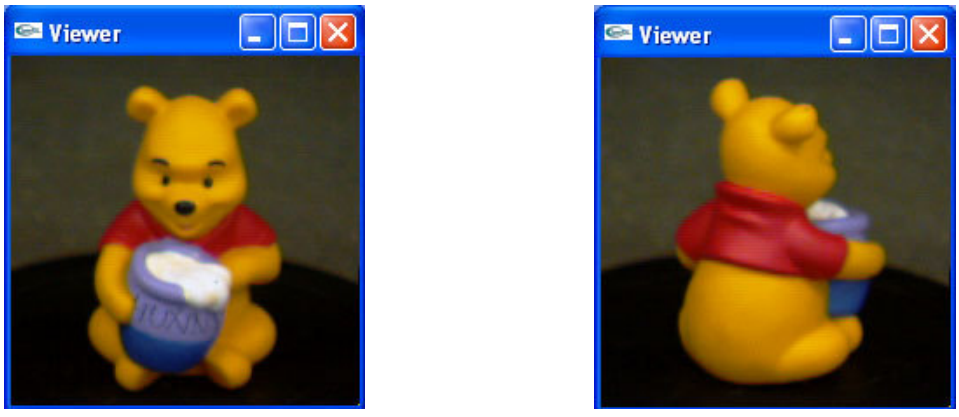
The object, the final part of the sampling setup, was so chosen to illustrate the ability of our proposed method to handle occlusions and non-uniform surfaces to great precision. The object used was a mannequin, placed behind two vertical rods which catered for most of the occluded regions. The existence of several non-uniform surfaces on the mannequin was also noticeable. The system was also tested on a simpler object, a pooh bear, for comparison. Figure 4.2 illustrates a few snapshots of these objects.

4.2 Software Components

The program was developed in Open GL, with C++ using Microsoft Visual C++ development environment. The object was scanned by the Range scanner, and the



(a)



(b)

Figure 4.2: The sampled objects (a) Mannequin (b) Pooh Bear

scanner software output format was converted to a simpler color map and depth map format. The color map was represented as Portable Pixel Map images (.ppm files), while the range map was represented using text files (.txt files). A one to one correspondence could be found between the (r,g,b) pixels in the color map and the (x,y,z,flag) surfels of the range map.

The camera, often taken for granted in most rendering systems, constitutes one of the key components of the rendering engine. We represent the camera as a 4x4 transformation matrix, such that given a Camera placed at a point c_1 and given its representation as matrix C_1, any point in the global co-ordinate system could be re-projected into the camera's view plane by a simple matrix multiplication between the point co-ordinates and the camera matrix C_1.

4.3 Other Issues

In this section, we shall look at some of the issues involved in the system implementation.

4.3.1 Range Map

Each range image file has a three-line header giving the number of rows and columns in the image. This is followed by four image sections. The first is the so-called 'flag' image, where a pixel value of 1 means the corresponding (x, y, z) values at that pixel are valid. If the flag value is zero, the (x, y, z) components for that pixel are ignored. Following the flag image is the image of X-coordinates, the image of Y-coordinates, and the image of Z-coordinates. All are floating-point images. The X and Y images are required only when calculating the normals. The Z-values are the range/depth under consideration. A very

high value was used for the depth, in cases where the pixels don't correspond to the object but to some background. However we don't look at these values owing to the flags section.

4.3.2 Surface Normals

The surface normals are not extracted from the object but are found from the range maps obtained. The surface normal is defined as the 3D vector, originating at the pixel under consideration and orthogonal to the surface at that point.

The algorithm written, essentially takes in as input a range map and returns the surface normals for the width * height pixels in the range map. The basic idea of this algorithm is to approximate the normal at any pixel position by averaging the 8 normals that correspond to the surrounding 8 neighboring surfaces. To better understand this, let us consider Fig 4.3.

Let X be the pixel for which we are trying to find the surface normal. Points 1-8 are the neighboring pixels. We aim to approximate the normal at X, by averaging the normals of the surfaces depicted by labels A-H. The normals of these surfaces A-H can be calculated by the knowledge of the 3D coordinates of the triangle's vertices describing the surface.
For instance, the normal of surface A can be approximated to the cross product of the vectors X->1 and X->2. By similar cross product calculations, the normals of surfaces A-H are estimated. The Normal N, at X is approximated to the average of these 8 surface normals. The N is then normalized.

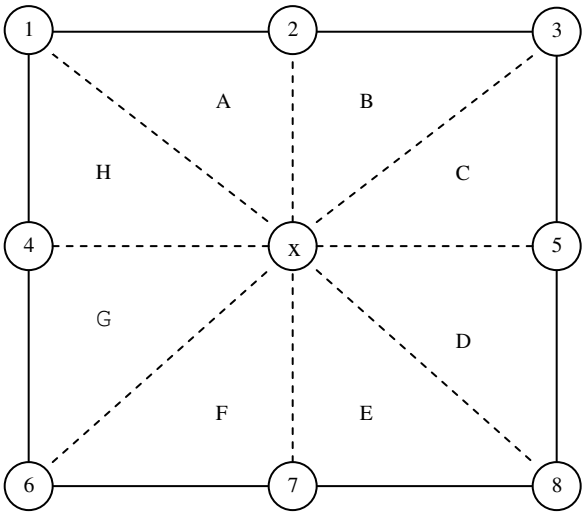


Fig 4.3: Surface Normal Approximation

The basic idea is to consider the image plane to be placed horizontally, with vertical lines coming out of each pixel, indicating the depth values at the corresponding pixels. Now if we place a cloth on these lines, we can see a surface that's formed by these depth values. Essentially what we are trying to do in the normal calculation is to estimate the normals of the 8 surfaces formed by the cloth around one line, and average them out to find the normal of the pixel under consideration.

There is one consideration that ought to be discussed in approximating the surface normals. The 8 neighboring points may not exist or even if they do, they may not lie on the same surface. Essentially we are talking about cases where a few of these 8 pixels lie on the edge of a surface or are on a whole other surface altogether. To take into account such cases, we keep track of the normals involved, by making use of the angle the normal makes with the normal of the image plane. The normals are then sorted based on these angles. The extreme cases, where the angles differ a lot from the rest of the normals, imply

that they are the normals for an edge or for a different surface. In such a case, we discard those points and don't involve them in calculating the normal.

The normals thus calculated are converted back to a global coordinate system. Suppose the surface normals are calculated from the range map corresponding to the camera C_1, the normals in the global co-ordinate system would be:

$$\mathbf{n_w = C_1^T\, n_s} \qquad \textbf{(4.1)}$$

(this is so because, given any point p in the image's viewing co-ordinates, by definition,
$n_s^T p_s = 0 \quad \Rightarrow \quad n_s^T (C_1\, C_1^{-1})\, p_s = 0 \quad \Rightarrow \quad (n_s^T C_1)\,(C_1^{-1}\, p_s) = 0 \quad \Rightarrow \quad \mathbf{(n_s^T C_1)\, p_w = 0}$ **-- (1)**

where, p_w is the point in world co-ordinate system.
But by definition, in the world co-ordinate system,
$\mathbf{n_w^T p_w = 0}$ **-- (2)**
from **(1)** and **(2)**, $n_w^T = n_s^T C_1 \quad \Rightarrow \quad n_w = C_1^T n_s$)

4.3.3 Uniform Dense Sampling

Most image based rendering engines have the problem of striking a balance between excessive sampling and inadequate object information. These are the problems posed with uniform sampling. A dense uniform solution would ensure that none of the original data are lost, but would result in an extremely slow walkthrough. We have the luxury to sample the object densely, as we are going to filter out what's redundant and not required. Hence, our first stage of sampling is a very straightforward dense sampling approach.

We place the object over a turntable, as depicted in Fig 4.1, earlier in this chapter, to ensure an accurate rotational motion to ±0.2-degree precision. The mount underneath the

object helps the object to move up and down to an accuracy of ±0.2cm. Together, these two motions simulate the positioning of the camera anywhere on a cylindrical surface surrounding the object.

For the sample implementation, we used the MIDDLE Lens and sampled along only one sampling circle. But as we discussed in section 3.5.2, we can sample along various concentric sampling circles, to ensure that we only use up those patches while re-rendering which correspond to the sampling rate of the current camera view. Samples taken with a TELE Lens along a sampling circle close to the object, and with a WIDE Lens along a sampling circle far from the object, can be added to the existing sample set, to further enhance the rendered output and frame rate.

RESULTS AND DISCUSSIONS

In the previous chapters, we have described the proposed system and its sample implementation. In this chapter, we shall analyze the results and improvements seen with this proposed system and compare the results thus obtained, with those of other systems.

5.1 Results

The results observed from the sample implementation are promising. As can be seen from Figure 5.1, the quality of the rendered output is comparable to current systems using splatting. No holes are observed, and unlike splatting, the rendered output is completely a result of the sampled data, and not of any interpolation or synthetic approximations.

The rendering speed can be qualified as quite fast, as a real time walkthrough shows no signs of processing lag. Such a high speed would have been impossible, if all of the data initially sampled were to be retained for rendering the synthetic views during the camera walkthrough. With our method, we retained the adequate data to ensure the high quality of the rendered output, and disposed the redundant data, to ensure the high real time rendering speed noticed in the camera walkthrough.

Figure 5.1, depicts some snapshots taken during the camera walkthrough of our system for two objects – (a) a mannequin and (b) pooh bear[†].

[†] Sparse Pooh-bear samples were downloaded from the web, and subsequently morphed to synthetically create dense samples of the same. The poor quality of the Pooh bear output largely owes to this process of interpolation during morphing and the inadequacy of original data.

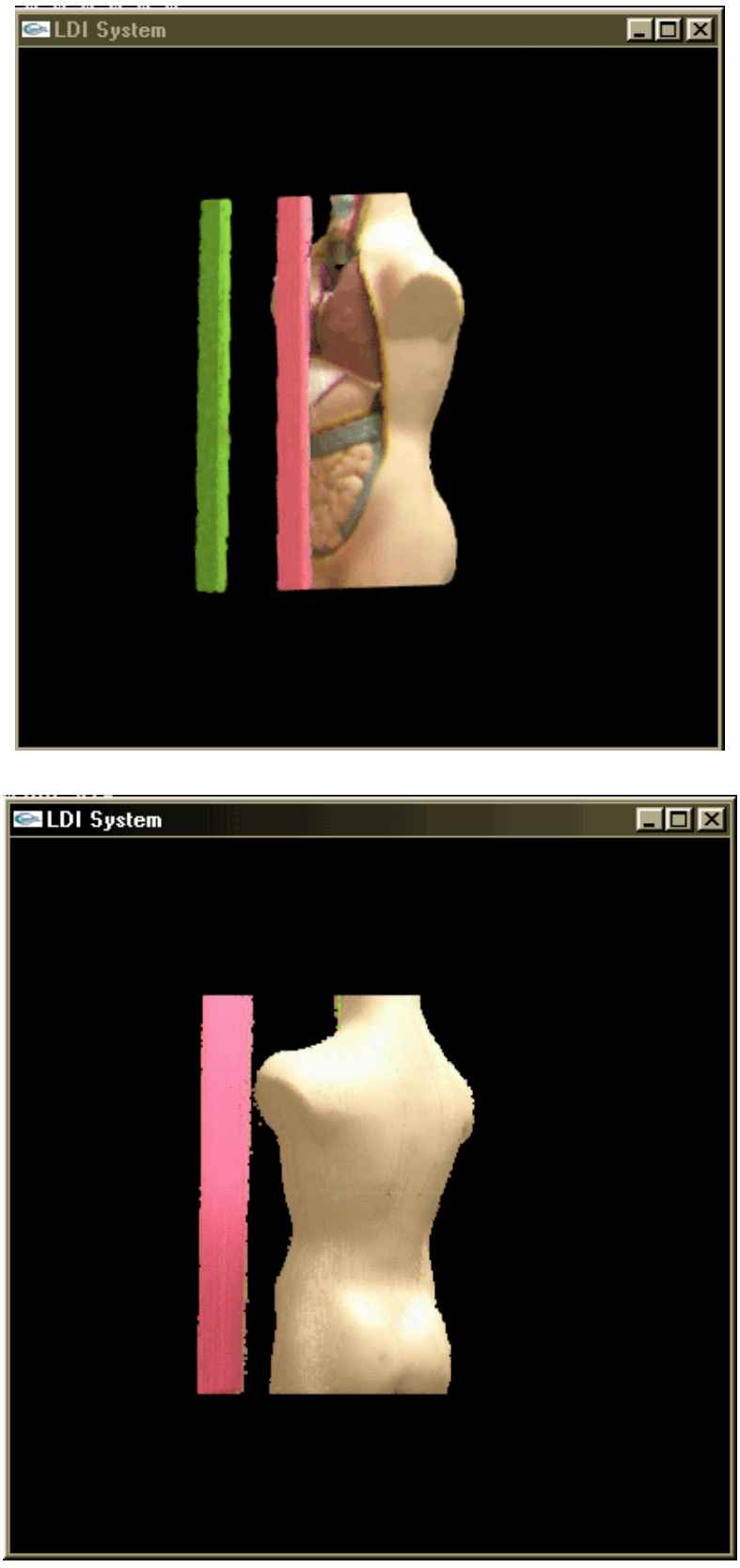


Figure 5.1 (a): Synthetic Views generated by the improved system

Figure 5.1 (b): Synthetic Views generated by the improved system

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Adaptively filtered set of patches from the sampled set of reference images of 296x222 resolution, sampled along a circle around the object, at regular intervals of 2.82^o Size: 97.3 Mb
Frame Rate:	10.7 fps
Patch loading time:	Proportional to the size of a patch. Typical Patch size: 2 Kb Time to load a patch: 0.71 milliseconds

(a) Mannequin Object

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Adaptively filtered set of patches from the sampled set of reference images of 200x200 resolution, sampled along a circle around the object, at regular intervals of 2.82^o Size: 44.1 Mb
Frame Rate:	13.7 fps
Patch loading time:	Proportional to the size of a patch. Typical Patch size: 2 Kb Time to load a patch: 0.71 milliseconds

(b) Pooh Bear Object

Figure 5.2: Statistical information for the improved system

Figure 5.2 shows the frame rate observed during the camera walkthrough and other statistics of the system.

5.2 Comparison

In this section we shall compare the results discussed in the previous section with a sample implementation of the original LDI system [Shade et al., 1998], with uniform sparse sampling (with and without splatting) and uniform dense sampling.

5.2.1 Uniform Sparse Sampling

With a uniform sparse sampling, with reference images taken along a circular orbit around the object, at regular intervals of 19.74 degrees, the following results were obtained. It is observed from Figure 5.3 (a) that the quality of the rendered views is much better in the improved system when compared to that of the sparsely sampled non-splatted system. In the case of the splatted system, very little difference is noticed, as both the systems have no holes. However, while the improved system's rendered output is a result of the original sampled data, the same can not be said about the latter, which is noticed on close observation. In case of Figure 5.3 (b), where the pooh-bear object is not as complex as the mannequin object, the difference in quality is still noticeable, though not as much as in the former case. This implies that in case of objects with large portions of smooth uniform surfaces, adaptive sampling wouldn't have a clear advantage over sparse sampling, as most sections of the surface are covered even by a uniform sparse sampling system.

In case of both the examples, the speed of the improved system is comparable to the sparsely sampled non-splatted system, while it is better when compared to the splatted version. Figure 5.4 shows the frame rate observed during the camera walkthrough and other statistics of the sparse system. Table 5.1 at the end of this chapter illustrates the statistical difference between these systems.

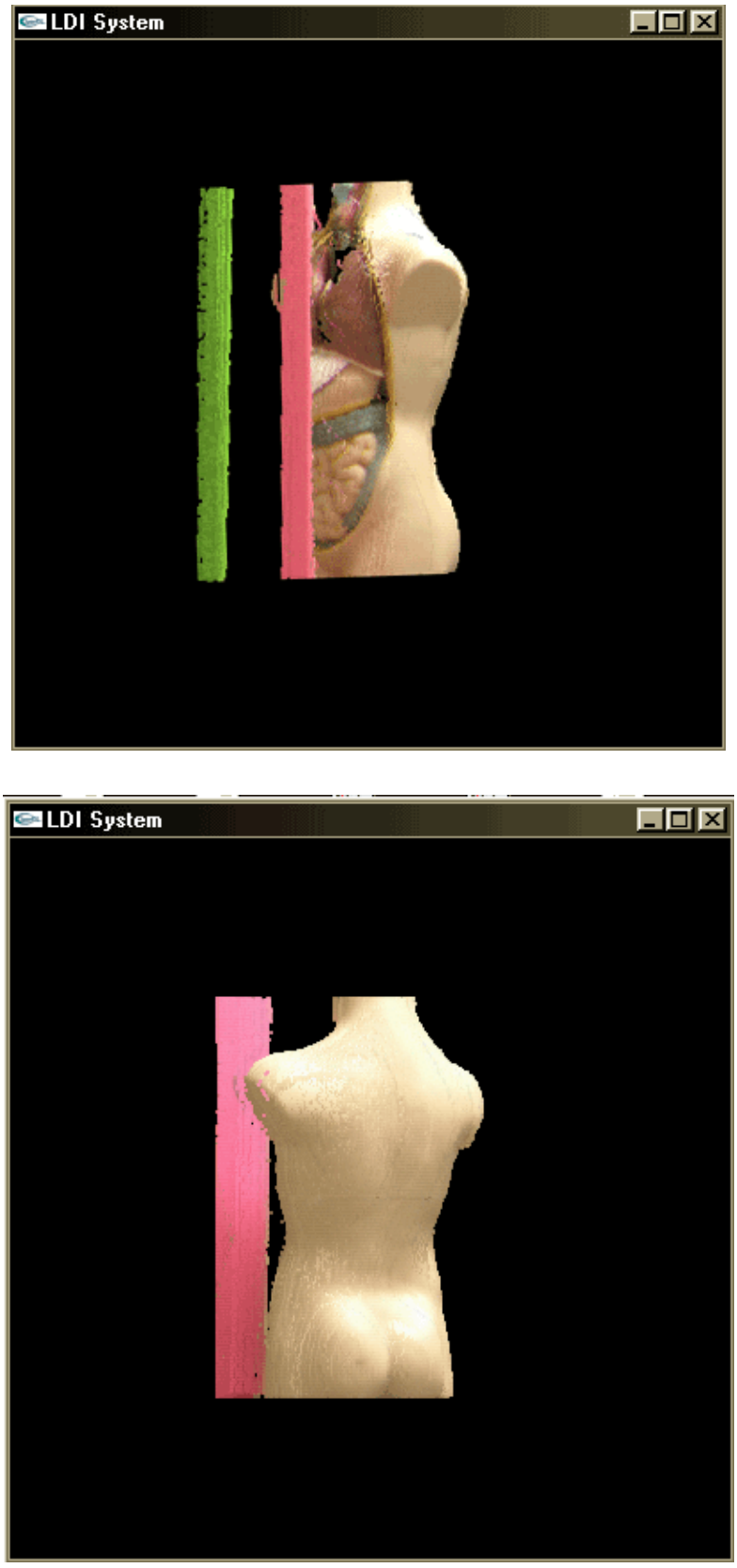


Figure 5.3 (a): Synthetic Views generated by the sparsely sampled LDI system, without splatting

Figure 5.3 (a): Synthetic Views generated by the sparsely sampled LDI system, without splatting

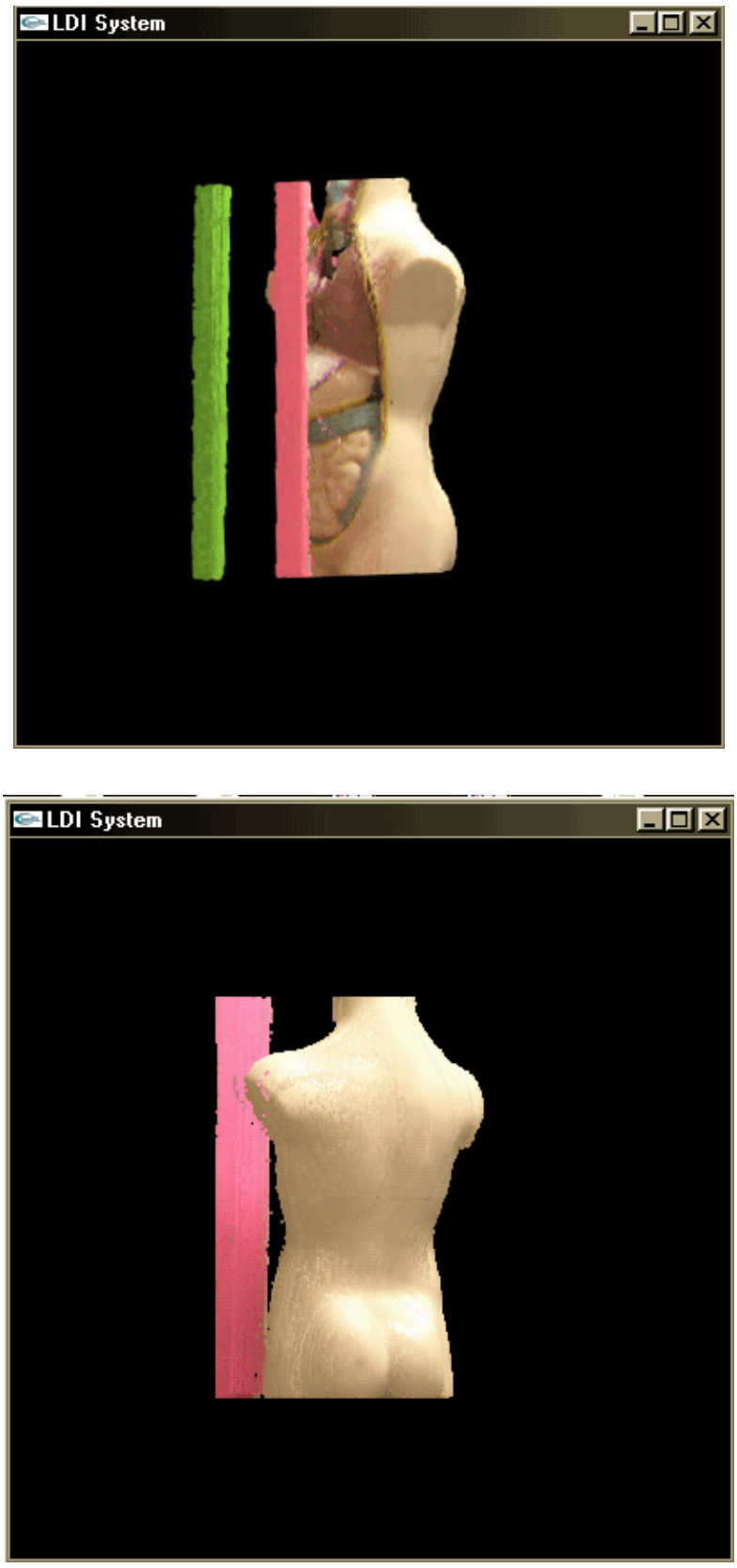


Figure 5.3 (b): Synthetic Views generated by the sparsely sampled LDI system, with splatting

Figure 5.3 (b): Synthetic Views generated by the sparsely sampled LDI system, with splatting

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Reference images of 296x222 resolution, sampled along a circle around the object, at regular intervals of 19.74^{o} Size: 40.6 Mb
Frame Rate:	12.2 fps
Image loading time:	Proportional to the size of an image. Reference Image size: 2.25 Mb Time to load an image: 0.8 seconds

(i) Mannequin Object

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Reference images of 200x200 resolution, sampled along a circle around the object, at regular intervals of 19.74^{o} Size: 24.3 Mb
Frame Rate:	15.3 fps
Image loading time:	Proportional to the size of an image. Reference Image size: 1.35 Mb Time to load an image: 0.5 seconds

(ii) Pooh bear Object

(a)

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Reference images of 296x222 resolution, sampled along a circle around the object, at regular intervals of 19.74^{o} Size: 40.6 Mb
Frame Rate:	7.1 fps
Image loading time:	Proportional to the size of an image. Reference Image size: 2.25 Mb Time to load an image: 0.8 seconds

(i) Mannequin Object

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Reference images of 200x200 resolution, sampled along a circle around the object, at regular intervals of 19.74^{o} Size: 24.3 Mb
Frame Rate:	9.3 fps
Image loading time:	Proportional to the size of an image. Reference Image size: 1.35 Mb Time to load an image: 0.5 seconds

(ii) Pooh bear Object

(b)

Figure 5.4: Statistical information for the sparsely sampled LDI system.

(a) without splatting. (b) with splatting

5.2.2 Uniform Dense Sampling

With a highly dense uniform sampling with reference images taken along a circular orbit around the object, at regular intervals of 2.82 degrees, the results obtained are as depicted in Figure 5.5.

It is observed that the quality of the rendered views of the improved system is comparable to that of the dense sampled system in both the examples, despite the fact that the dense sampled system had a lot more input data at its disposal. The speed of the improved system is much better when compared to the dense sampled system. Figure 5.6 illustrates the statistical information observed with the uniform dense sampled system in case of both the examples – (a) Mannequin object and (b) Pooh bear object.

Figure 5.5 (a): Synthetic Views generated by the densely sampled LDI system.

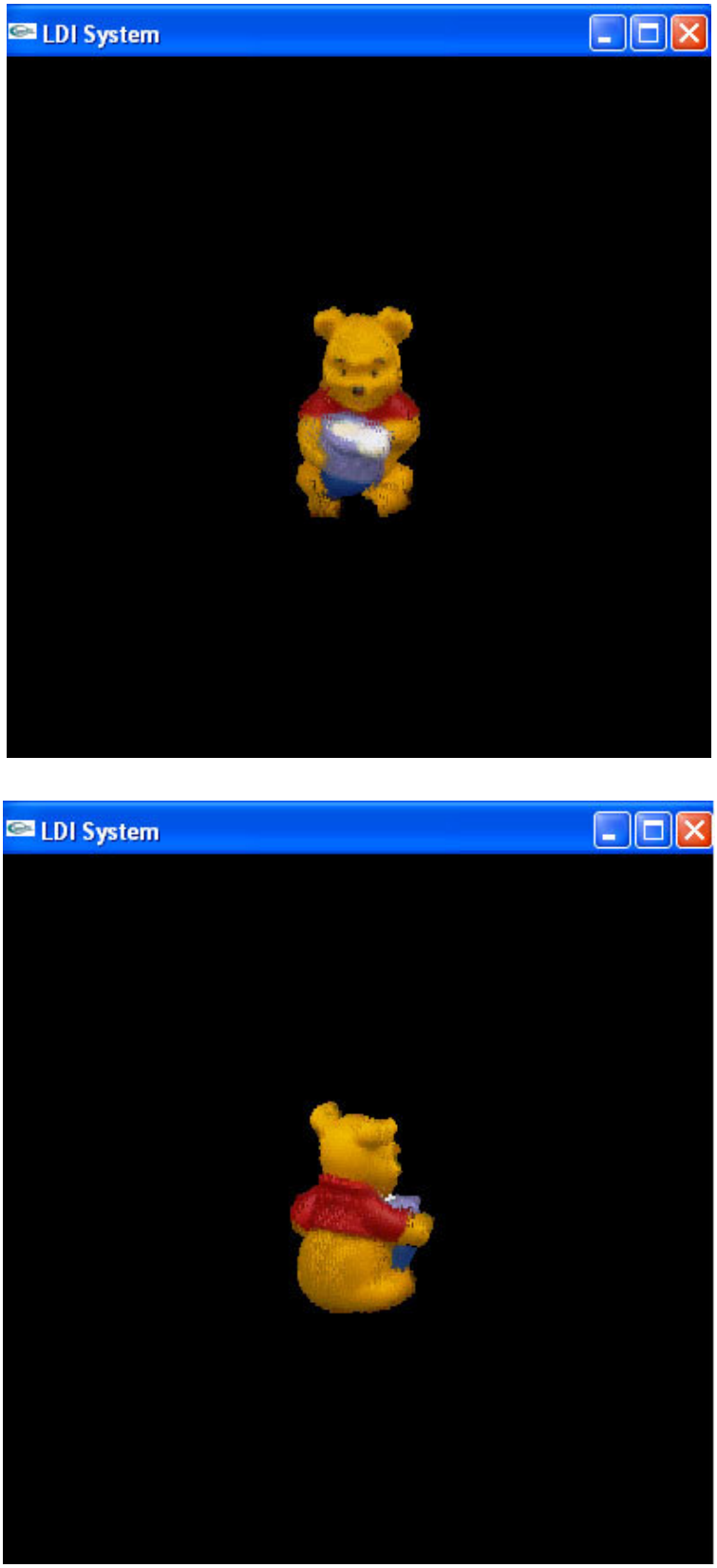


Figure 5.5 (b): Synthetic Views generated by the densely sampled LDI system.

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Reference images of 296x222 resolution, sampled along a circle around the object, at regular intervals of 2.82^o Size: 286 Mb
Frame Rate:	1.3 fps
Image loading time:	Proportional to the size of an image. Reference Image size: 2.25 Mb Time to load an image: 0.8 seconds

(a) Mannequin Object

System Configuration:	CPU: Pentium-4, 1.6 GHz system RAM: 256 MB
Input Data:	Reference images of 200x200 resolution, sampled along a circle around the object, at regular intervals of 2.82^o Size: 170.1 Mb
Frame Rate:	3.1 fps
Image loading time:	Proportional to the size of an image. Reference Image size: 1.35 Mb Time to load an image: 0.5 seconds

(b) Pooh bear Object

Figure 5.6: Statistical information for the densely sampled LDI system.

5.2.3 Summary

Table 5.1, illustrates the statistical difference between our adaptive sampling system and the previous systems – the sparse sampling system, with and without splatting, and the dense sampling system, for both the examples of the mannequin and pooh bear objects.

Attributes ↓	Our Improved System	Sample Implementation of Original LDI System [Shade et al., 1998]		
		Sparsely Sampled (no splatting)	Sparsely Sampled (with splatting)	Densely Sampled
Input Data Size	**97.3 Mb**	**40.6 Mb**	**40.6 Mb**	286 Mb
Rendering Speed / Frame Rate	**Fast** **10.7 fps**	**Fast** **12.2 fps**	Average 7.1 fps	Slow 1.3 fps
Reference View Selection	**Pre-computed Patch Lookup Table**	Closest Reference Images during walkthrough		
Splatting	**No**	**No**	Yes	**No**
Holes	**No**	Yes	Mostly No	**No**
Quality	**Good**	Poor	Average	**Good**

(a) Mannequin Object

Attributes ↓	Our Improved System	Sample Implementation of Original LDI System [Shade et al., 1998]		
		Sparsely Sampled (no splatting)	Sparsely Sampled (with splatting)	Densely Sampled
Input Data Size	**44.1 Mb**	**24.3 Mb**	**24.3 Mb**	170.1 Mb
Rendering Speed / Frame Rate	**Fast** **13.7 fps**	**Fast** **15.3 fps**	Average 9.3 fps	Slow 3.1 fps
Reference View Selection	**Pre-computed Patch Lookup Table**	Closest Reference Images during walkthrough		
Splatting	**No**	**No**	Yes	**No**
Holes	**No**	Yes	Mostly No	**No**
Quality	**Good**	Poor	Average	**Good**

(b) Pooh bear Object

Table 5.1: Comparison of the different systems

It can be noticed that the frame rates of our adaptive system is comparable to that of the sparse system without splatting and is a lot more than the dense system. From the figures

5.1 and 5.5, it's also noticed that the quality of our adaptive system is comparable to the high quality of the dense sampling system.

In case of the mannequin object, a significant difference in quality was noticed between the sparse system and our adaptive system. This owes to the fact that a sparse system couldn't capture all of the visible surfaces of an object as complicated as the mannequin, as most surfaces were visible only from certain view points in the sampling camera trajectory. However, in case of the pooh bear object, which had a lot fewer occluded and non uniform surfaces, the quality of the adaptive system is only slightly better than the sparse system. This is also reflected in the comparison of the sample-data sizes of the sparse system and our adaptive system. In case of the pooh-bear, the data captured to adaptively cover all the visible surfaces was not much more than what was captured with uniform sparse sampling.

CONCLUSION

Sampling issue has been an important and challenging problem in the area of Image Based Rendering. In particular, in rendering frameworks like the Layered Depth Image Based Rendering approach, where a considerable knowledge of the geometry of the scene is known in addition to the acquisition of sample images, there has always been scope to address the sampling issue. The task is complicated because of the reliance of the problem on the structure of the object.

In this thesis, we proposed a method to improve the Layered Depth Image system, by adaptively sampling a scene, to avoid various computations during rendering to make sure that a decent synthetic view is generated within an acceptable time frame. The quality of the output is enhanced, owing to the fact that the synthetic view is not generated by interpolating or splatting neighboring pixels, but with original sampled data. In addition to the improvisation of the quality during rendering, the rendering speed is enhanced by the pre-computed patch lookup table, which simplifies the reference view selection process to a simple lookup of a hash table.

In the evaluation of our proposed method, we demonstrated the advantages of this approach by considering an object with occlusions, and quite a number of non-uniform surfaces. It was established that even an object of such complexity, which could have otherwise been difficult to render without an extremely dense uniform sampling, was rendered much more accurately than a splatted synthetic image. The rendering speed was

comparable to sparse sampling, and better than the splatted system. We hence successfully implemented an adaptive LDI system, with enhanced realism and fast rendering. This is an improvement over the previous LDI, with a sparse sample set and a method of splatting to cater for the visual artifacts.

To summarise, the following goals have been achieved:

- A Layered Depth Image framework has been implemented, that enables rendering of complex 3-D environments, catering for absence of holes and visual artefacts in the modeled world.
- An efficient approach has been devised to tabulate the pre-acquired set of imagery, to ensure fast reference view selection and rendering of the synthetic views.
- A method has been formulated which retains the realism of the 3-D environment, through dense samples of the real world, and yet achieves a rendering engine which is as fast as a sparse sampled LDI system.

FUTURE WORK

Based on the proposed method of improving the Layered Depth Image system, this chapter discusses the future prospects in the area, worth considering.

7.1 Reflectance Properties

In this thesis, the object surface was assumed not to exhibit specular reflection. The reason for this was the fact that we were attempting to sample the whole region space with minimal number of camera viewpoints, meaning, with least redundancy of sampled data. However, exhibition of directional reflectance properties would imply that a surface sampled obliquely would differ from its view from the front. To retain the reflectance properties, either more views would be required for the same surface to minimize computations whilst rendering or the reflectance properties must be deciphered, negated from the samples, and re-applied while rendering.

There is scope for further enhancing the patch recognition and the sampling graph generation methods, by including the aspect of reflectance properties of the surface being scanned. The method to sample, adapting to both the shape and the texture of a surface can be challenging.

7.2 Lighting Effects

With a robust skeleton of the entire scene being rendered, there is considerable scope for including lighting effects like shadows, inter-reflectance and refraction with minimal cost.

Since the geometry information was not just gathered and understood, and instead was put to use to frame a clearly mapped skeleton for the rendering process, the hash tables which define exactly which surfaces are under consideration from any point in the walkthrough, could be put to good use for calculating the lighting effects.

7.3 Three-Dimensional Adaptive Sampling

In this thesis, the sampling problem was addressed, by adaptively sampling the object by considering its 2-dimensional vertical cross-sections. The sampling arcs depicted the points on the sampling circle, from where an edge (representing a patch of the object) on the plane under consideration could be effectively sampled. The visibility regions demarcated the walkthrough area, into horizontal segments, each segment determining the visible surfaces of the object from any camera coordinate in the segment.

This work fails to exploit the possible extension of adaptive sampling in the three-dimensional scope. When calculating the sampling points, to sample the required set of reference patches to effectively capture all visible surfaces of the object, the sampling points were determined in a 2-dimensional context. This doesn't consider the concept of sampling arcs in the vertical scope, resulting in sampling more points than can be seen on the view plane and the possibility of data redundancy amongst patches sampled in different 2-dimensional contexts. In addition, in a scenario where there is vertical occlusion, the visibility regions thus formed would not indicate the fact that some of the patches are occluded by vertical occlusion, resulting in unnecessary rendering of data which cant be seen from a given camera point. The scan line information in the patch lookup tables can only filter out patches which don't fit into the vertical field of view of

the camera and can't provide information of lack of visibility owing to vertical occlusions.

There is considerable scope for extending the work of adaptive sampling to exploit the possible three-dimensional extension. This would further reduce the size of the reference sample data and significantly bring down the number of points rendered in case of objects with a lot of vertical occlusion. The adaptive sampling approach can be extended to three dimensions by approaching the problem of determination of a sampling arc, by attempting to find the *sampling spherical-cap*. Also, the concept of visibility regions and patch lookup table could be extrapolated to cater for the vertical angle of view of the walkthrough camera, where the visibility region would be a region of space determined by a surface similar to a tessellated sphere rather than a two-dimensional contour.

7.4 Experiments on more complex scenes

In this research, the adaptive sampling approach was put to test on two objects, a mannequin with two rods in front of it, and a pooh-bear. Experiments can be conducted on more complex scenes comprising of objects with a lot more concave surfaces or a combination of complex objects. Also, the current experiments were conducted by sampling at only one sampling rate. As already stated, the method supports samples of different sampling rates, being capable of rendering at the resolution of the walkthrough camera. Experiments to zoom into a higher sampling rate, with camera coordinates stationed at a point of lower sampling rate could be other ways to analyze the behavior of the adaptive system.

The solution could be extended to consider a three dimensional environment rather than a

group of objects, where the camera is not outside the cylinder encapsulating the objects, but is inside of it. The modification to the sampling-arc derivation and the construction of the patch lookup table to cater for the same, might be an interesting area to explore.

REFERENCES

[Agrawala et al., 2000] Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich and Laurent Moll. Efficient image-based methods for rendering soft shadows. Proceedings of the conference on Computer graphics, 2000, pp. 375-384.

[Aliaga et al., 1999] Daniel Aliaga, Jon Cohen, Andrew Wilson, Eric Baker, Hansong Zhang, Carl Erikson, Kenny Hoff, Tom Hudson, Wolfgang Stuerzlinger, Rui Bastos, Mary Whitton, Fred Brooks and Dinech Manocha. MMR: An interactive massive model rendering system using geometric and image based acceleration. Proceedings of the 1999 symposium on Interactive 3D graphics, 1999, pp. 199 – 206.

[Angel, 2000] Edward Angel. Interactive Computer Graphics: A top down approach with OpenGL. 2nd Edition, Addison Wesley Longman, Inc., United States of America, 2000.

[Ashikhmin et al., 2000] Michael Ashikhmin, Simon Premoze and Peter Shirley. A Microfacet-based BRDF Generator. Proceeding of the 27th Annual Conference on Computer Graphics, 2000, Jul 2000, pp. 65 - 74.

[Bastos et al., 1999] Rui Bastos, Kenneth Hoff, William Wynn and Anselmo Lastra. Increased photorealism for interactive architectural walkthroughs. Proceeding of the 1999 symposium on Interactive 3D graphics, 1999, pp. 183 - 190.

[Boivin, Gagalowicz, 2001] Samuel Boivin and Andre Gagalowicz. Image-Based Rendering of Diffuse, Specular and Glossy Surfaces from a single image. Proceeding of the 28th Annual Conference on Computer Graphics, 2001, Aug

2001, pp. 107 - 116.

[Cabral et al., 1999] Brian Cabral, Marc Olano and Philip Nemec. Reflection space image based rendering. SIGGRAPH, Proceedings of the SIGGRAPH 1999 annual conference on Computer graphics, 1999, pp. 165 - 170.

[Chai et al., 2000] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan and Heung-Yeung Shum. Plenoptic Sampling. Proceedings of the SIGGRAPH 2000 annual conference on Computer graphics, 2000, pp. 307 – 318.

[Chang et al., 1999] Chun-Fa Chang, Gary Bishop and Anselmo Lastra. LDI Tree: A hierarchical representation for image-based rendering. Proceedings of the SIGGRAPH 1999 annual conference on Computer graphics, 1999, pp. 291 – 298.

[Chen, 1995] Shenchang Eric Chen. QuickTime VR: An image based approach to virtual environment navigation. Proceedings of the 22nd annual ACM conference on Computer graphics, 1995, pp. 29 - 38.

[Chen and Willams, 1993] Shenchang Eric Chen and Lance Williams. View Interpolation for image synthesis. Proceedings of the 20th annual conference on Computer graphics, 1993, pp. 279 – 288.

[Cohen et al., 1997] Michael Cohen, Marc Levoy, Jitendra Malik, Leonard McMillan and Eric Chen. Image-based rendering: really new or déjà vu?. Proceedings of the 24th annual conference on Computer graphics & interactive techniques, 1997, pp. 468 – 470.

[De Bonet, 1997] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. Proceedings of the 24th annual conference on Computer graphics & interactive techniques, 1997, pp. 361 – 368.

[Debevec et al., 2000] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker,

Westley Sarokin and Mark Sagar. Acquiring the Reflectance Field of a Human Face. Proceedings of SIGGRAPH 2000, Jul 2000, pp. 145 – 156.

[Fleshman et al., 1999] Shachar Fleishman, Daniel Cohen-Or and Dani Lischinski. Automatic Camera Placement for Image-Based Modeling. Proceedings of the Pacific Graphics '99, 1999.

[Garcia, 1998] Miguel Angel Garcia. A Two-Stage Algorithm for Planning the Next View from Range Images. Proceedings of the British Machine Vision Conference' 97, 1998, pp. 720 – 729.

[Gortler, Li-wei, Cohen, 1997] Steven J. Gortler, Li-wei He and Michael F. Cohen. Rendering Layered Depth Images. Harvard Computer Science Technical Report tr9709, Harvard University, 1997.

[Gortler et al., 1996] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, Michael F. Cohen. The Lumigraph. SIGGRAPH, Proceedings of the 23rd annual conference on Computer graphics, 1996, pp. 43 - 54.

[Grossman, Dally, 1998] J.P. Grossman, William J. Dally. Point Sample Rendering. Proceedings of the 9th Eurographics workshop on rendering, Jun 1998, pp. 181 - 192.

[Heckbert, 1989] P.Heckbert. Fundamentals of texture mapping and image warping. Technical report, CS Division, U. C Berkeley, Jun 1989.

[Huang et al., 1998] Ho-Chao Huang, Shung-Hua Nain, Yi-Ping Hung and Tse Cheng. Disparity-based view morphing: A new technique for image based rendering. Proceedings of the ACM Symposium on Virtual reality software and technology 1998, 1998, pp. 9 – 16

[Jensen et al., 2001] Henrik Wann Jensen, Fredo Durand, Michael M. Stark and Simon

Premoze. A Physically-Based Night Sky Model. Proceedings of the 28th Annual Conferenceon Computer Graphics 2001, Aug 2001, pp. 399 - 408

[Joy, 1997] Kenneth I. Joy. The depth buffer visible surface algorithm. Online Computer Graphics Notes, Computer Science Department, University of California, Davis, 1997.

[Kang, 1997] Sing Bing Kang. A survey of image-based rendering techniques. Cambridge Technical Report Series, August 1997.

[Katayama et al., 1999] Akihiro Katayama, Yukio Sakagawa, Hiroyuki Yamamoto and Hideyuki Tamura. Shading and shadow casting in image-based rendering without geometric models. Proceedings of the conference on SIGGRAPH 99: conference abstracts and applications, 1999, pp. 275.

[Lee, 1998] Lee Liang Chye. An Image-Based approach to render complex 3-D environments. MSc. Thesis, National University of Singapore, 1998.

[Levoy and Hanrahan, 1996] Marc Levoy and Pat Hanrahan. Light field rendering. Proceedings of the 23rd annual conference on Computer graphics, 1996, pp. 31 - 42.

[Leung and Chen, 2001] Wing Ho Leung and Tsuhan Chen. Line-Space Representation and Compression for Image-Based Rendering. Carnegie Melon Technical Report AMP01-02, 2001.

[Lluch et al., 2004] Javier Lluch, Emilio Camahort and Roberto Vivo. An Image-Based Multiresolution Model for interactive foliage rendering. WSCG'04, the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2004, pp. 507.

[Massios, Fisher, 1998] Nikolaos A. Massios and Robert B. Fisher. A Best Next View

Selection Algorithm incorporating a Quality Criterion. Proceedings of the British Machine Vision Conference (BMVC) ' 97, Sep, 1998, pp. 780 - 789.

[McMillan, 1995] Leonard McMillan. A list-priority rendering algorithm for redisplaying projected surfaces. UNC-Chapel Hill Computer Science Technical Report #95-005, University of North Carolina, 1995.

[McMillan and Bishop, 1995] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An image based rendering system. Proceedings of the 22nd annual ACM conference on Computer graphics, 1995, pp. 39 – 46.

[Namboori, Teh and Huang, 2004] Ravinder Namboori, Hung Chuan Teh, Zhiyong Huang. An Adaptive Sampling Method for Layered Depth Image: Computer Graphics International, Jun, 2004, pp. 206-213

[Preetham et al., 1999] A.J. Preetham, Peter Shirley and Brian Smits. A Practical Analytic Model for Daylight. Proceedings of the 26th Annual conference on Computer graphics, 1999, pp. 91 – 100.

[Oh et al., 2001] Byong Mok Oh, Max Chen, Julie Dorsey and Fredo Durand. Image-Based Modeling and Photo Editing. Proceedings of the 28th annual ACM conference on Computer graphics, Aug 2001, pp. 433 – 442.

[Pfister et al., Gross, 2000] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar and Markus Gross. Surfels: Surface Elements as Rendering Primitives. Proceedings of the ACM SIGGRAPH '00, Jul, 2000, pp. 335 – 342.

[Pito, 1995] Richard Pito. A solution to the next Best View Problem for Automated CAD Model Acquisition of Free-form Objects Using Range Cameras, Technical Report 95-23, GRASP Lab, University of Pennsylvania, May 1995.

[Popescu et al., 1998] Voicu Popescu, Anselmo Lastra, Daniel Aliaga and Manuel de

Oliveira Neto. Efficient warping for architectural walkthroughs using layered depth images. Proceedings of the conference on Visualization '98, 1998, pp. 211 – 215.

[Sato et al., 1997] Yoichi Sato, Mark D. Wheeler and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. Proceedings of the 24th annual conference on Computer graphics & interactive techniques, 1997, pp. 379 - 387.

[Seitz and Dyer, 1996] Steven M. Seitz and Charles R. Dyer. View Morphing. Proceedings of the 23rd annual conference on Computer graphics, 1996, pp. 21 - 30.

[Shade et al., 1998] Jonathan Shade, Steven Gortler, Li-wei He and Richard Szeliski. Layered depth images. Proceedings of the 25th annual conference on Computer Graphics, 1998, pp. 231 - 242.

[Shum, He, 1999] Heung-Yeung Shum and Li-wei He. Rendering with Concentric Mosaics. Proceedings of the 26th annual conference on Computer Graphics, Jul 1999, pp. 299 - 306.

[Sillion et al., 1991] Francois X. Sillion, James R. Arvo, Stephen H. Westin and Donald P. Greenberg. A global illumination solution for general reflectance distributions. Proceedings of the 18th international conference on Computer graphics, 1991, pp. 187 - 196.

[Stuerzlinger, 1998] Wolfgang Stuerzlinger. Imaging all Visible Surfaces. Computer Science Technical Report TR98-010, Mar 1998

[Wand et al.2001] Michael Wand, Matthias Fischer, Ingmar Peter, Friedhelm Meyer auf

der Heide and Wolfgang Straber. The Randomised z-Buffer Algorithm: Interactuve Rendering of Highly Complex Scenes. Proceedings of the conference on Computer graphics, SIGGRAPH '01, 2001, pp. 361 - 370.

[Westin et al., 1992] Stephen H. Westin, James R. Arvo and Kenneth E. Torrance. Predicting reflectance functions from complex surfaces. Proceedings of the 19th annual conference on Computer graphics, 1992, pp. 255 - 264.

[Xiao et al., 1991] Xiao D. He, Kenneth E. Torrance, Francois X. Sillion and Donald P. Greenberg. A comprehesive physical model for light reflection. Proceedings of the 18th international conference on Computer graphics, 1991, pp. 175 - 186.

[Xiao et al., 1992] Xiao D. He, Patrick O. Heynen, Richard L. Phillips, Kenneth E. Torrance, David H. Salesin and Donald P. Greenberg. A fast and accurate light reflection model. Proceedings of the 19th annual conference on Computer graphics, 1992, pp. 253 - 254.

[Yu and Malik, 1998] Yizhou Yu and Jitendra Malik. Recovering photometric properties of architectural scenes from photographs. Proceedings of the 25th annual conference on Computer Graphics, 1998, pp. 207 - 217.

[Yu et al., 1999] Yizhou Yu, Paul Debevec, Jitendra Malik and Tim Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. Proceedings of the SIGGRAPH 1999 annual conference on Computer graphics, 1999, pp. 215 – 224

[Zhang, Chen, 2001] Cha Zhang and Tsuhan Chen. Generalised Plenoptic Sampling. Carnegie Melon Technical Report AMP01-06, Sep 2001.

[Zwicker et al., 2001] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar and Markus

Gross. Surface Splatting. Proceedings of the 28th Annual Conference on computer graphics, SIGGRAPH 2001, Aug 2001, pp. 371 - 378

APPENDIX A

Sampling Arc Derivation

The following steps describe the mathematical derivation of the Sampling Arc formulae mentioned in section 3.3.3 of Chapter 3.

A.1Initial Conditions

1. Co-ordinate System: (s,t)
2. s–axis is parallel to the edge
3. t–axis is perpendicular to the edge
4. Centre of the Sampling Circle lies at the origin

A.2 Given Input

1. Radius of the Sampling Circle (R)
2. Focal length of Camera (f)
3. Left - end point of line segment (s_1,t_1)
4. Right - end point of line segment (s_2,t_2)

A.3 Derived Input

$$d_0 = \sqrt{(s_2 - s_1)^2 + (t_2 - t_1)^2}$$

$$\Delta d = d_0 / 10$$

$$Z_0 = ((t_1 + t_2) / 2) - \sqrt{R^2 - s_0{}^2}$$

$$s_0 = (s_1 + s_2) / 2$$

$$t_0 = ((t_1 + t_2) / 2) - Z_0$$

$$d_s = (d_0 / 2) - \Delta d$$

$$\Delta d' = (f / Z_0) \times \Delta d$$

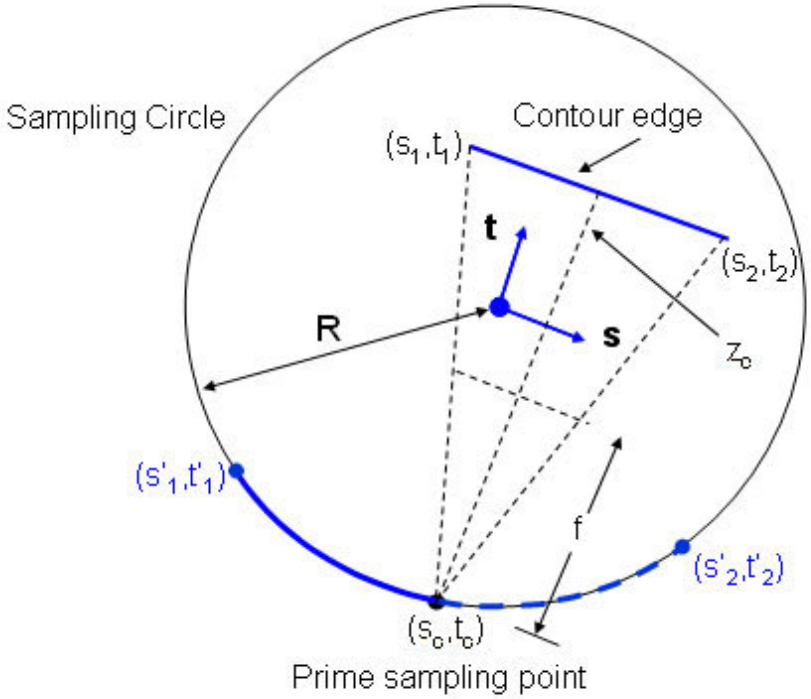


Figure A.1: Sampling Arc

A.4 Derivation of the left end-point of the Sampling Arc

Let, the left end-point of the Sampling Arc be (s', t')
So, we have,

$$s'^2 + t'^2 = R^2$$
$$\Delta s = (s' - s_0)$$
$$\Delta t = (t' - t_0)$$
$$Z' = \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2} \quad \leftarrow \quad (1)$$
$$\cos\phi = \Delta s / Z'$$
$$\tan\phi = (Z_0 - \Delta t) / \Delta s$$

From Figure A.2, we have the relation,

$$\Rightarrow (\Delta d' + d_s') / f = d_0 / 2Z_0$$
$$\therefore \quad d_s' = ((d_0 f / 2Z_0) - \Delta d') \quad \leftarrow \quad (2)$$

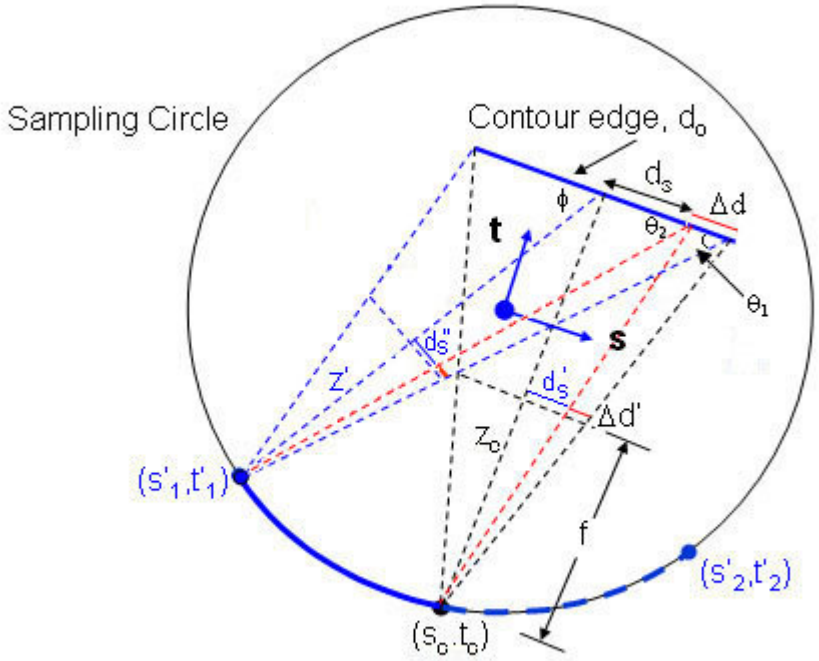


Figure A.2: Sampling Arc derivation

Also, we have,

$$\Rightarrow d_s' / f = d_s / Z_0$$

$$\therefore \; d_s = (Z_0 / f) d_s'$$

$$\therefore \; d_s = ((d_0 / 2) - (Z_0 / f)\Delta d') \quad \leftarrow (3)$$

According to the figure,

$$\theta = \tan^{-1}((d_s'' + \Delta d' / 2) / f) \quad \leftarrow (4)$$

$$\theta_1 = (\phi - \theta) \quad \leftarrow (5)$$

$$\theta_2 = \phi - \tan^{-1}(d_s'' / f) \quad \leftarrow (6)$$

By Sin Rule, we have,

$$Z' / \sin\theta_2 = d_s / \sin(\tan^{-1}(d_s'' / f)) \quad \leftarrow (7)$$

$$Z' / \sin\theta_1 = d_0 / 2\sin(\tan^{-1}((d_s'' + \Delta d' / 2) / f)) \quad \leftarrow (8)$$

From Equations (7) and (8), we have,

$$\Rightarrow \sin\theta_1 / \sin\theta_2 = (2d_s / d_0) \times \sin(\tan^{-1}((d_s'' + \Delta d'/2)/f)) / \sin(\tan^{-1}(d_s''/f))$$
$$= \sin(\phi - \tan^{-1}((d_s'' + \Delta d'/2)/f)) / \sin(\phi - \tan^{-1}(d_s''/f))$$

$$\therefore 2d_s / d_0 = (\cot(\tan^{-1}((d_s'' + \Delta d'/2)/f)) - \cot\phi) / (\cot(\tan^{-1}(d_s''/f)) - \cot\phi)$$
$$= ((f/(d_s'' + \Delta d'/2)) - \cot\phi) / ((f/d_s'') - \cot\phi)$$

$$\therefore (2d_s - d_0)/d_0 = -f\Delta d'/2((f - d_s''\cot\phi) \times (d_s'' + \Delta d'/2))$$

After putting the value of d_s from Eqn (3) , we have :

$$(f - d_s''\cot\phi)(d_s'' + \Delta d'/2) = d_0 f^2 / 4Z_0$$
$$\therefore d_s''^2 \cot\phi + d_s''((\Delta d'\cot\phi/2) - f) + ((d_0 f^2/4Z_0) - f\Delta d'/2) = 0$$

$$Put\ M = ((d_0 f^2/4Z_0) - f\Delta d'/2)$$
$$= (f/2)((d_0 f/2Z_0) - \Delta d')$$
$$= d_s f^2 / 2Z_0$$

$$\therefore d_s''^2 \cot\phi + d_s''((\Delta d'\cot\phi/2) - f) + M = 0$$

$$\therefore\ d_s'' = ((f - (\Delta d'\cot\phi/2)) \pm \sqrt{(f - (\Delta d'\cot\phi/2))^2 - 4M\cot\phi}\ ) / 2\cot\phi \qquad \leftarrow (9)$$

Now , we equate the values of Z' from Eqns (1) and (7) :

$$\therefore\ \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2} = d_s \sin\theta_2 / \sin(\tan^{-1}(d_s''/f))$$
$$\therefore\ \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2} = d_s \sin(\phi - \tan^{-1}(d_s''/f)) / \sin(\tan^{-1}(d_s''/f))$$
$$\therefore\ d_s((\sin\phi\cot(\tan^{-1}(d_s''/f))) - \cos\phi) = \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2}$$
$$\therefore\ d_s \sin\phi((f/d_s'') - \cot\phi) = \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2}$$
$$\therefore\ d_s(Z_0 - \Delta t)/(\sqrt{\Delta s^2 + (Z_0 - \Delta t)^2})((f/d_s'') - \cot\phi) = \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2}$$

$\therefore \quad (f / d_s'') - \cot\phi = (\Delta s^2 + (Z_0 - \Delta t)^2) / (d_s (Z_0 - \Delta t))$

$\therefore \quad f / d_s'' = ((\Delta s^2 + (Z_0 - \Delta t)^2) / (d_s (Z_0 - \Delta t))) + (\Delta s / (Z_0 - \Delta t))$

$\therefore \quad f / d_s'' = (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s) / (d_s (Z_0 - \Delta t))$

$\therefore \quad d_s'' = f d_s (Z_0 - \Delta t) / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s) \qquad \leftarrow (10)$

Now , equate the values of d_s'' from Eqns (9) and (10) :

$\therefore \quad ((f - (\Delta d' \cot\phi / 2)) \pm \sqrt{(f - (\Delta d' \cot\phi / 2))^2 - 4M \cot\phi}\ ) / 2\cot\phi =$
$f d_s (Z_0 - \Delta t) / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)$

$\therefore \quad (f - (\Delta d' \cot\phi / 2)) \pm \sqrt{(f - (\Delta d' \cot\phi / 2))^2 - 4M \cot\phi} =$
$2 f d_s \Delta s / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)$

Now, put $T = (f - (\Delta d' \cot\phi / 2))$

$\therefore \quad \pm \sqrt{T^2 - 4M \cot\phi} = (2 f d_s \Delta s / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)) - T$

$\therefore \quad T^2 - 4M \cot\phi = ((2 f d_s \Delta s / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)) - T)^2$

$\therefore \quad (4 f^2 d_s^2 \Delta s^2 / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)^2)$
$- (4 T f d_s \Delta s / (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)) + 4M \cot\phi = 0 \qquad \leftarrow (11)$

Put $\quad Q = (\Delta s^2 + (Z_0 - \Delta t)^2 + d_s \Delta s)$

After putting the values of $\cot\phi$ and M in Eqn (11), we get :

$\therefore \quad (f d_s \Delta s / Q^2) + (f / (2 Z_0 (Z_0 - \Delta t))) - (T / Q) = 0$

Now, put the value of T.

$\therefore \quad (f d_s \Delta s / Q^2) + (f / (2 Z_0 (Z_0 - \Delta t))) - (f / Q) + (\Delta d' \Delta s / (2Q(Z_0 - \Delta t))) = 0$

On solving it, we get :

$$\therefore\ (-d_s \Delta s / Q) = ((Qf + Z_0 \Delta s \Delta d') / (2 f Z_0 (Z_0 - \Delta t))) - 1$$

$$\therefore\ 2 f d_s \Delta s Z_0 (Z_0 - \Delta t) = Q(f Z_0^{\,2} - Z_0 \Delta s \Delta d' - f(\Delta s^2 + \Delta t^2 + d_s \Delta s))$$

Put $\Delta t = (t' - t_0)$

$$\therefore\ 2 f d_s Z_0 \Delta s((Z_0 + t_0) - t') = ((d_s - 2 s_0)\Delta s + t_0^{\,2} + (Z_0 + t_0)^2 - 2(Z_0 + t_0)t') \times ((f Z_0^{\,2} - 2 f t_0^{\,2}) + \Delta s(2 s_0 f - f d_s - Z_0 \Delta d') + 2 f t_0 t')$$

Put $T = \Delta s$

$$\therefore\ T^2((d_s - 2 s_0)(2 s_0 f - f d_s - Z_0 \Delta d')) + T(((d_s - 2 s_0)(f Z_0^{\,2} - 2 f t_0^{\,2} + 2 f t_0 t')) + ((2 s_0 f - f d_s - Z_0 \Delta d')(t_0^{\,2} + (Z_0 + t_0)^2 - 2(Z_0 + t_0)t'))) + ((t_0^{\,2} + (Z_0 + t_0)^2 - 2(Z_0 + t_0)t')(f Z_0^{\,2} - 2 f t_0^{\,2} + 2 f t_0 t')) = 0$$

$$\therefore\ T^2[(d_s - 2 s_0)(2 s_0 f - f d_s - Z_0 \Delta d')] + T[(4 f d_s t_0 - 8 f s_0 t_0 + 4 f d_s Z_0 - 4 f s_0 Z_0 + 2 Z_0 \Delta d'(Z_0 + t_0))t' - (4 f d_s t_0^{\,2} + 2 f d_s Z_0^{\,2} + 4 f d_s Z_0 t_0 - 4 f s_0 Z_0 t_0 - 8 s_0 f t_0^{\,2} + Z_0 \Delta d'(t_0^{\,2} + (Z_0 + t_0)^2))] + [(-4 f t_0 (Z_0 + t_0))t'^2 + (8 f t_0^{\,3} + 8 f t_0^{\,2} Z_0 - 2 f Z_0^{\,3})t' + ((t_0^{\,2} + (Z_0 + t_0)^2)(f Z_0^{\,2} - 2 f t_0^{\,2}))] = 0$$

$$\therefore\ T^2 K + T(Mt' - N) + P = 0$$

Where,

$$K = (d_s - 2 s_0)(2 s_0 f - f d_s - Z_0 \Delta d')$$

$$M = (4 f d_s t_0 - 8 f s_0 t_0 + 4 f d_s Z_0 - 4 f s_0 Z_0 + 2 Z_0 \Delta d'(Z_0 + t_0))$$

$$N = (4 f d_s t_0^{\,2} + 2 f d_s Z_0^{\,2} + 4 f d_s Z_0 t_0 - 4 f s_0 Z_0 t_0 - 8 s_0 f t_0^{\,2} + Z_0 \Delta d'(t_0^{\,2} + (Z_0 + t_0)^2))$$

$$P = P_1 t'^2 + P_2 t' + P_3$$
$$P_1 = (-4 f t_0 (Z_0 + t_0))$$
$$P_2 = (8 f t_0{}^3 + 8 f t_0{}^2 Z_0 - 2 f Z_0{}^3)$$
$$P_3 = (t_0{}^2 + (Z_0 + t_0)^2)(f Z_0{}^2 - 2 f t_0{}^2)$$

$$\therefore\ T = ((N - Mt') \pm \sqrt{(Mt' - N)^2 - 4KP}\ ) / 2K$$
$$= (s' - s_0)$$
$$= (\sqrt{R^2 - t'^2} - s_0)$$

Put $X = (N - Mt')$

$$\therefore\ X \pm \sqrt{X^2 - 4KP} = 2K(\sqrt{R^2 - t'^2} - s_0)$$
$$\therefore\ X^2 - 4KP = (2K(\sqrt{R^2 - t'^2} - s_0) - X)^2$$
$$\therefore\ K(\sqrt{R^2 - t'^2} - s_0)^2 - X(\sqrt{R^2 - t'^2} - s_0) + P = 0$$

$$\therefore\ (2Ks_0 + X)(s_0 - \sqrt{R^2 - t'^2}) + (P - Kt'^2 + Kt_0{}^2) = 0$$
$$\therefore\ s_0 + (P - Kt'^2 + Kt_0{}^2) / (2Ks_0 + X) = \sqrt{R^2 - t'^2}$$
$$\therefore\ (s_0 + (P - Kt'^2 + Kt_0{}^2) / (2Ks_0 + X))^2 = (R^2 - t'^2)$$
$$\therefore\ ((P - Kt'^2 + Kt_0{}^2)^2 / (2Ks_0 + X)^2) + 2s_0(P - Kt'^2 + Kt_0{}^2) / (2Ks_0 + X)$$
$$= ((R^2 - s_0{}^2) - t'^2)$$
$$= (t_0{}^2 - t'^2)$$

$$\therefore (P - Kt'^2 + Kt_0{}^2)^2 + 2s_0(P - Kt'^2 + Kt_0{}^2)(2Ks_0 + X) + (t'^2 - t_0{}^2)(2Ks_0 + X)^2 = 0$$
$$\therefore\ X^2(t'^2 - t_0{}^2) + 2Ks_0 X(t'^2 - t_0{}^2) + P(4Ks_0{}^2 - 2Kt'^2 + 2Kt_0{}^2 + 2s_0(N - Mt'))$$
$$+ K^2 t'^4 + K^2 t_0{}^4 - 2K^2 t_0{}^2 t'^2 + P^2 = 0$$

Put the value of X as $(N - Mt')$.

$$\therefore\ (N - Mt')^2(t'^2 - t_0{}^2) + 2Ks_0(N - Mt')(t'^2 - t_0{}^2) + P(4Ks_0{}^2 - 2Kt'^2 + 2Kt_0{}^2 + 2s_0(N - Mt')) + K^2t'^4 + K^2t_0{}^4 - 2K^2t_0{}^2t'^2 + P^2 = 0$$

Put the value of P as $(P_1t'^2 + P_2t' + P_3)$.

$$\therefore\ (N^2 + M^2t'^2 - 2MNt')(t'^2 - t_0{}^2) + 2Ks_0(N - Mt')(t'^2 - t_0{}^2) + (P_1t'^2 + P_2t' + P_3)(4Ks_0{}^2 - 2Kt'^2 + 2Kt_0{}^2 + 2s_0(N - Mt')) + K^2t'^4 + K^2t_0{}^4 - 2K^2t_0{}^2t'^2 + (P_1t'^2 + P_2t' + P_3)^2 = 0$$

$$\begin{aligned}\therefore\quad & t'^4[(M^2 + K^2 + P_1^2 - 2KP_1)] \\ & + t'^3[(2P_1P_2 - 2MN - 2MKs_0 - 2KP_2 - 2MP_1s_0)] \\ & + t'^2[(2KP_1t_0{}^2 - 2KP_3 + 4KP_1s_0{}^2 + 2P_1P_3 + P_2{}^2 + N^2 \\ & \qquad - 2K^2t_0{}^2 + 2KNs_0 - M^2t_0{}^2 + 2s_0NP_1 - 2s_0MP_2)] \\ & + t'[(2MNt_0{}^2 + 2KMs_0t_0{}^2 + 2P_2P_3 + 4KP_2s_0{}^2 + 2KP_2t_0{}^2 + 2s_0NP_2 - 2s_0MP_3)] \\ & + [(2s_0NP_3 - N^2t_0{}^2 - 2KNs_0t_0{}^2 + K^2t_0{}^4 + P_3{}^2 + 4KP_3s_0{}^2 + 2KP_3t_0{}^2)] \quad = \quad 0\end{aligned}$$

Hence the left end-point of the Sampling Arc can be written as,

$$\begin{aligned}\Rightarrow\quad & t'^4[(M^2 + K^2 + P_1^2 - 2KP_1)] \\ & + t'^3[(2P_1P_2 - 2MN - 2MKs_0 - 2KP_2 - 2MP_1s_0)] \\ & + t'^2[(2KP_1t_0{}^2 - 2KP_3 + 4KP_1s_0{}^2 + 2P_1P_3 + P_2{}^2 + N^2 \\ & \qquad - 2K^2t_0{}^2 + 2KNs_0 - M^2t_0{}^2 + 2s_0NP_1 - 2s_0MP_2)] \\ & + t'[(2MNt_0{}^2 + 2KMs_0t_0{}^2 + 2P_2P_3 + 4KP_2s_0{}^2 + 2KP_2t_0{}^2 + 2s_0NP_2 - 2s_0MP_3)] \\ & + [(2s_0NP_3 - N^2t_0{}^2 - 2KNs_0t_0{}^2 + K^2t_0{}^4 + P_3{}^2 + 4KP_3s_0{}^2 + 2KP_3t_0{}^2)] \quad = \quad 0\end{aligned}$$

Where,

$$K = (d_s - 2s_0)(2s_0 f - fd_s - Z_0 \Delta d')$$
$$M = (4fd_s t_0 - 8fs_0 t_0 + 4fd_s Z_0 - 4fs_0 Z_0 + 2Z_0 \Delta d'(Z_0 + t_0))$$
$$N = (4fd_s {t_0}^2 + 2fd_s {Z_0}^2 + 4fd_s Z_0 t_0 - 4fs_0 Z_0 t_0 - 8s_0 f{t_0}^2 + Z_0 \Delta d'({t_0}^2 + (Z_0 + t_0)^2))$$
$$P_1 = (-4ft_0(Z_0 + t_0))$$
$$P_2 = (8f{t_0}^3 + 8f{t_0}^2 Z_0 - 2f{Z_0}^3)$$
$$P_3 = ({t_0}^2 + (Z_0 + t_0)^2)(f{Z_0}^2 - 2f{t_0}^2)$$

A.5 Derivation of the right end-point of the Sampling Arc

Let, the Right end-point of the Sampling Arc be, (s', t')
So, we have,

$$s'^2 + t'^2 = R^2$$
$$\Delta s = (s' - s_0)$$
$$\Delta t = (t' - t_0)$$
$$Z' = \sqrt{\Delta s^2 + (Z_0 - \Delta t)^2} \quad \leftarrow \quad (1)$$
$$\cos\phi = \Delta s / Z'$$
$$\tan\phi = (Z_0 - \Delta t) / \Delta s$$

Also, we have the relation,

$$\Rightarrow (\Delta d' + d_s') / f = d_0 / 2Z_0$$
$$\therefore \quad d_s' = ((d_0 f / 2Z_0) - \Delta d') \quad \leftarrow \quad (2)$$

Also, we have,

$$\Rightarrow d_s' / f = d_s / Z_0$$
$$\therefore \quad d_s = (Z_0 / f) d_s'$$
$$\therefore \quad d_s = ((d_0 / 2) - (Z_0 / f)\Delta d') \quad \leftarrow (3)$$

Now, for the Right end-point, we have,

$$\theta = \tan^{-1}((d_s'' + \Delta d'/2)/f) \qquad \leftarrow (4)$$

$$\theta_1 = (\pi - (\phi + \theta)) \qquad \leftarrow (5)$$

$$\theta_2 = (\pi - (\phi + \tan^{-1}(d_s''/f))) \qquad \leftarrow (6)$$

By Sin Rule, we have,

$$Z'/\sin\theta_2 = d_s/\sin(\tan^{-1}(d_s''/f)) \quad \leftarrow (7)$$

$$Z'/\sin\theta_1 = d_0/2\sin(\tan^{-1}((d_s'' + \Delta d'/2)/f)) \quad \leftarrow (8)$$

From Equations (7) and (8), we have,

$$\Rightarrow \sin\theta_1/\sin\theta_2 = (2d_s/d_0)\times\sin(\tan^{-1}((d_s'' + \Delta d'/2)/f))/\sin(\tan^{-1}(d_s''/f))$$
$$= \sin(\phi + \tan^{-1}((d_s'' + \Delta d'/2)/f))/\sin(\phi + \tan^{-1}(d_s''/f))$$

$$\therefore 2d_s/d_0 = (\cot(\tan^{-1}((d_s'' + \Delta d'/2)/f)) + \cot\phi)/(\cot(\tan^{-1}(d_s''/f)) + \cot\phi)$$
$$= ((f/(d_s'' + \Delta d'/2)) + \cot\phi)/((f/d_s'') + \cot\phi)$$
$$\therefore (2d_s - d_0)/d_0 = -f\Delta d'/2((f + d_s''\cot\phi)(d_s'' + \Delta d'/2))$$

After putting the value of d_s from Eqn (3) , we have :

$$(f + d_s''\cot\phi)(d_s'' + \Delta d'/2) = d_0 f^2/4Z_0$$
$$\therefore d_s''^2\cot\phi + d_s''(f + (\Delta d'\cot\phi/2)) + (f\Delta d'/2 - (d_0 f^2/4Z_0)) = 0$$

$$Put\ M = (f\Delta d'/2 - (d_0 f^2/4Z_0))$$
$$= (f/2)(\Delta d' - (d_0 f/2Z_0))$$
$$= (-d_s f^2/2Z_0)$$

$$\therefore d_s''^2 \cot\phi + d_s''(f + (\Delta d' \cot\phi/2)) + M = 0$$

$$\therefore\ d_s'' = (-(f + (\Delta d' \cot\phi/2)) \pm \sqrt{(f + (\Delta d' \cot\phi/2))^2 - 4M \cot\phi}\ )/2\cot\phi \qquad \leftarrow (9)$$

After solving for the right end-point of the Sampling Arc, as we had done for the left end-point, we have,

$$\begin{aligned}
\Rightarrow\ \ & t'^4[(M^2 + K^2 + P_1^2 - 2KP_1)] \\
& + t'^3[(2P_1P_2 - 2MN - 2MKs_0 - 2KP_2 - 2MP_1s_0)] \\
& + t'^2[(2KP_1t_0^2 - 2KP_3 + 4KP_1s_0^2 + 2P_1P_3 + P_2^2 + N^2 \\
& \qquad - 2K^2t_0^2 + 2KNs_0 - M^2t_0^2 + 2s_0NP_1 - 2s_0MP_2)] \\
& + t'[(2MNt_0^2 + 2KMs_0t_0^2 + 2P_2P_3 + 4KP_2s_0^2 + 2KP_2t_0^2 + 2s_0NP_2 - 2s_0MP_3)] \\
& + [(2s_0NP_3 - N^2t_0^2 - 2KNs_0t_0^2 + K^2t_0^4 + P_3^2 + 4KP_3s_0^2 + 2KP_3t_0^2)] \quad = \quad 0
\end{aligned}$$

Where,

$$\begin{aligned}
K &= (d_s + 2s_0)(2s_0f + fd_s + Z_0\Delta d') \\
M &= (4fd_st_0 + 8fs_0t_0 + 4fd_sZ_0 + 4fs_0Z_0 + 2Z_0\Delta d'(Z_0 + t_0)) \\
N &= (4fd_st_0^2 + 2fd_sZ_0^2 + 4fd_sZ_0t_0 + 4fs_0Z_0t_0 + 8s_0ft_0^2 + Z_0\Delta d'(t_0^2 + (Z_0 + t_0)^2)) \\
P_1 &= (4ft_0(Z_0 + t_0)) \\
P_2 &= (2fZ_0^3 - 8ft_0^3 - 8ft_0^2Z_0) \\
P_3 &= (t_0^2 + (Z_0 + t_0)^2)(2ft_0^2 - fZ_0^2)
\end{aligned}$$

A.6 Example

Inputs :

$R = 4 \cdot 25$

$f = 1 \cdot 4$ (f *is taken as about* $(R/3)$)

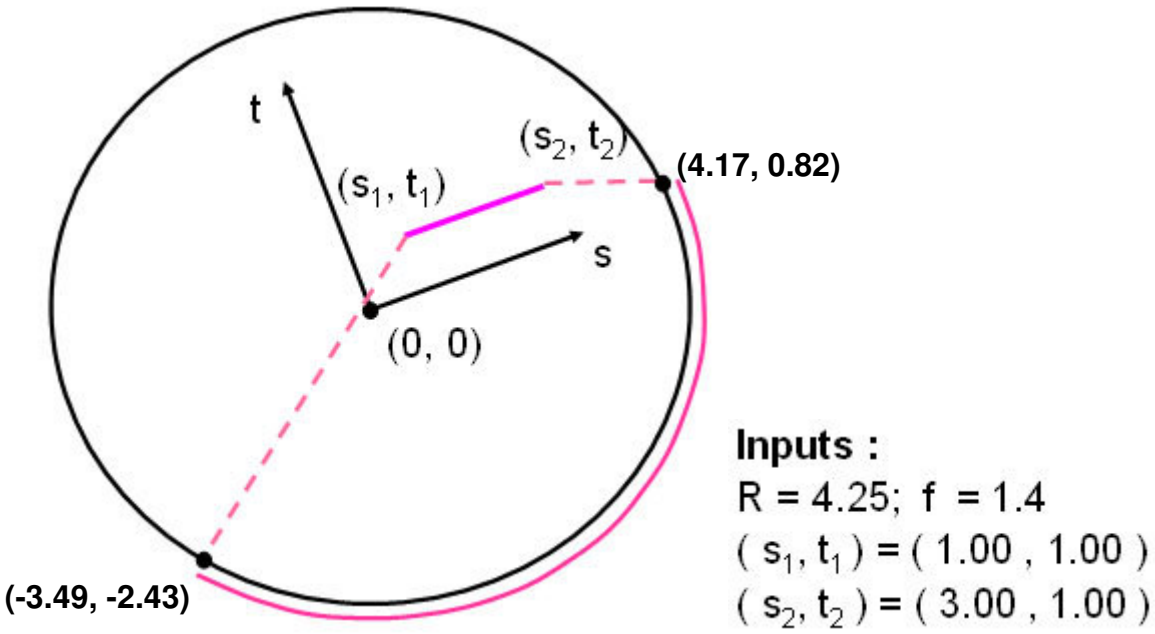


Figure A.3: Example

$(s_1, t_1) = (1\ ,\ 1)$

$(s_2, t_2) = (3\ ,\ 1)$

Output :

Left-end point of Sampling Arc : (– 3·49 , – 2·43)

Right-end point of Sampling Arc : (4·17 , 0·82)

And the remaining two points on the Sampling Circle, which are on the other side of the Line Segment:

(3·58 , 2·29) and (– 4·10 , 1·14)

APPENDIX B

Published Work

The following paper was published with Computer Graphics International 2004, based on this thesis work [Namboori, Teh and Huang, 2004].

An Adaptive Sampling Method for Layered Depth Image

Ravinder Namboori, Hung Chuan Teh, Zhiyong Huang
Department of Computer Science, School of Computing,
National University of Singapore
Singapore 117543
{namboori, tehhc, huangzy}@comp.nus.edu.sg

CGI, Jun'04, pp. 206-213
http://doi.ieeecomputersociety.org/10.1109/CGI.2004.12

Sampling issue is an important problem in image based rendering. In this paper, we propose an adaptive sampling method to improve the Layered Depth Image framework. Different from the existing methods of interpolating or splatting neighboring pixels, our method selects a set of sampling views based on the scene analysis that can guarantee the final rendering quality. Furthermore, the rendering speed is accelerated by the pre-computed patch lookup table, which simplifies the reference view selection process to a simple lookup of a hash table. We have implemented our method. The experiment study shows the advantage of the method.

Keywords: image based rendering, layer depth images, data sampling, image warping.