A FUZZY LOGIC BASED APPROACH FOR ENHANCING DEPTH PERCEPTION IN COMPUTER GRAPHICS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Zeynep Çipiloğlu

May, 2010

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Tolga K. Çapın(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Bülent Özgüç

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Hüseyin Boyacı

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

A FUZZY LOGIC BASED APPROACH FOR ENHANCING DEPTH PERCEPTION IN COMPUTER GRAPHICS

Zeynep Çipiloğlu
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Tolga K. Çapın
May, 2010

Rapid progress in 3D rendering and display technologies brings the problem of better visualization of 3D content. Providing correct depth information and enabling the user to perceive the spatial relationship between the objects is one of the main concerns during the visualization of 3D content.

In this thesis, we introduce a solution that can either be used for automatically enhancing the depth perception of a given scene, or as a component that suggests suitable rendering methods to application developers.

In this novel solution, we propose a framework that decides on the suitable depth cues for a given 3D scene and the rendering methods which provide these cues. First, the system calculates the importance of each depth cue using a fuzzy logic based algorithm which considers the user's tasks in the application and the spatial layout of the scene. Then, a knapsack model is constructed to keep the balance between the rendering costs of the graphical methods that provide these cues and their contribution to depth perception. This cost-profit analysis step selects the proper rendering methods for the given scene.

In this work, we also present several objective and subjective experiments which show that our automated depth perception enhancement system is statistically ($p < 0.05$) better than the other method selection techniques that are tested.

Keywords: Depth perception, depth cues, cue combination, computer graphics, perceptually-aware rendering.

ÖZET

BULANIK MANTIK TABANLI YAKLAŞIMLA BİLGİSAYAR GRAFİĞİNDE DERİNLİK ALGISININ ARTIRILMASI

Zeynep Çipiloğlu
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Asst. Prof. Dr. Tolga K. Çapın
Mayıs, 2010

3B görüntüleme ve ekran teknolojilerindeki hızlı gelişim, 3B görsel içeriğin uygun şekilde sunulması problemini de beraberinde getirmektedir. 3B bilgisayar görüntüleri üretilirken, kullanıcının nesnelerin uzamsal özelliklerini doğru olarak algılayabilmesini kolaylaştırmaya dikkat edilmelidir.

Bu tezde önerilen çözüm, verilen bir 3B sahnede derinlik algısının otomatik olarak iyileştirilmesinde ya da 3B uygulama geliştiricilere derinlik algısının artırılması için uygun görüntüleme yöntemlerinin önerilmesinde bir araç olarak kullanılabilir.

Sunulan çözümde, verilen 3B sahne için uygun olan derinlik ipuçları ve bu ipuçlarını sağlayan görüntüleme yöntemleri otomatik olarak belirlenmektedir. Öncelikle, bulanık mantık tabanlı bir algoritmayla, verilen sahne için her bir ipucunun önem derecesi, uygulamanın amacı ve sahnenin uzamsal özellikleri göz önünde bulundurularak hesaplanmaktadır. Daha sonra, önemli derinlik ipuçlarını sağlayan grafiksel görüntüleme yöntemlerinin maliyetlerini ve derinlik algısına katkılarını dengelemeye yönelik olarak bir knapsack modeli oluşturulmaktadır. Bu kar-zarar analizi sayesinde derinlik algısını artırma amaçlı grafiksel yöntemlerden verilen sahne için uygun olanları belirlenmektedir.

Bu çalışmada ayrıca, önerilen otomatik derinlik algısı iyileştirme sisteminin test edilen diğer yöntem seçim tekniklerine göre istatistiksel olarak ($p<0.05$) daha başarılı olduğunu gösteren birkaç nesnel ve öznel kullanıcı testine de yer verilmektedir.

Anahtar sözcükler: Derinlik algısı, derinlik ipuçları, ipucu birleşmesi, bilgisayar grafiği, algıya dayalı görüntüleme.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor Asst. Prof. Dr. Tolga Çapın for his endless support, guidance, and encouragement.

I would also like to thank to my jury members, Prof. Dr. Bülent Özgüç and Asst. Prof. Dr. Hüseyin Boyacı for spending their time to read and evaluate this thesis.

Special thanks to my family for their endless support and patience during my education.

I am also grateful to my friends for their friendship and patience during the user experiments. Especially, thanks to Abdullah Bülbül, for his support during all phases of this study.

Finally, I would like to thank to TÜBİTAK-BİDEB and EU 7th Framework "All 3D Imaging Phone" Project for the financial support during my M.S. study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motivation

3D rendering methods and display technologies such as head-mounted displays and autostereoscopic displays have advanced significantly in the past few years. This rapid development in the 3D technology also brings the need of better visualization of the 3D content. People desire to see realistic scenes and feel as if they were present in the virtual environment, especially when they are playing games or watching 3D movies. Another application area of 3D computer-generated imagery is Information Visualization. Presenting the information effectively is the main concern of the Information Visualization field. Therefore, usage of the third dimension in an effective manner becomes very important.

It is known that "what is perceived" is more important than "what is displayed". Poorly designed 3D contents will have a strong negative effect on what is perceived. In addition, incorrect usage of depth will also lead to physiological problems such as eye strain. Ware emphasizes the significance of depth perception in Information Visualization as follows [53]:

> "It is inevitable that there is now an abundance of ill conceived 3D design, just as the advent of desktop publishing brought poor use of typography and the advent of cheap color brought ineffective and often garish

> use of color. Through an understanding of space perception, we hope to reduce the amount of poor 3D design and clarify those instances in which 3D representation is really useful."

It is clear that providing correct depth information during the design of a 3D scene is very important, however, it is not easy to deal with this additional depth issue for a 3D application designer. It requires understanding of real-life depth cues that are used to perceive the spatial relationships between the objects by the human visual system. Therefore, an automated system for improving the depth perception of an input 3D scene would be very beneficial. In order to develop such a system, an algorithm that combines different depth cues and rendering methods is needed. There are a variety of studies on enhancing depth perception in computer graphics. A number of methods have been proposed to improve depth perception in 3D computer-generated imagery. However, these methods are generally limited and insufficient, since either they are proposed to operate on specific domains or they do not provide a solution to unify different depth enhancement methods appropriately. In conclusion, a comprehensive system that combines existing depth enhancement methods properly according to the given scene is required.

Overview of the System

Our aim is to develop a framework that applies refinements to a given input scene, in order to provide better depth perception. For this purpose, we need an approach to combine the techniques used for depth enhancement. Although the first approach is to provide all possible depth cues at the same time, this is not always the best solution. Providing all the cues may lead to problems such as high computational cost, unnecessary scene complexity, and cue conflicts. Hence, a system designed to enhance depth perception should consider the aspects such as the nature of the task, distance of the objects, other scene attributes and computational costs of the methods.

In this work, we propose a framework that allows the user to add different

depth enhancement methods to the scene and preview the effects of these methods. In addition, we present an algorithm that automatically selects the proper methods for the given scene, depending on the task, spatial layout of the scene, and the costs of the rendering methods. The algorithm makes use of *fuzzy logic* for determining the significance of different depth cues, and *knapsack problem* for modeling the trade-off between the cost and the profit of a depth enhancement method.

Challenges

Developing a system for enhancing depth perception in computer graphics is a challenging job due to several reasons. First of all, we cannot restrict the problem to only the use of Computer Graphics techniques. It requires a comprehensive investigation of two different domains: Human Visual Perception and Computer Graphics. Furthermore, the principles of the human visual system are not fully understood yet. For instance, there is not a single, accepted model to identify how the human visual system combines different depth cues to obtain a final percept. Thus, adapting the principles of human visual system to Computer Graphics is a challenging task.

Secondly, which depth cues and rendering methods will perform best for the given scene depends on many different factors. The proposed solution should be multidimensional, which considers many aspects such as the target task, spatial layout of the scene, distance of the objects, costs of the methods, capabilities of the devices, and so on.

Summary of the Contributions

The contributions of this thesis can be summarized as follows:

- A survey on the visual cues that are used to perceive the spatial relationship between the objects in 3D by the human visual system, the combination models for these depth cues, and rendering methods that provide these cues in computer-generated imagery,
- A tool, for 3D graphical application developers, to apply different depth

enhancement methods on a given scene, using a simple graphical user interface,

- A fuzzy logic based algorithm for automatically determining the proper depth cues for the given scene and task,
- A knapsack model for selecting proper depth enhancement methods, evaluating the cost and profit of these rendering methods,
- Demonstration of different rendering methods that are used for improving depth perception in computer-generated imagery,
- An experimental study to evaluate the effectiveness of the proposed algorithms.

Outline of the Thesis

- Chapter 2 presents a comprehensive investigation of the previous work on the topic of Depth Perception, from the perspectives of Human Perception and Computer Graphics fields.
- In Chapter 3, our proposed system for enhancing the perception of depth in an input 3D scene is explained in detail.
- Chapter 4 contains the results of an experimental evaluation of the proposed depth enhancement system.
- Chapter 5 concludes the thesis with a summary of the current system and future directions for the improvements on this system.

Chapter 2

Background

This chapter is mainly divided into two sections. In the first section, the principles of depth perception are investigated from the Perception point of view, while the second section presents the application of these perceptual principles to Computer Graphics.

2.1 Depth Perception

2.1.1 Depth Cues

Depth cues, which help the human visual system to perceive the spatial relationships between the objects in 3D, construct the core part of the depth perception. These visual cues can be categorized as pictorial, oculomotor, binocular, and motion-based cues. These depth cues are shown in Table 2.1 and explained in detail in the subsequent parts, based on the studies by Howard and Rogers [17], Shirley [47], and Ware [53].

Table 2.1: Categorization of the depth cues.

Pictorial	**Oculomotor**	**Binocular**	**Motional**
Occlusion	Accommodation	Binocular disparity	Motion parallax
Cast shadow	Convergence		Motion perspective
Linear perspective			Kinetic depth
Size gradient			
Relative height			
Texture gradient			
Relative brightness			
Aerial perspective			
Depth of focus			
Shading			

Pictorial Cues

Pictorial cues are the cues that are generally used by artists to provide 3D effect in 2D paintings.

Occlusion: Occlusion, also known as interposition, is the most basic depth cue which arises when an object occludes another object (Figure 2.1). When an object overlaps some part of the other, it is known that the object that is blocked is further. Interposition only gives binary information about the depth order of the objects; it does not provide information about the magnitude of the depth.

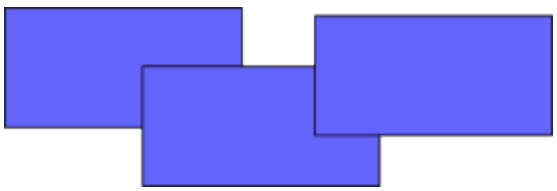

Figure 2.1: Occlusion depth cue. (The left rectangle is perceived furhtest and the right one is perceived closest.)

Cast shadow: Shadows also give information about depth. If the object is in shadow, it is further from the light source. When the object moves away from

the light source, it gets darker. The number and position of light sources are also effective on the perception of depth. Shadows of the objects on the ground facilitate the perception of the objects' relative positions by connecting them to the ground plane (Figure 2.2). While interpreting shadows, two important assumptions of the human visual system should be considered: "single, stationary light source" and "light-from above" assumptions. In other words, the human visual system assumes that there is a single, stationary light source and the illumination direction is from above, while interpreting the scenes [17].

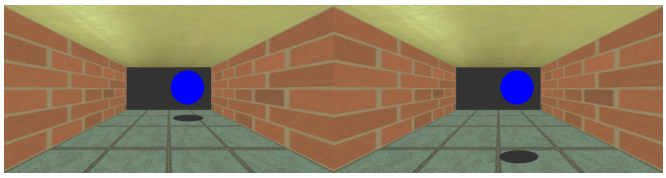

Figure 2.2: Shadow depth cue.

Linear perspective: In real life, parallel lines seem converging, as they move away, towards the horizon. This is known as linear perspective. Linear perspective is obtained by perspective projection, which scales the image by the inverse of the depth while projecting it onto its corresponding location on the projection plane. Linear perspective helps to perceive the surface inclination especially when it is textured. Moreover, a perspective ground plane aids to understand the relative positions of the objects above it. (See Figure 2.3.)

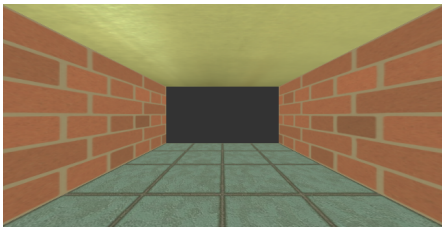

Figure 2.3: Linear perspective depth cue.

Size gradient: The size of an object is inversely proportional to the distance from the viewer. In other words, larger objects may seem closer to the viewer. Therefore, the relative distance can be inferred from the size of the objects, when the

objects are of the same type. This is known as "relative size" (Figure 2.4). "Familiar size" is another type of size gradient depth cue. In this type, "relative size" interpretation is combined with the previous knowledge about the sizes of the known objects to estimate the absolute depths of the objects. For example, people generally know the size of a car relative to a person and if the absolute distance of a human in an image is known, the absolute distance of the car in that image can be inferred from the previous knowledge about the size of a car and a human. However, according to Howard and Rogers, only long term familiarity is effective [17].

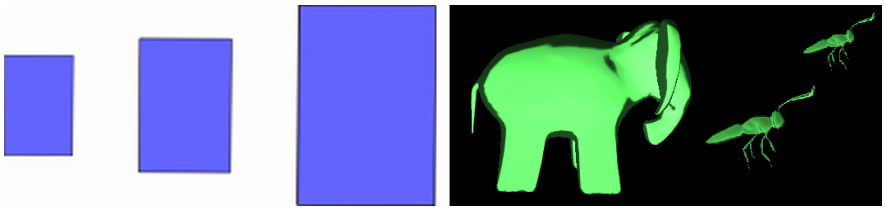

Figure 2.4: Size gradient depth cues. Left: relative size (the largest rectangle may seem closest), Right: familiar size (the lower bug seems closer than the elephant).

Relative height: The objects that are higher in the visual field are perceived as distant (left of Figure 2.5). For instance, Roelofs and Zeeman report that the lower one among two equal circles appears closer and smaller, when there is no other source of depth information [17]. When the world is divided by a horizon line; for the objects under the horizon, the objects that are closer to the horizon seem further. On the other hand, above the horizon line, the objects closer to the horizon seem closer. In the right of Figure 2.5, the clouds that are higher in the visual field seem closer.

Figure 2.5: Relative height depth cue. (Painting on the right: "The Coast of Protrieux" by Eugene Boudin)

Texture gradient: In textured surfaces, when the surface gets further away, the texture becomes smoother and finer (Figure 2.6). Texture elements are spaced more densely with the increasing distance. Hence, texture gradient is helpful especially for perceiving the shape and slant of a surface. According to the experiments by Gibson, perceived inclination of a surface increased with the increase in texture gradient and this effect is more obvious for regular textures than irregular textures [17].

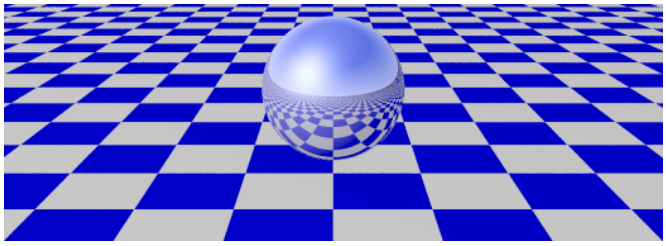

Figure 2.6: Texture gradient depth cue.

Relative brightness: The intensity level of an object varies with depth. Brighter objects are prone to be seen closer to the viewer. Hence, by comparing the intensities of the objects, relative depth information can be obtained. The brightness level of an object approaches to the background as its distance to the viewer increases. For instance, in a dark environment, as the distance to the viewer increases, the object gets darker; conversely, if the background color is bright, further objects seem brighter. (See Figure 2.7)

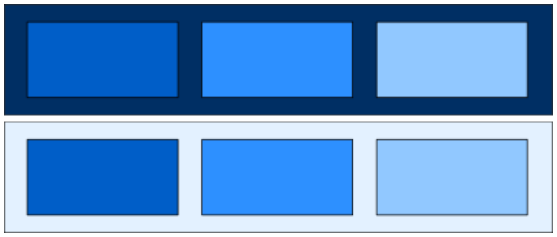

Figure 2.7: Relative brightness depth cue. (On the upper part, the brightest rectangle is perceived closest; on the lower part, the darkest rectangle is perceived closest.)

Aerial perspective: Atmospheric scattering reduces the brightness and the contrast in the distant parts of the scene. In other words, in a natural scene, further objects seem hazy and bluish due to the scattering of the light in the atmosphere. Hence, aerial perspective primarily increases the perceived distance. It may affect the perception in different ways, however: it may obscure the texture gradient or it may cause an object to be seen closer and larger by blurring the borders of the object [17]. (See Figure 2.8.)

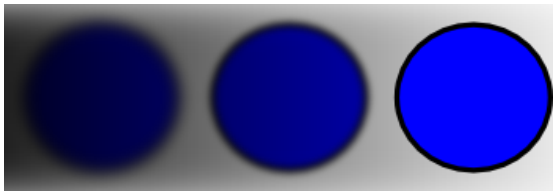

Figure 2.8: Aerial perspective.

Depth of focus: In essence, depth of focus is a term used in optics. The depth of focus of an eye lens is defined as "the range over which objects are in focus when the eye is adjusted for a particular distance" [53]. Our eyes fixate on different objects in the world to bring them to sharp focus. The objects other than the object in the sharp focus seem blurry. Ordinal information about the spatial relationship of the objects can be obtained from this property. (See Figure 2.9.)

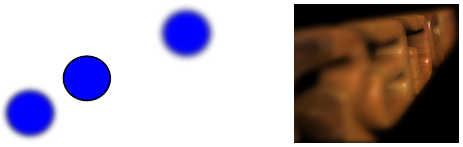

Figure 2.9: Illustrations of depth of focus depth cue.

Shading: Shading is defined as "the variation in irradiance from a surface due to changes in the orientation of the surface to the incident light and/or variations in specularity" [17]. Shading provides important information about the surface shape by enabling the observer to distinguish between convexities and concavities; because the shade of a surface depends on the viewpoint, orientation with respect to the light source, and surface reflectance characteristics [47]. The number of

light sources is also effective in perceiving the surface shape. The process of shape perception from shading is known as "shape-from-shading". However, shading is assumed to give ambiguous depth information when used alone [17]. It requires knowledge about the direction of illumination. The human visual system has a prior of a simple lighting model with a single, stationary light source located left-above [17]. In Figure 2.10, spheres with different shading models are shown. As shown in the figure, the specularity, transparency, and other shading-related properties of the spheres affect the perceived convexity/concavity of the objects in different ways.

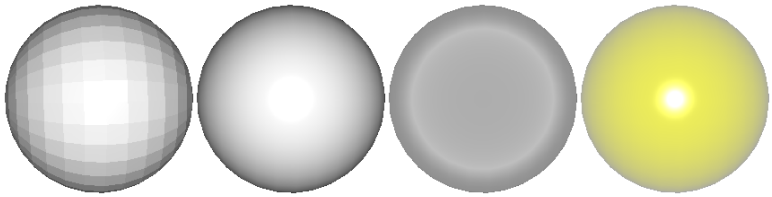

Figure 2.10: Spheres with different shading models to illustrate the shading depth cue. Left to right: flat shading, Phong shading, Cook-Torrance Shading, Gooch shading.

Oculomotor Cues

Depth cues that are caused by the muscular control of the eye lens are known as oculomotor cues.

Accommodation: The process of the distortion in the eye lens to fixate on a point is called accommodation. The amount of accommodation that the eye lens performs to focus on an object varies with depth. In a limited range, absolute depth information can be obtained using the accommodation of the eye lens. However, this is a weak depth cue and ineffective beyond about 2m [47]. (See Figure 2.11.)

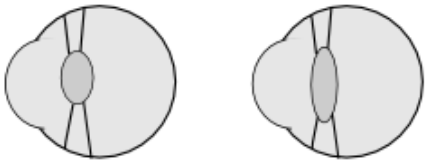

Figure 2.11: Accommodation depth cue. (Left: The eye lens is distorted to see a close object, right: the eye lens is distorted to see a further object.)

Convergence: Convergence, or vergence, is another oculomotor cue which strongly interacts with the accommodation. It is the fixation of the eyes towards a single location in space in order to maintain a single binocular vision. Also, the angle between the eye and the focus point, which is shown as θ in Figure 2.11 is called as convergence angle. The increase in the convergence angle indicates that the fixation point comes closer (Figure 2.12). In the human visual system, accommodation helps to determine the convergence angle, and convergence angle assists in setting the focal length [47].

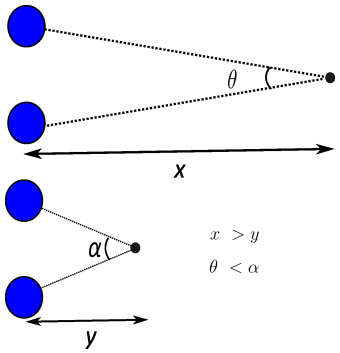

Figure 2.12: Convergence depth cue.

Binocular Cues

Binocular disparity: Left and right eyes look at the world from slightly different angles, which results in slightly different retinal images (Figure 2.13). This provides binocular vision. This is a very strong cue that provides true 3D vision, unlike pictorial cues. However, binocular disparity becomes ineffective for far distances. The human visual system is able to match the points in one retinal

image to the corresponding points in the other retinal image, which is referred to as correspondence process [47]. In order to completely match the two retinal images, the disparity between the two images should not exceed a threshold value. If the disparity becomes large, double vision called diplopia occurs. According to Patterson and Martin, the maximum disparity before the fusion breaks down is only 1/10 degree [53].

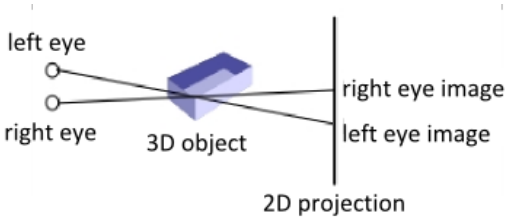

Figure 2.13: Binocular disparity depth cue.

Motional Cues

When an object or the viewer himself is moving, the retinal image will change due to the motion. From the difference in the retinal images during motion, some clues about the depth can be obtained. Understanding the relative depth and shape of the objects using this type of motion-related depth cues is known as structure-from-motion [53]. According to Rogers and Graham, structure-from-motion cues are at least as important as binocular disparity in providing depth information [53]. There are different kinds of structure-from-motion cues:

Motion parallax: As the user moves his eyes side to side, the images of the closer objects move more in the visual field than those of further objects. This is because the angular speed of an object is inversely related to the distance from the viewer. In Figure 2.14, the angular velocity (α) of the closer (blue) object is greater than the angular velocity (Θ) of the further (red) object. As an example, in a moving car, close objects such as trees translate very fast, while further objects such as mountains move slowly.

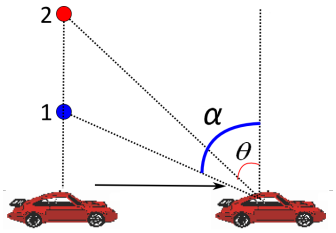

Figure 2.14: Motion parallax depth cue.

Motion perspective: Motion perspective is very similar to motion parallax. This time, the user is stationary and the objects are moving. In this scenario, the velocity of the retinal images of the further objects seem slower than the near objects.

Kinetic depth: Motion can also be used in perceiving the three-dimensional shape of an object, if there is a rotation around an axis perpendicular to the direction of view [47]. This is called kinetic depth effect. The overall shape of an arbitrary object can be perceived better when it rotates around its local axis, since the ambiguities due to the projection from 3D to 2D are resolved with the rotation (Figure 2.15). While interpreting the shape of an object, the human visual system assumes that the object is rigid [53].

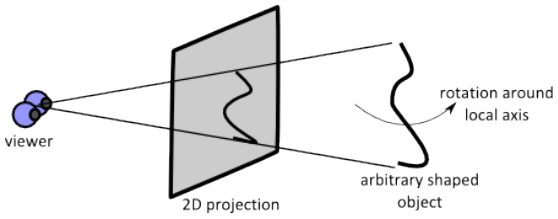

Figure 2.15: Kinetic depth cue.

2.1.2 Depth Cues in Combination

In the field of human visual perception, how the human visual system unifies different sources of depth cues into a single knowledge is a widely-investigated topic. Many studies have investigated the interaction of different cues. There is

not a single, accepted cue combination model, however.

Cue Combination Models

The mostly-accepted models of cue interaction are generally the variations of the following categories: *cue averaging, cue dominance, cue specialization, range extension,* and *probabilistic models* [17].

Most of the research on cue combination focuses on the *cue averaging* models, in which each cue is associated with a weight determining its reliability. The overall perception is obtained by summing up the individual depth cues multiplied by their weights [17]. Maloney and Landy [30] and Oruc et al. [40] present a weighted-linear combination model based on the cue summation and averaging models. A more specialized form of linear combination is presented by Clark and Yuille [9] by distinguishing between weak and strong fusion. In the weak fusion model, the interactions between depth cues are omitted; whereas the strong fusion estimates the nonlinear interactions between the individual cues. Landy et al. suggest the modified weak fusion approach, in which cue interactions are limited to cue promotion [28]. In cue promotion, the cues that do not supply complete depth information are promoted by using information from richer cues [17].

The variations of linear combination models are not designed to handle severe cue conflicts. *Cue dominance* is a model proposed to consider conflicting situations, by vetoing some sources of information totally [17]. In other words, if two depth cues provide conflicting information, one of them may be suppressed and the final percept may be based on the other cue. For instance, in Figure 2.16, occlusion and relative size depth cues give conflicting information about the depth order. In such cases, the judgement is generally based on the occlusion cue.

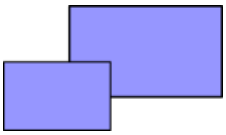

Figure 2.16: An example of cue conflict.

Cue specialization models are based on the idea that different cues may be

used for interpreting different components of a stimulus. For instance, when the aim is to detect the curvature of a surface, binocular disparity is more effective; on the other hand, if the target is to interpret the shape of the object, shading and texture cues contribute more to the result [17].

Range extension is another cue interaction type, whose main concern is that different cues may be effective in different ranges. For example, binocular disparity is a strong cue in the near distances, while perspective becomes more effective at far distances.

Lastly, Bulthoff and Yuille present a *probabilistic* approach to estimate the cue interactions by considering some prior assumptions of the human visual system on the scene and material attributes, using a Bayesian framework [7]. One example to the prior assumptions of the human visual system is single light source assumption. Rigidity and surface smoothness are other known examples of these prior assumptions. In this Bayesian framework, cues are assumed to be strongly interacting with each other.

Apart from the above cue integration models, there are a variety of experimental studies that investigate the interaction between different depth cues. In one of these studies, Hubona investigates the relative contributions of binocular disparity, shadows, lighting, and background scenes to 3D perception [19]. As the most obvious result, he found that stereoscopic viewing strongly improved the depth perception with respect to both accuracy and the response time. On the other hand, there were further interactions between other cues. Mather and Smith focus on stereopsis and image blur cues; and suggest that when both cues are available, stereopsis dominates [34]. Also, they state that image blur is used by the visual system for far distances. Wanger et al. explore the effects of pictorial depth cues on perceiving the depth, and concludes that the effectiveness of a depth cue is highly affected by the target task [52].

Task-based Depth Perception

A number of researchers establish their work on the *cue specialization* model described in the previous section and accept the target task as the most significant

factor on determining the visual cues that enhance the depth perception [4, 12, 45, 52]. For instance, according to the experimental results obtained by Wanger et al. [52], shadow and perspective significantly increase the accuracy in positional tasks while perspective decreases the performance in orientation related tasks.

However, the tasks discussed in the above studies are very limited. Ware presents a more comprehensive list of the tasks and a survey of the depth cues according to their effectiveness under these tasks [53]. In his work, the tasks are categorized as listed below. The detailed explanation of these tasks is in Section 3.2.1.

- Tracing data paths in 3D graphs
- Judging the morphology of surfaces and surface target detection
- Finding patterns of points in 3D space
- Judging the relative positions of objects in space
- Judging the relative movement of self within the environment
- Reaching for objects
- Judging the "up"direction
- Feeling a sense of presence (aesthetic impression)
- Navigation (way finding)

For instance, according to his investigations, perspective is a strong cue when the task is "judging the relative positions" or "judging the up direction"; while it becomes ineffective for the tasks "tracing data paths in 3D graphs" and "finding patterns of points in 3D space". Another example is that stereo viewing is a strong cue for the tasks other than "feeling a sense of presence" when coupled with near vicinity. According to Ware et al., stereoscopic viewing and kinetic depth together significantly increased the accuracy when the task is "tracing data paths in 3D graphs" [55].

Distance-based Depth Perception

Another factor that determines the effectiveness of a cue is the distance from the viewer to the object of interest, as the *range extension* models suggest. Cutting and Vishton provide a distance-based classification of depth cues by dividing the space into three ranges and investigating the visual sensitivity of the human visual system to different depth cues in each range [10]. According to his measurements; the strength of the pictorial cues are invariant with the distance; and the effectiveness of some of the sources such as binocular disparity, motion parallax, and oculomotor cues are dissipating with the distance; whereas aerial perspective is the only cue that becomes stronger as the distance increases. The effectiveness of each depth cue according to the distance is shown in Figure 2.17.

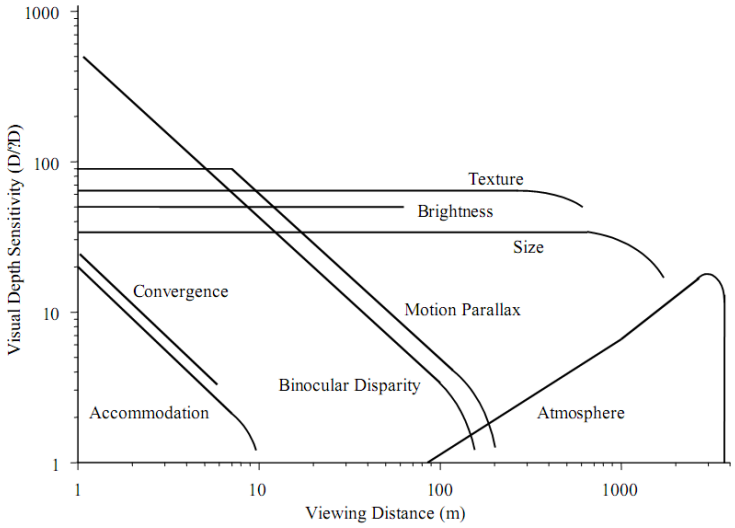

Figure 2.17: The effectiveness of depth cues as a function of distance [41].

2.2 Depth Perception in Computer Graphics

Based on the depth cues and principles discussed in the previous section, different rendering methods have been developed for enhancing depth perception in 3D rendered scenes.

2.2.1 Depth Enhancement Methods

In this section, we discuss different rendering methods that are used for improving the depth perception in Computer Graphics. Table 2.2 contains a categorization of these methods according to the depth cues they provide. Afterwards, the methods in each category are explained in more detail.

Table 2.2: Depth perception enhancement methods according to depth cues.

Depth Cues	**Depth Enhancement Methods**
Occlusion, Size gradient, Relative height	Matrix transformations, ground plane, room, placing objects of known sizes, dropping lines to ground
Relative brightness, Aerial perspective	Fog, proximity luminance
Texture gradient	Texture mapping, bump mapping
Shading, Shadow	Cast shadows, ambient occlusion, vicinity shading cool-to-warm shading, boundary enhancement
Linear perspective	Perspective projection
Depth of focus	Depth-of-field
Accommodation, Convergence, Binocular disparity	Stereo rendering, multi-view rendering
Motional Cues	Eye tracking, face tracking Mouse, keyboard controlled motion

Occlusion, Size Gradient, Relative Height

It is possible to obtain the cues *occlusion*, *size gradient*, and *relative height* by transforming the objects in the scene. For instance, scaling the size of the objects of interest to occlude the other objects is an example for this kind of transformation [49]. Moreover, for the *relative height* depth cue, drawing lines from the objects to the ground plane (Figure 2.18) is a commonly-used artificial method to make the height between the object and the ground more visible [53].

A *ground plane* or a *room* facilitates the interpretation of the cues *relative height* and *size gradient*. In addition, *placing objects of known sizes* is a technique for enabling the user to judge the sizes of unknown objects more easily [53].

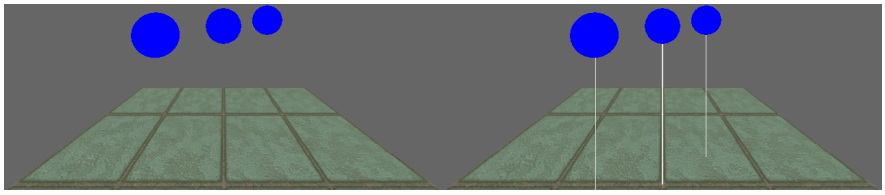

Figure 2.18: Dropping lines to the ground as an artificial depth cue [53].

Another point to consider is that most of the 3D scenes are rendered under *perspective projection*, which already provides *size gradient* and *relative height* cues to some extent. Using additional transformations for *size gradient* and *relative height* may damage the understandability of the perspective projection and may cause cue conflicts.

Relative Brightness, Aerial Perspective

In order to add atmospheric effects to the graphical contents, it is very common to use *fog*. *Fog* is obtained by interpolating the color of a pixel between the surface color and the fog color with respect to the distance of the object [2]. The distant objects are highly exposed to fog when compared to the closer objects. Thus, the effect of the fog increases with the depth. It provides both *aerial perspective* and *relative brightness*, since the distant objects seem more hazy and blurry and the contrast decreases with the depth.

To make the *relative brightness* more obvious, Dosher et al. have proposed another method called *proximity luminance covariance*, which alters the contrast of the objects in the direction of the background color as the distance increases [53]. *Proximity luminance covariance* generates an atmospheric effect, as it makes closer objects seem brighter and further objects seem darker. Lastly, *cool-to-warm shading*, which will be discussed in more detail in the section related to *shading* cue, also provides atmospheric effects to some extent.

Texture Gradient

In Computer Graphics, surfaces are covered with textures to obtain *texture gradient* cue. Traditional texture mapping does not need to be discussed here, since there are several *bump mapping* techniques that improve the surface details and 3D appearance of the objects when compared to ordinary texture mapping. These techniques are normal, parallax, and relief mapping, which are implemented by modifying the per-pixel shading routine.

In *normal mapping*, a normal map is stored in addition to the actual texture map, to perturb the shade of each pixel according to the normal retrieved from the normal map of that pixel [2]. *Parallax mapping* improves this idea to provide parallax effect, by enabling the bumps to occlude each other as the user moves [2]. This parallax effect is performed by displacing the texture coordinates according to the value of the height map at that pixel and the view angle relative to the surface normal [23]. *Relief mapping* presents more accurate results with self-occlusions, self-shadows, and parallax effect [39]. In this technique, relief textures, which can be considered as extended textures with an orthogonal displacement per texel, are converted into ordinary textures using warping equations and mapped to the surface [39]. Figure 2.19 shows a comparison of normal and parallax mapping to the traditional texture mapping and Figure 2.20 shows an example of relief mapping.

Figure 2.19: Left ro right: traditional texture mapping, normal mapping, parallax mapping.

Figure 2.20: Relief mapping example [38].

Shading and Shadows

In visual perception, *shading* and *shadow* cues are examined together, since it is difficult to separate them totally. In Computer Graphics, it becomes harder to split them as they both depend on the illumination calculation and most of the shading methods produce shadows at the same time.

Mamassian et al. [31] and Kersten et al. [26] show the effect of moving cast shadows on interpreting the shapes of the objects and the spatial layout of the scene. This interpretation is done based on the idea that cast shadows move according to the movement of four factors: light source, background, object, and viewpoint. The human visual system combines the information from these four sources and infers the spatial layout of the 3D scene.

Therefore, shadows would be a useful tool especially for perceiving the relative positions of the objects. Shadow generation techniques are generally slow, however. For this reason, real-time shadows that are generated by approximation techniques are needed.

One example of approximated shadows is proposed by Zhang and Crawfis [57]. This method keeps an accumulated shadow buffer and uses convolution to splat the shadow across the slices. In addition to saving computational time, this algorithm produces shadows that are visible in the range of 0 and 1, not only 0 or 1.

Another point to consider is the shadow quality. Wanger and Leonard investigate the effects of shadow quality on perceiving the distance, shape and size of objects; and finds no significant effect of the shadow quality on the perception of spatial layout [51]. On the other hand, they claim that soft shadows have a strong negative effect, especially on shape perception tasks. Kersten et al. also support the claim of the ineffectiveness of shadow quality on perception [25]. Thus, computationally cheaper hard shadows are generally adequate and more effective for depth perception than expensive soft shadows. On the other hand, the positive effect of the soft shadows on the realism of the scenes should not be oversighted.

Shading is another important cue to aid shape and depth perception. In Computer Graphics, there are numerous shading models and techniques. However, most of the time, these models require a complete illumination calculation, which requires high rendering cost. Hence, several techniques have been developed in order to approximate the global illumination calculation for real-time rendering. Some of these techniques, such as ambient occlusion and vicinity shading, are discussed below.

The *ambient occlusion* technique aims to increase the realism of 3D graphics in real time without a complete global illumination calculation. Most of the proposed ambient occlusion methods require pre-computation for the visibility of vertices to each other. Bunnell presents a new technique, *dynamic ambient occlusion*, for calculating the light transfer from diffuse surfaces using GPU [8]. The algorithm assumes that each polygon mesh is a set of surface elements that can emit, transmit or reflect light. Firstly, the mesh is converted into surface elements, where each vertex is assumed to be a surface element. Then, for each surface element, an accessibility value, which represents the amount of hemisphere above the surface element not occluded by the geometry, is calculated by approximation. The surfaces are darkened according to the accessibility values. This method is efficient because it works without calculating the visibility of one element from another. The effect of this method can be seen in Figure 2.21. Tarini et al. combine this method with edge cueing methods in order to better visualize molecular data in real time [50]. Shanmugam et al. divide the ambient occlusion process into two parallel domains: detailed (high-frequency) domain and distant

(low-frequency) domain. The high-frequency approach uses an image-space approximation for ambient occlusion, whereas the low-frequency approach performs a spherical approximation using the GPU [46].

Figure 2.21: Left: Phong shading. Right: ambient occlusion. (The 3D model is Hebe.3ds from the sample models of AMD RenderMonkey.)

The *vicinity shading* technique provides perceptual cues on the relative depth, without a complete illumination calculation. Stewart assumes that the perception of a surface is superior under uniform, diffuse lighting compared to point source lighting; and based on this idea, each surface point is shaded according to uniform diffuse lighting that is blocked only in the proximity of that point [48]. Even though vicinity shading seems to provide better perceptual cues, it should also be verified by user experiments. The comparison of the vicinity shading with regular diffuse plus specular shading is shown in Figure 2.22.

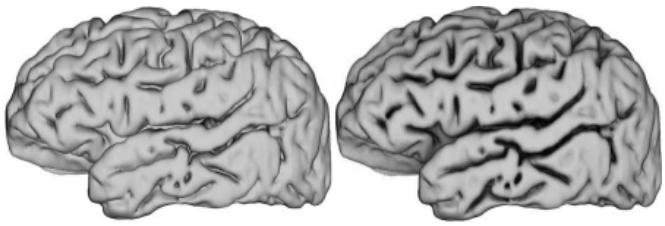

Figure 2.22: Left: regular diffuse plus specular shading. Right: vicinity shading. (©2003 IEEE. This figure has been taken from A. J. Stewart, "Vicinity Shading for Enhanced Perception of Volumetric Data," Proceedings of the 14th IEEE Visualization, page 47, 2003 [48].)

Other than the above shading and shadow approximation techniques, there are

also non-photorealistic (NPR) shading models, which aim to provide additional cues about depth. A technique that is commonly used by technical illustrators is known as *cool-to-warm shading*. According to this method, cool colors (blue tones) are used for the distant objects and warm colors (yellow tones) are used for near objects. This idea is also used by many researchers in computer graphics [13, 14, 42]. Cool-to-warm shading, which is also known as Gooch shading, is performed by interpolating between cool colors to warm colors according to the distance from the light source. This kind of shading provides atmospheric effect on the scene. Figure 2.23 demonstrates the effect of cool-to-warm shading.

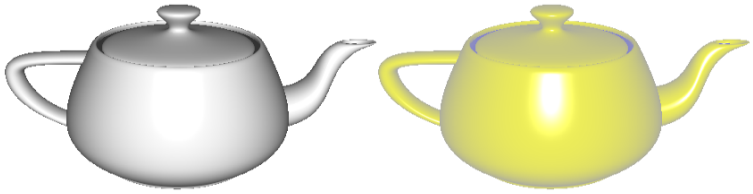

Figure 2.23: Left: Phong shading. Right: cool-to-warm shading.

Boundary enhancement using silhouette and feature edges is a commonly-used tool in NPR. There are different methods to extract feature edges of geometrical models. One method uses geometric buffers to store the geometrical properties of an image and takes these G-buffers as input to extract the discontinuities in the image. Nienhaus et al. [37] and Saito et al. [44] use this idea to find silhouette and crease edges in an image. The limitation of this method is that it cannot be applied on transparent images.

Markosian uses another approach based on Appel's hidden line algorithm, to solve the trade-off between the performance and the accuracy of determining key edges [33]. This method identifies the silhouettes based on a probabilistic approach. Not every silhouette is rendered in every frame, instead, larger silhouettes are rendered. A probability value, which is inversely proportional to the dihedral angle of that edge, is assigned to each edge. The edges with the highest probabilities are examined, and when a silhouette edge is found, the adjacent edges are also checked. In this way, checking a small fraction of the edges suffices to detect silhouette edges.

McGuire and Hughes propose a hardware solution for detecting feature edges in terms of local, per-vertex calculations by introducing the "edge mesh", which packs the information about an edge into the vertex attributes [35]. The edge mesh is calculated in a pre-processing step and sent to the vertex shader to be used in extracting the key edges such as silhouettes, creases, and boundaries. The algorithm requires a pre-processing step and the edge mesh is nine times larger than the original mesh.

The above edge enhancement methods operate on the geometry of the scene elements, and they are highly dependent on the scene complexity. As an alternative to these methods, an image-space approach is proposed based on the idea of retrieving the depth information from depth buffer, and using this data to enhance the depth effect on these areas in real time. Luft et al. use this idea to enhance images that contain depth information [29]. In this method, the difference between the original and the low-pass filtered depth buffer is computed to find spatially important areas. Then, color contrast on these areas is increased to give better 3D effect. Tarini et al. [50] also employ the depth buffer to perform depth-aware contour lines and contrast changes for increasing the visibility of borders of the atoms in molecular visualization.

Halos are also delegated for enhancing the perception of 3D objects, especially in volume rendering [5, 42]. In the method of Bruckner et al. [5], an interactively defined halo transfer function is used to determine the halo intensities. The method first identifies the regions to emit halo, then generates the halo intensity value field. Afterwards, this intensity field is converted to color and opacity values with the help of a halo profile function. No pre-computation is needed since the halo contribution is computed during the volume rendering. However, which settings of halo intensities perform the best is an open issue. An example of the halo method by Bruckner et al. is shown in Figure 2.24.

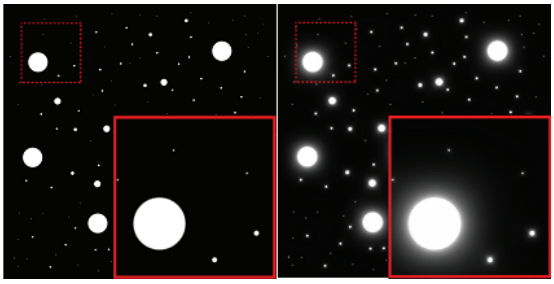

Figure 2.24: Halo effect (Left: without halos, Right: with halos). (©2007 IEEE. This figure has been taken from S. Bruckner and E. Groller, "Enhancing depth-perception with flexible volumetric halos," IEEE Transactions on Visualization and Computer Graphics, 13(6): 1344-1351, 2007 [5].)

Linear Perspective

In order to provide *linear perspective*, *perspective projection*, in which the sizes of the objects are scaled in an amount inversely proportional to the distance, is used while projecting the 3D scene to 2D screen. Moreover, a *ground plane*, or a *room* will reveal the effect of the perspective, especially when the surface is covered with suitable textures such as grid or checkerboard [53].

Depth-of-focus

A method called *depth-of-field* is used to simulate the *depth-of-focus* cue. According to this method, objects in the range of focus are rendered sharp, while the objects outside of this range are rendered blurry and the blurriness level increases as the objects get further away from the range of focus. Haeberli and Akeley implement this method by using the accumulation buffer to accumulate the images rendered from various viewpoints across the aperture of the lens [15]. Kraus and Strengert present another solution, which decomposes the original image into several sub-images according to the pixel depth, blurs these sub-images uniformly, and blends the blurred sub-images in back-to-front order to obtain the depth-of-field image [27]. An example application of this method is shown in Figure 2.25.

Figure 2.25: Result of depth-of-field method by Kraus and Strengert [27]. (©2007 Wiley-Blackwell. This figure has been taken from M. Kraus and M. Strengert, "Depth-of-field rendering by pyramidal image processing," Computer Graphics Forum, 26: 645-654.)

Accommodation, Convergence, Binocular Disparity

None of the rendering methods discussed above provides the binocular and oculomotor depth cues when used in 2D displays. In order to obtain these cues, we need an apparatus that provide multiple views of a 3D scene. For this purpose, there are several technologies such as anaglyph glasses, shutter glasses, parallax barrier displays, lenticular displays, holographic displays, and head-tracked displays [11]. These technologies provide at least two different views of the scene and enable binocular and oculomotor cues in computer generated scenes. The principles underlying these technologies are out of the scope of this work and the reader is referred to the references [11] and [16] for further details.

There are not specific methods to provide *accommodation* and *convergence* cues in Computer Graphics, since they are related to the muscular control of the eye lens. However, *multi-view rendering* comprises these cues too. In addition, in order to prevent the eye strain caused by the improper calculation of the camera parameters in multi-view rendering, Ware et al. propose several adjustments. In this work, eye separation is dynamically calculated as a function of depth, and a method called *cyclopean scale* is used to transform the scene to a proper range where the nearest part of the scene is located just behind the screen [54] (Figure 2.26).

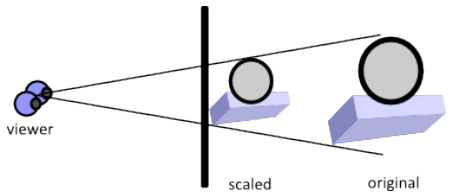

Figure 2.26: Cyclopean scale [54].

Motion Related Cues

The methods used for controlling the motion of the scene objects can be employed in different ways for providing *motion parallax*, *motion perspective*, and *kinetic depth* cues. For instance, rotating an object around its local axis can be used for obtaining *kinetic depth* effect. Controlling the movement of the scene objects using different input devices such as mouse, keyboard or joystick could provide *motion perspective*. In addition, tracking the user's position by head-tracking, eye-tracking or face-tracking and controlling the motion of the scene elements according to the information coming from the position of the user can be a tool for *motion parallax*.

Bulbul et al. propose a face tracking method to be used as a means for human computer interaction on mobile devices, based on the motion parallax effect [6]. A color-based tracking algorithm is presented in this study, to use the users face position as analogous to the camera position in a 3D graphics application. By this way, the user's head movements control the position of the camera and a motion parallax effect is obtained by enabling the user to see the scene from different viewpoints. The reader is referred to the survey on face tracking algorithms [32] for different solutions.

2.2.2 Depth Enhancement Methods in Combination

In the previous subsection, we have presented the rendering methods that are used for enhancing depth perception in Computer Graphics. Although these methods provide significant improvements on the depth perception when applied

individually, there are not sufficient studies that concern about the combination of these methods.

Tarini et al. propose a system for enhanced visualization of molecular data [50]. In this work, ambient occlusion and edge cueing schemes, such as depth aware contour lines and halo effect, are applied for molecular visualization. This system provides significant enhancements on visualization; however, it is domain specific and does not concern about how to combine different methods.

Weiskopf and Ertl developed a more comprehensive depth cueing framework based on the principles of color vision [56]. In this study, only color properties such as intensity and saturation are employed for providing depth cues by transforming the color values according to the distance. In addition, they introduce a concept called semantic depth cueing, which applies color transformations to make the important objects more visible.

There are a number of patents in this area. One of these patents presents a framework for integrating some of the monocular depth cues including shading, brightness, occlusion, and image blur [49]. In this method, the original image is segmented and objects are identified as foreground, background, or object of interest. Then, several intermediate images are created. In one of these intermediate images, the object of interest casts shadow; in the second one, the brightness level of the object of interest is increased; and in the third one, the background and foreground objects are blurred. Lastly, these intermediate images are combined and the sizes of the objects of interest are increased to occlude the other objects in the scene [49]. At the end, an image with increased depth effect is produced and the object of interest is made more visible.

As the literature survey indicates, there is a lack of comprehensive frameworks for uniting different methods of depth enhancement. The current solutions are limited, as they are either domain specific or they involve only the combination of several methods. Hence, there is a need for a framework that puts all the depth cues and different rendering methods together in an appropriate manner, to provide better 3D perception.

Chapter 3

System for Enhancing Depth Perception

In this work, we propose a system that enables the 3D graphical application developers to apply different methods of depth perception in an easy way. In this system, the user is able to select the methods he wants to apply and preview the scene with the selected methods using the graphical user interface of the system. Although this functionality facilitates the job of the content creator significantly, this is not the inventive part of the framework. The system we propose presents an algorithm for automatically selecting the proper depth cues for the current scene and the rendering methods that provide these depth cues.

3.1 General Architecture

In the system, while automatically selecting the suitable cues and rendering methods for the given scene, we consider the following factors:

- the distance of the objects in the scene,
- important tasks in the application,

- the suitability of the spatial layout of the scene for the cues,
- and the costs of the rendering methods.

The system takes the above items as input and calculates the weights of each depth cue. According to the weights of the cues and the costs of the rendering methods, suitable methods for the given scene are determined. Hence, our algorithm can be considered as a mixture of the *cue averaging*, *cue specialization*, and *range extension* models of cue combination described in Chapter 2.

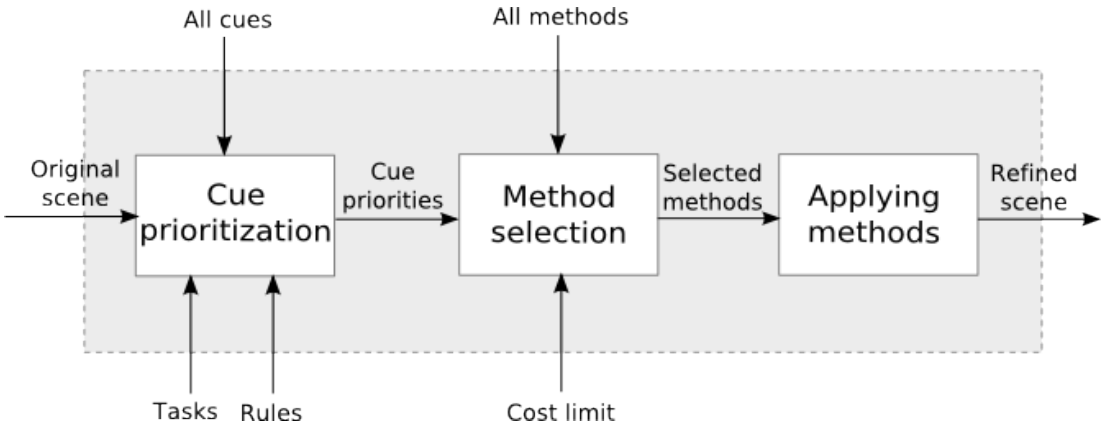

Figure 3.1: General Architecture of the System.

The general architecture of the automatical depth perception enhancement process can be seen in Figure 3.1. Our approach first determines the priority of each depth cue based on the task, distance of the objects and several scene attributes, using fuzzy logic. The next stage is to decide on the rendering methods that provide the cues whose priorities are calculated as high in the previous stage, since there are generally more than one rendering methods that provide the same cue. In this stage, we consider the costs of the methods if there is a cost limitation and try to solve the cost and cue priority trade-off. After selecting the proper rendering methods, we apply these methods to the given scene and produce a refined scene with a better depth perception.

3.2 Cue Prioritization

The first stage of our algorithm is cue prioritization as shown in Figure 3.1. The purpose of this stage is to determine which depth cues are appropriate for the given scene. At the end of this stage, a priority value, which represents the effectiveness of that cue for the given scene, is assigned to each depth cue; considering the user's tasks, distance of the objects, and different scene characteristics.General architecture of this stage is shown in Figure 3.2.

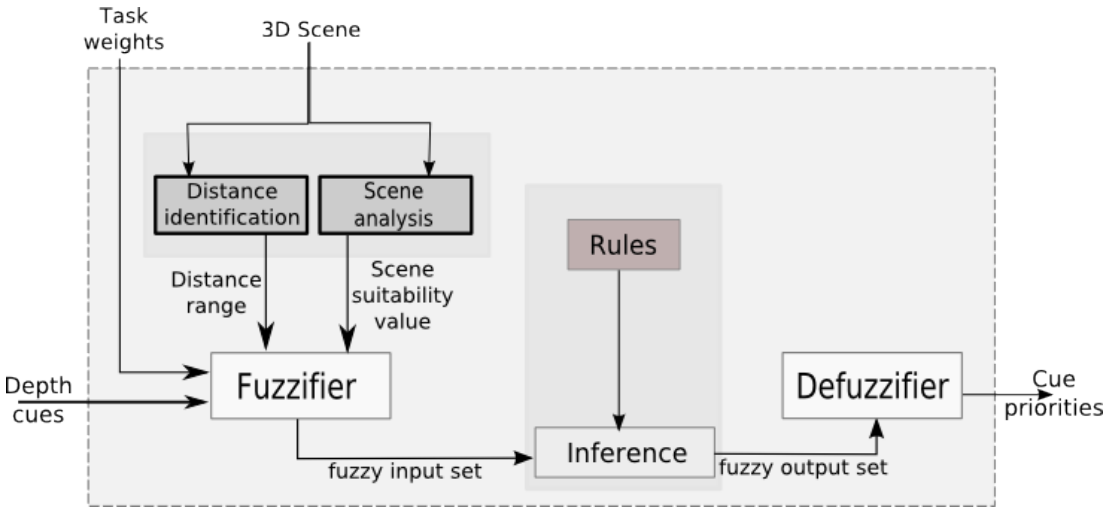

Figure 3.2: Fuzzy Cue Prioritization Stage.

The depth cues used in our system are described in detail, in Section 2.1.1. The system maintains a cue priority vector which stores the priority values for each depth cue. The priority values are in the range of (0, 1) and all the cues have 0 priority, initially. At the end of this stage, the cue priority vector is filled with the values that indicate how strong the corresponding cue is for the given scene.

Why Fuzzy Logic?

In order to calculate the cue priorities, our choice is to fuzzy logic as the decision making method. A short tutorial on what fuzzy logic is and how it is applied is provided in Appendix A. It is recommended to read that section first, for the

readers without prior knowledge on fuzzy logic.

First of all, according to Mendel, there are two forms of problem knowledge: objective and subjective. Objective knowledge is used in engineering problem formulations such as mathematical equations, while subjective knowledge contains vague terms such as "small", "big", "too many", and etc. [36]. Most of the time, binary logic is insufficient to represent such kind of subjective knowledge. In binary logic, the truth values are 0 or 1; however, in fuzzy logic, these values can be any quantity between 0 and 1.

Secondly, fuzzy logic systems are widely used in image processing, pattern recognition, controlling robots and artificial intelligence applications. Fuzzy logic is also used to model complex systems such as human intelligence, perception, and cognition. Some examples of the use of fuzzy logic in Perception are [3, 43, 20].

Additionally, the problem of combining different depth cues and rendering methods depends on many factors such as task, distance, cost, and etc. Fuzzy logic systems provide a robust solution for this kind of multi-input systems whose mathematical modeling is difficult.

In conclusion, our problem covers two different domains: Human Visual Perception and Computer Graphics. We consider the Human Visual Perception domain as the subjective part, and we prefer to formulate the subjective part of our problem using fuzzy logic since the domain contains really vague terms such as "strong", "weak", "effective", and etc.

3.2.1 Fuzzification

In the fuzzification step of the algorithm, crisp set of input variables are converted to fuzzy set of variables as in other fuzzy logic systems. In this process, linguistic variables are defined for each input variable, firstly. Then, linguistic terms that correspond to the values of the linguistic variables are defined. Lastly, membership functions are constructed to quantify the linguistic terms. In this system, we have three types of input variables: task related, distance related, and

scene layout related variables. Each of these variables is explained in detail, in the following sections.

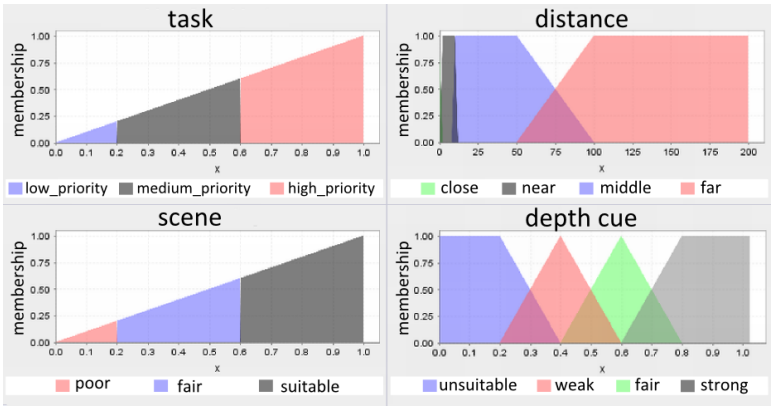

Figure 3.3: Membership functions.

Tasks

One of the inputs to the fuzzy cue prioritization stage is the task weights since *cue specialization* model constitutes an important part of the proposed algorithm. In our system, we have followed the task classification by Ware [53]. According to this classification, the tasks are as follows:

- **Judging the relative positions of objects in space:** In a 3D scene, one fundamental task is to judge the relative distances of the objects. When the scene is complex, it is difficult to understand the positions of the objects relative to each other. In the near vicinity, fine depth judgements can be made using the binocular disparity cue [53]. For the far distances, binocular disparity becomes ineffective, however.

- **Reaching for objects:** In interactive applications, it is important to facilitate the user to reach the objects using the input device, when necessary. Especially for the Virtual Reality (VR) applications, it is difficult to adjust the eye-hand coordination without mismatch between the feedback from the scene and the proprioceptive sense of body position [53].

- **Judging the morphology of surfaces and surface target detection:**

In computer generated scenes, reflecting the actual shapes of the objects and the surface details is a challenging issue. For example, the shading model dramatically affects the shape perception. Shape-from-shading, structure-from-motion, and texture cues are also widely investigated in the literature, to be used in revealing the surface shapes. Thus, the use of visual cues properly is an important issue for visualizing the shapes and surfaces of the objects.

- **Tracing data paths in 3D graphs:** A graph can be considered as an information visualization method using networks of nodes and arcs. The nodes in the graph are the entities or components of the data collection, and the arcs represent the relationship between these components. For example, tree is a widely-used graph structure to visualize hierarchical data. Various forms of 3D graphs are employed in the field of Information Visualization, like cone trees. As the tree or graph becomes more complex, tracing the paths in the graph gets harder. Presenting more data in a 3D graph, without distracting the understandability requires efficient usage of 3D space and visual cues.

- **Finding patterns of points in 3D space:** In 2D, a common information visualization method is using scatter plots. Clouds of data points can be considered as the application of 2D scatter plots in 3D. However, in 3D space, it becomes complicated to interpret the positions of the points in the plot.

- **Judging the "up" direction:** In real life, the gravity and the ground help us to determine the directions, however, an artificial environment generally lacks such kind of cues to perceive the directions. Hence, in computer generated images, it is important to provide visual cues that help judging the directions.

- **The aesthetic impression of 3D space (Presence):** An important problem with the virtual environments is the difficulty in providing a sense of presence. In order to make the user feel that he is actually present in the virtual environment, the aesthetic impression of the scene is important.

For each task listed above, a fuzzy linguistic variable whose value is between 0 and 1 is kept. These values correspond to the weights of the tasks in the application and initially assigned by the application developer using any heuristics he desires. Fuzzification of the task related input variables are obtained by piecewise linear membership functions which divide the region into three as "low_priority", "medium_priority", and "high_priority" (Table 3.1).

Table 3.1: Linguistic variables and terms used for task input variables.

Linguistic Variable	**Linguistic Terms**
judging_relative_positions	{low_priority, medium_priority, high_priority }
reaching_for_objects	{low_priority, medium_priority, high_priority }
surface_target_detection	{low_priority, medium_priority, high_priority }
tracing_data_paths_in_3d_graph	{low_priority, medium_priority, high_priority }
patterns_of_points_in_3d	{low_priority, medium_priority, high_priority }
judging_up_direction	{low_priority, medium_priority, high_priority }
aesthetic_impression	{low_priority, medium_priority, high_priority }

As an example, the membership function of task "reaching_for_objects" is shown in the top left of Figure 3.3 on page 35. The membership functions for other tasks are also the same. Using these membership functions and the task weights, each task is labeled as "low_priority", "medium_priority", or "high_priority" to be used in the rule base according to Eq. 3.1.

$$\begin{aligned} \mu_{low\_priority}(x) &= x, \quad x \in [0, 0.2) \\ \mu_{medium\_priority}(x) &= x, \quad x \in [0.2, 0.6) \\ \mu_{high\_priority}(x) &= x, \quad x \in [0.6, 1) \end{aligned} \tag{3.1}$$

where x is the crisp input value which corresponds to the weight of the task and $\mu_{low\_priority}$, $\mu_{medium\_priority}$, and $\mu_{high\_priority}$ are the functions for low_priority, medium_priority, and high_priority respectively.

Distance Range

As stated previously, *range extension* is another cue combination model that constructs our hybrid model. For this part, we consider the distance of the objects to the user as an input to the system. In order to represent the distance range of the objects, two input linguistic variables "minDistance" and "maxDistance" are defined. These values are calculated as the minimum and maximum distances between the scene elements and the viewpoint, and mapped to the range 0-100. In order to fuzzify the input variables for distance, we use the trapezoidal membership functions, which are constructed based on the distance range classification in [10], shown in the top right of Figure 3.3 on page 35. Based on these membership functions, the degree of the membership of the input variables "minDistance" and "maxDistance" are determined for each linguistic term as "close, "near", "middle", or "far" (Table 3.2, Eq. 3.2).

Table 3.2: Linguistic variables and terms used for distance input variables.

Linguistic Variable	**Linguistic Terms**
minDistance	{ close, near, medium, far }
maxDistance	{ close, near, medium, far }

$$
\begin{aligned}
\mu_{close}(x) &= -x/2+1, \quad x \in [0,2) \\
\mu_{near}(x) &= \begin{cases} x/2, & x \in [0,2) \\ 1, & x \in [2,10) \\ -x/2+6, & x \in [10,12] \end{cases} \\
\mu_{medium}(x) &= \begin{cases} x/2-4, & x \in [8,10) \\ 1, & x \in [10,50) \\ -x/50+2, & x \in [50,100] \end{cases} \\
\mu_{far}(x) &= \begin{cases} x/50-1, & x \in [50,100) \\ 1, & x \in [100,\infty) \end{cases}
\end{aligned}
\tag{3.2}
$$

where x is the crisp input value which corresponds to the absolute distance from the viewer and μ_{close}, μ_{near}, μ_{medium}, and μ_{far} are the functions for close, near, medium, and far respectively.

Scene Suitability

The spatial layout of the scene may affect the behaviors of different depth cues in different ways. For instance, it is known that the human visual system has a prior that there is a single, stationary light source located left-above, while interpreting the shapes of the objects [17]. Hence, if there are more than one light sources or the direction of the light is not from above, shading and shadow cues may not produce the expected effects in the depth perception. As another example case, if there are a large number of points in a 3D scatter plot, cast shadows do not contribute to the depth perception [53]. In addition, relative height cue is effective on the depth perception given that the object sizes are not too different from each other and the horizon line is not too far from the center of the picture [17].

Table 3.3: Linguistic variables and terms used for scene input variables.

Linguistic Variable	**Linguistic Terms**
$scene_c$	{ poor, fair, suitable }

In order to handle such kind of scene specific situations in our system, we define another input linguistic variable "scene" whose value is between 0 and 1, for each depth cue (Table 3.3). Initially, the value of "scene" is 1 which means that the scene is assumed to be suitable for each depth cue. To determine the suitability value for each cue, a scene analysis step is performed. In this step, the scene is analyzed separately for each depth cue and if there is an inhibitive situation similar to the cases described above, the "scene" value for that cue is penalized. If there are multiple constraints for the same depth cue, the minimum of the calculated values is accepted as the "scene" value.

In the current implementation, several scene characteristics are used as follows:

First of all, according to Madison et al., cast shadows give the best results when the objects are slightly above the ground plane [53]. To handle this scene layout related property, we count the number of objects which are slightly above the ground plane, in the scene. While deciding whether an object is slightly above the ground plane, we use Eq. 3.3. Then, we calculate the $scene_{shadow}$ as the ratio of the number of objects that are slightly above the ground plane (*numOfObjectsSlightlyAbove*) between the total number of objects (*totalNumOfObjects*) in the scene (Eq. 3.4).

$$isSlightlyAbove(o) = \begin{cases} yes, & o_y - ground_y \leq roomHeight/3 \\ no, & otherwise. \end{cases} \tag{3.3}$$

where *isSlightlyAbove(o)* is the function that determines if the given object o is slightly above the ground plane or not, o_y is the y-coordinate of object o, $ground_y$ is the y-coordinate of the ground plane, and *roomHeight* is the height of the room.

$$scene_{shadow} = \frac{numOfObjectsSlightlyAbove}{totalNumOfObjects} \tag{3.4}$$

Another scene characteristic to be analyzed for shadow depth cue is based on the assumption that the number of objects in the scene should not be high, when the task is *patterns_of_points_in_3d* [53]. For this purpose, we use Eq. 3.5 to calculate the value of $scene_{shadow}$.

$$scene_{shadow} = min(1 - \frac{totalNumOfObjects}{threshold},\ 0) \tag{3.5}$$

where *totalNumOfObjects* is the total number of objects in the scene and *threshold* is the threshold value after which we assume that the number of objects are high. As the *threshold* value, we use 200.

Lastly, since we use bump-mapping method to provide texture gradient cue in our framework, we check whether there is a suitable normal map for the given texture map and use this information to obtain the $scene_{texture\_gradient}$ value.

The "scene" values which are calculated at the scene analysis module are fuzzified as "poor", "fair", or "suitable" using the piecewise linear membership function (Eq. 3.6) shown in the bottom left of Figure 3.3 on page 35.

$$\begin{aligned} \mu_{poor}(x) &= x, \quad x \in [0, 0.2) \\ \mu_{fair}(x) &= x, \quad x \in [0.2, 0.6) \\ \mu_{suitable}(x) &= x, \quad x \in [0.6, 1) \end{aligned} \tag{3.6}$$

where x is the crisp input value which corresponds to the scene suitability value for the given depth cue and μ_{poor}, μ_{fair}, and $\mu_{suitable}$ are the functions for poor, fair, and suitable respectively.

3.2.2 Inference

The inference engine of the fuzzy logic system maps the fuzzy input values to fuzzy output values using a set of IF-THEN rules. In this work, our rule base is constructed based on a literature survey of experimental studies on depth perception. For each depth cue, there is a different set of rules. According to the values of the fuzzified input variables, the rules are evaluated using the fuzzy operators shown in Table 3.4. *Maximum* accumulation method is used for combining the results of the individual rules. (See Appendix A for other possible accumulation methods.)

Table 3.4: Fuzzy logic operators used in the evaluation of the rules.

Operator	**Operation**	**Fuzzy Correspondance**
AND	μ_A(x) & μ_B(x)	min{ μ_A(x), μ_B(x) }
OR	$\mu_A(x) \parallel \mu_B(x)$	max{ $\mu_A(x)$, $\mu_B(x)$ }
NOT	$\neg\mu_A(x)$	$1 - \mu_A(x)$

Table 3.5 contains sample rules used to evaluate the priority value of the shadow depth cue. The complete set of rules can be found in Appendix B. For the fuzzy logic related part of the system; *jFuzzyLogic* library [22], which is a Java implementation of the FCL(Fuzzy Control Language) [1], is used.

Table 3.5: Sample fuzzy rules for shadow depth cue.

	Shadow Rules
1.	IF scene is *poor* THEN shadow is *unsuitable*
2.	IF scene is *fair* THEN shadow is *fair*
3.	IF scene is *suitable* AND asthetic_impression is *low_priority* THEN shadow is *strong*
4.	IF scene is *suitable* AND asthetic_impression is *medium_priority* THEN shadow is *fair*
5.	IF scene is *suitable* AND asthetic_impression is *high_priority* THEN shadow is *weak*
6.	IF scene is *suitable* AND patterns_of_points_in_3d is *low_priority* THEN shadow is *strong*
7.	IF scene is *suitable* AND patterns_of_points_in_3d is *medium_priority* THEN shadow is *fair*
8.	IF scene is *suitable* AND patterns_of_points_in_3d is *high_priority* THEN shadow is *weak*
9.	IF scene is *suitable* AND surface_target_detection is *low_priority* THEN shadow is *strong*
10.	IF scene is *suitable* AND surface_target_detection is *medium_priority* THEN shadow is *fair*
11.	IF scene is *suitable* AND surface_target_detection is *high_priority* THEN shadow is *weak*

Figure 3.4 visualizes the rules for shadow which are listed in Table 3.5. For

the pictorial depth cues, the distance range is not checked since they are effective for all distances [10]. If the scene layout is suitable for the given cue, task-related variables are controlled and according to the task weights, the priority value of the shadow is determined.

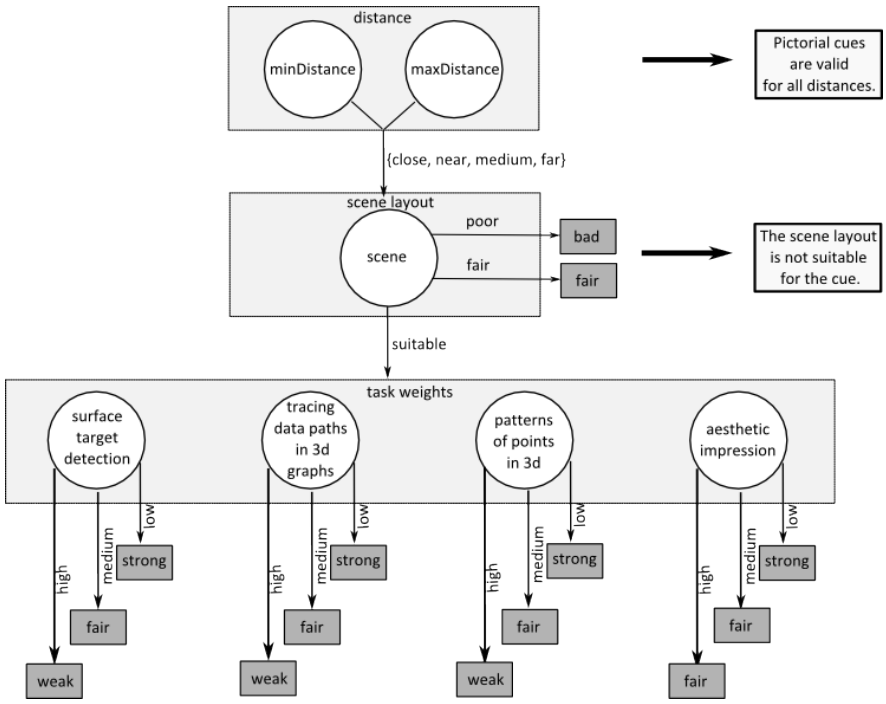

Figure 3.4: A decision tree-like structure for the shadow related rules in Table 3.5.

3.2.3 Defuzzification

The inference engine produces fuzzy output variables with values “strong”, “fair”, “weak”, or “unsuitable” for each depth cue (Table 3.6). These fuzzy values should be converted to non-fuzzy correspondences to be used in the subsequent stages of the system. The purpose of the defuzzification step is to perform this fuzzy to non-fuzzy mapping which is obtained by the triangular and trapezoidal membership functions (Eq. 3.7) shown in the bottom right of Figure 3.3 on page 35. The membership functions are the same for each depth cue.

Table 3.6: Linguistic variables and terms used for cue output variables.

Linguistic Variable	**Linguistic Terms**
size_gradient	{ unsuitable, weak, fair, strong }
relative_height	{ unsuitable, weak, fair, strong }
relative_brightness	{ unsuitable, weak, fair, strong }
texture_gradient	{ unsuitable, weak, fair, strong }
shading	{ unsuitable, weak, fair, strong }
shadow	{ unsuitable, weak, fair, strong }
aerial_perspective	{ unsuitable, weak, fair, strong }
linear_perspective	{ unsuitable, weak, fair, strong }
depth_of_focus	{ unsuitable, weak, fair, strong }
accommodation	{ unsuitable, weak, fair, strong }
convergence	{ unsuitable, weak, fair, strong }
binocular_disparity	{ unsuitable, weak, fair, strong }
motion_parallax	{ unsuitable, weak, fair, strong }
motion_perspective	{ unsuitable, weak, fair, strong }
kinetic_depth	{ unsuitable, weak, fair, strong }

$$
\begin{aligned}
f_0(x) &= \begin{cases} 1, & x \in [0, 0.2) \\ -5x + 2, & x \in [0.2, 0.4) \end{cases} \\
f_1(x) &= \begin{cases} 5x - 1, & x \in [0.2, 0.4) \\ -5x + 3, & x \in [0.4, 0.6) \end{cases} \\
f_2(x) &= \begin{cases} 5x - 2, & x \in [0.4, 0.6) \\ -5x + 4, & x \in [0.6, 0.8) \end{cases} \\
f_3(x) &= \begin{cases} 5x - 3, & x \in [0.6, 0.8) \\ 1, & x \in [0.8, 1] \end{cases}
\end{aligned}
\tag{3.7}
$$

where x is the fuzzy output value which corresponds to the result of the evaluation of the rules for the given cue and f_0, f_1, f_2, and f_3 are the functions for unsuitable,

weak, fair, and strong respectively.

As the Defuzzification algorithm, we use the "center of gravity" (COG) algorithm in Eq. 3.8. (See Appendix A for other defuzzification algorithms.)

$$U = \frac{\int_{min}^{max} u\,\mu(u)\,du}{\int_{min}^{max} \mu(u)\,du} \tag{3.8}$$

where U is the result of defuzzification, u is the output variable, μ is the membership function after accumulation, min is the lower limit for defuzzification, and max is the upper limit for defuzzification [1].

Figure 3.5 shows the result of a sample run of the system for the shadow depth cue. In the figure, shaded regions belong to the fuzzy output of the system. This result is defuzzified using the COG algorithm and the final priority value for the shadow cue is calculated as the center of gravity of this region.

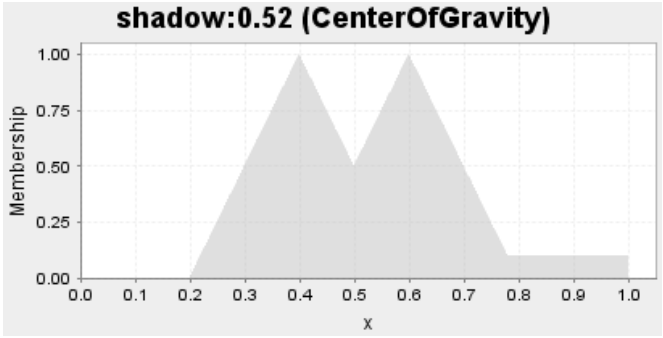

Figure 3.5: A sample output of the system for shadow depth cue.

In Figure 3.6, a sample demonstration of the overall fuzzy cue prioritization stage for the shadow depth cue is illustrated. The process operates left to right: First, the input variables are converted to fuzzy variables and used in the evaluation of the rules. Then, the results of the individual rules are combined and a fuzzy output variable is obtained. Lastly, the fuzzy output value is mapped to a crisp value which is the priority of that depth cue. The same process is performed for each depth cue. At the end of this stage, priority values for each depth cue, between 0 and 1, are produced to be used in the method selection stage.

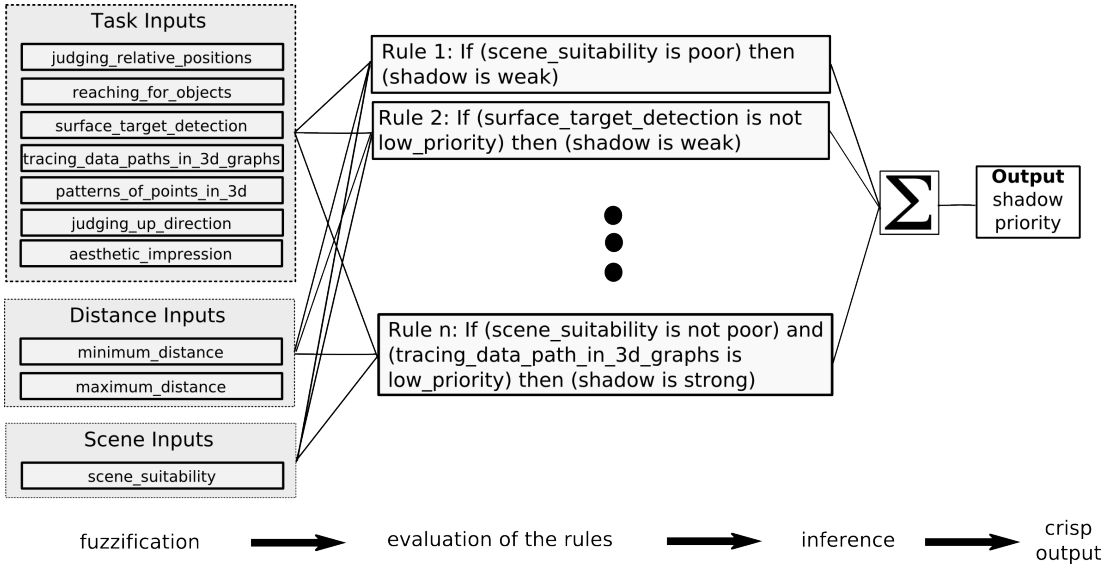

Figure 3.6: Demonstration of the fuzzy cue prioritization stage on shadow depth cue.

3.3 Method Selection

After calculating the priority of each depth cue, the next stage is to provide the cues with high priority using proper rendering methods. However, there are different depth enhancement methods that provide the same cue, as well as methods that can provide multiple cues at the same time. Hence, how to select the rendering methods that improve the 3D perception is another major part of the problem. The purpose of the method selection stage of this algorithm is to select proper rendering methods that provide the depth cues selected in the previous stage.

Table 3.7 shows the depth cues and corresponding rendering methods used to provide these depth cues in our system. In the table, some of the methods are labeled as "helper". This means that these rendering methods do not provide the corresponding depth cue directly, however either they increase the effect of the method or there is a dependency between the rendering methods that provide this cue. For instance, *perspective projection* does not provide *texture gradient* if

the surface is not textured; nevertheless, when the surface is textured, *perspective projection* increases the effect of *texture gradient* cue. As another example, *ground plane* does not make sense alone, yet it is obligatory for shadows on the ground to be visble.

Table 3.7: Rendering methods corresponding to the depth cues.

Depth Cues	**Depth Enhancement Methods**
Size gradient	Perspective projection
Relative height	Perspective projection, dropping lines to ground
	Helper: ground plane
Relative brightness	Proximity luminance, fog
Aerial perspective	Fog, proximity luminance, Gooch shading
Texture gradient	Texture mapping, bump mapping
	Helper: perspective projection, ground plane, room
Shading	Gooch shading, boundary enhancement,
	texture mapping
Cast shadow	Shadow map
	Helper: ground plane, room
Linear perspective	Perspective projection
	Helper: ground plane, room, texture mapping
Depth of focus	Depth-of-field, multi-view rendering
Accommodation	Multi-view rendering
Convergence	Multi-view rendering
Binocular disparity	Multi-view rendering
Motion parallax	Face tracking, multi-view rendering
Motion perspective	Mouse/keyboard controlled motion

The architecture that carries out the mapping from the depth cue to rendering method is demonstrated in Figure 3.7. The inputs to the system are the cue priority vector from the "Cue Prioritization" stage, current frame rate in frames per second (FPS) from the application, and a target FPS from the user. The FPS values are used to calculate the maximum cost which corresponds to the maximum allowable reduction in the current FPS (Eq. 3.9), since the cost of a rendering method is an important limitation while deciding on the proper rendering method.

$$maxCost = currentFPS - targetFPS \tag{3.9}$$

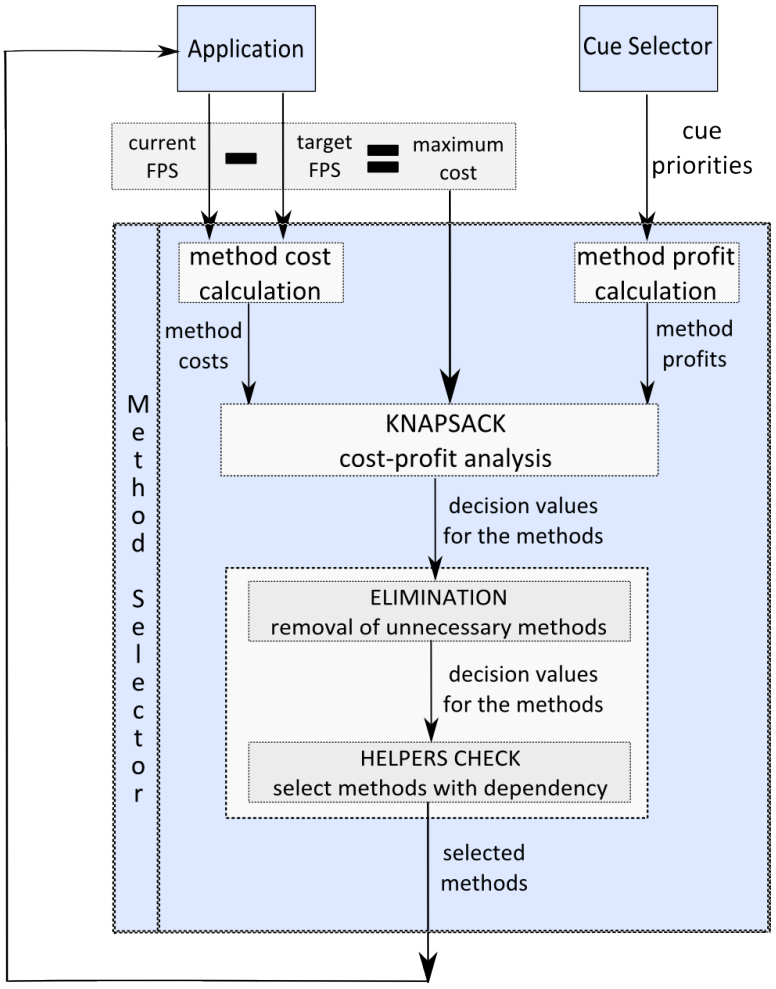

Figure 3.7: Method Selection Stage.

The core part of this stage is modeling the trade-off between the cost and the profit of a depth enhancement method as a *Knapsack* problem. According to this approach, a "profit" and a "cost" value are assigned to each depth enhancement method. "Profit" is used to quantify how much a method contributes to the enhancement of depth perception in the given scene and calculated as a weighted sum of the priorities of the depth cues that are provided by this method (Eq. 3.10), based on the "cue averaging" model described in Section 2.1.2.

$$profit_i = \sum_{j \in C_i} c_j \times p_j \tag{3.10}$$

where C_i is the set of all depth cues that the method i provides, p_j is the priority value of the cue j, and c_j is a constant that represents how much the method i provides the cue j.

In our system, we calculate the "cost" of a rendering method as the reduction in the current FPS, caused by this method. For this purpose, we defined a cost reduction table which keeps a FPS reduction value (R_i) in percentages for each method. These values are obtained empirically, as the average reduction in FPS due to the corresponding method, through different scenes. Then, the cost of each rendering method is calculated using this table and the current FPS, at run time (Eq. 3.11).

$$cost_i = \frac{R_i}{100} \times currentFPS \tag{3.11}$$

Using the "cost", "profit", and "maxCost" calculated as above, the optimal solution for Eq. 3.12 is computed which maximizes the total "gain" while keeping the total cost under the "maxCost".

$$\begin{aligned} Gain &= \sum_{i \in M} profit_i \times x_i \\ Cost &= (\sum_{i \in M} cost_i \times x_i) \leq maxCost \end{aligned} \tag{3.12}$$

where M is the set of all depth enhancement methods, *maxCost* limits the total cost, $cost_i$ is the cost of the method i, $x_i \in \{0, 1\}$ is the solution for method i. Note that, 0 value of the solution x_i means that "do not apply the method i", while 1 means "apply".

In order to solve this Knapsack problem, we use the dynamic programming approach [24]. According to the recursion in Eq. 3.13; if $c_i > C$, then item i is

too big for our knapsack. On the other hand, item i fits in our knapsack and there are two options: The first option is not taking item i and keeping the previous solution $z_{i-1}(C)$ same. The second option is taking the i^{th} item into knapsack which increases the profit by p_i and decreases the remaining capacity of the knapsack to C - c_i. The maximum of these two choices produces the optimal solution.

$$z_i(C) = \begin{cases} z_{i-1}(C), & c_i > C \\ max\{z_{i-1}(C),\ p_i + z_{i-1}(C - c_i)\}, & c_i \leq C \end{cases} \tag{3.13}$$

where $z_i(C)$ is the optimal solution for selecting items from 1 to i with maximum cost C, c_i and p_i are the cost and profit of the i^{th} item, respectively.

At the end of the cost-profit analysis step, we obtain the decision values of each depth enhancement method. It is possible to use these values directly as the final decisions; however, we apply two more steps to improve the quality of the system.

The elimination step can be considered as a post-processing on the selected methods. The purpose of this step is to eliminate additional cost by unselecting some of the methods that provide only the cues that are already provided by other methods. For example, although the main purpose of *multi-view rendering* is providing *binocular disparity*, it also creates the *depth-of-focus* effect. Hence, there is no need to spend additional cost for the *depth-of-field* method which only provides *depth-of-focus* cue, if a more "profitable" method, *multi-view rendering*, is already selected. (For such kind of methods, see Table 3.7.)

Another post-processing step is checking the use of helpers, in which the methods that are labeled as "helper" for the corresponding method in Table 3.7 are checked and selected if they are not already selected. The costs and profits of these methods are also added to the total cost and profit, respectively. For instance, if *shadow mapping* method is selected but *ground plane* is not enabled, this step selects *ground plane* and updates the total cost and profit accordingly.

The overall procedure in the method selection stage is summarized in Algorithm 1. As stated in the algorithm, the above procedure is repeated multiple times to obtain a more accurate estimation in the final FPS. This multiple pass scheme is used because most of the time, although the estimated FPS value reaches the target FPS, the actual FPS is above the target; since the FPS reduction amount of each method are the average values rather than exact numbers for the given scene. Hence, after applying the selected methods at the end of each pass, the current FPS and the maximum cost are recalculated to check whether there is still room for the costs of other methods. The number of passes is bounded by a *threshold* value. Three passes generally result in accurate estimations.

Algorithm 1 The algorithm of the method selection stage.

```
1. passNo = 0
2. get targetFPS from the user
3. while currentFPS > targetFPS and passNo < threshold
      3.1 calculate maximumCost as currentFPS - targetFPS
      3.2 calculate method profits as the weighted linear combination
                        of the cue priorities that they provide
      3.3 calculate method costs as the reduction in currentFPS
      3.4 apply knapsack analysis on non-selected methods
      3.5 unselect the methods with the same purpose
      3.6 select the methods that other mehods depend on
      3.7 apply selected methods
      3.8 increment passNo
```

Finally, at the end of the method selection stage, we have decided on which rendering methods will be applied to the given scene by considering the costs of the methods. Note that, other limitations can also be taken into account, such as memory requirements according to the platform we use. It is also possible to extend the system to consider multiple limitations at the same time, using multiply constrained knapsack problem.

3.4 Methods for Enhancing Depth Perception

As shown in Figure 3.1 on page 32, after receiving the decision values of the

rendering methods, the last step is to apply these methods to the given scene, to obtain a perceptually refined scene. In this section, important rendering methods used in our system will be explained.

Shadow Map

Cast shadows provide strong perceptual cues for depth in many tasks. In our framework, shadow is obtained by using shadow maps. The idea behind the shadow map is straightforward. The scene is rendered from the position of the light source and the depth values in the z-buffer are known as shadow map or shadow buffer [2] as shown in Figure 3.8. In other words, a depth test is performed from the light's point of view and the points that cannot pass the depth test should be in shadow. The scene is re-rendered from the original viewpoint and combined with the shadow map values to generate shadows.

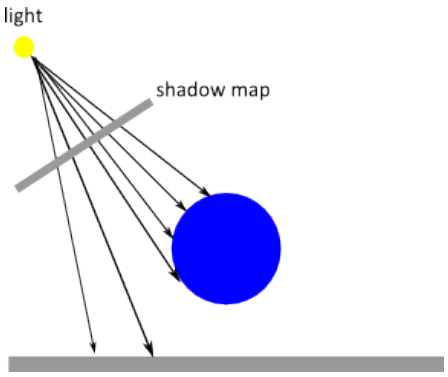

Figure 3.8: Illustration of the shadow map.

Figure 3.9: Shadow map method. (Left: original scene, Right: the scene with shadows)

Fog

Fog is a widely-used atmospheric effect which is generally employed for increasing the realism of the scene. In addition, it provides strong perceptual cues to depth since its effect increases with the distance. We use fog rendering method to provide aerial perspective cue. For rendering fog, we use the following equations:

$$c_{final} = f \times c_{surface} + (1 - f) \times c_{fog} \tag{3.14}$$

$$f = \begin{cases} \frac{z_{end} - z_p}{z_{end} - z_{start}}, & linear \\ e^{-d_f \times z_p}, & exponential \end{cases} \tag{3.15}$$

Here, the final color of each pixel (c_{final}) is interpolated between the surface color ($c_{surface}$) and the fog color (c_{fog}) according to the fog factor (f). The fog factor depends on the distance from the viewpoint and calculated using Eq. 3.15. Assume that z_p is the depth value of the current pixel. If we want a linear distribution of the fog, we calculate the fog factor (f) using the z_{start} and z_{end} values which are the depth values where fog starts and ends, respectively. On the other hand, if we want an exponential fall-off in the fog factor, we use the exponentiation in the equation in which the density of the fog is controlled by the parameter d_f. The parameters that are used in these equations are shown in Table 3.8 and the effect of fog method is shown in Figure 3.10.

Table 3.8: Parameters used in fog equations.

parameter	**value**
c_{fog}	(0.4, 0.4, 0.4)
z_{start}	$roomDepth/4$
z_{end}	$3 \times roomDepth/2$
d_f	0.03

Figure 3.10: Effect of the fog rendering method. (Left: original scene, Middle: the scene with linear fog, Right: the scene with exponential fog)

Gooch Shading

In this method, we apply Gooch shading, which is a non-photorealistic lighting model, also known as cool-to-warm shading. In this lighting model, cool colors are used to represent regions that are far from the light source and warm colors are used for the opposite. In other words, the color of the object is interpolated between cool colors and warm colors according to the light direction. Eq. 3.16 is used to calculate the illumination [13]. It is a straightforward interpolation equation between a cool color and a warm color.

$$I = \frac{(1 + \vec{n} \odot \vec{l})}{2} \times kCool + (1 - \frac{(1 + \vec{n} \odot \vec{l})}{2}) \times kWarm \tag{3.16}$$

where I is the illumination at that point, $\vec{n}$ is the surface normal vector, and $\vec{l}$ is the normalized light direction vector. Note that, since the dot product of $\vec{n}$ and $\vec{l}$ varies between -1 and 1, the result is clamped to [0,1] range. The resulting values form Eq. 3.17 are substituted in Eq. 3.16.

$$\begin{aligned} kCool &= kBlue + \alpha \times kd \\ kWarm &= kYellow + \beta \times kd \end{aligned} \tag{3.17}$$

In Eq. 3.17, cool color (*kCool*) of the object is calculated as a blueish color (*kBlue*) plus a constant (α) times of diffuse reflectance term (*kd*). Warm color of the object is calculated in a similar manner using a yellowish color (*kYellow*).

In order to improve the perception, we also add specular reflections using Eq. 3.18 to compute the specular term of the illumination.

$$ks = (\vec{r} \odot \vec{v})^{\lambda} \tag{3.18}$$

where *ks* is the specular term, $\vec{r}$ is the normalized reflection vector of light directon with respect to the surface normal, $\vec{v}$ is the normalized view vector, and λ is a parameter to determine the power. Table 3.9 contains the parameters used in our system.

Table 3.9: Parameters used in Gooch shading.

parameter	**value**
kBlue	(0, 0, 0.6)
kYellow	(0.6, 0.6, 0)
α	0.2
β	0.6
λ	32

Figure 3.11 compares the original scene to the Gooch shaded scene. As the results show, the surface details of the objects are more obvious in the Gooch shaded scene. It also adds an atmospheric effect to the scene.

Figure 3.11: Effect of the Gooch shading method. (Left: original scene, Right: Gooch shaded scene.)

Proximity Luminance

Proximity luminance is a method that changes the luminance of the objects according to their distance from the viewpoint. In other words, the closer the object to the viewpoint, the higher the contrast between the object and the background color. By using this method, *relative brightness* and *aerial perspective* depth cues can be added to the scene.

In this framework, we obtain the proximity luminance by modifying the luminance of each object by a function of its distance from the viewpoint. Firstly, in the vertex shader, we calculate the distance of the object to the user by transforming the vertex position to the view space and taking the length of this transformed vertex. In the pixel shader, we retrieve the vertex distance from the vertex shader and use this value to change the luminance of that pixel. For this purpose, we first convert the color value from RGB space to HSL space, modify the luminance value according to Eq. 3.19, and convert the modified color back to RGB space.

$$\forall p \in P, \quad L'_p = \lambda \times eyeDist_p^2 \times L_p \tag{3.19}$$

where P is the set of all pixels in the image, L_p is the current luminance value of pixel p, L'_p is the modified luminance value of pixel p, $eyeDist_p$ is the distance of pixel p to the viewpoint, and λ is a constant that determines the strength of the

method.

Even though it is not crucial, we also apply the same procedure to the saturation value to obtain a more appealing result, as Weiskopf and Ertl suggest [56]. Eq. 3.20 is used for this purpose, this time S_p and S'_p are the original and modified saturation values of pixel p, respectively.

$$\forall p \in P, \quad S'_p = \lambda \times eyeDist_p^2 \times S_p \tag{3.20}$$

The effect of the proximity luminance method is demonstrated in Figure 3.12.

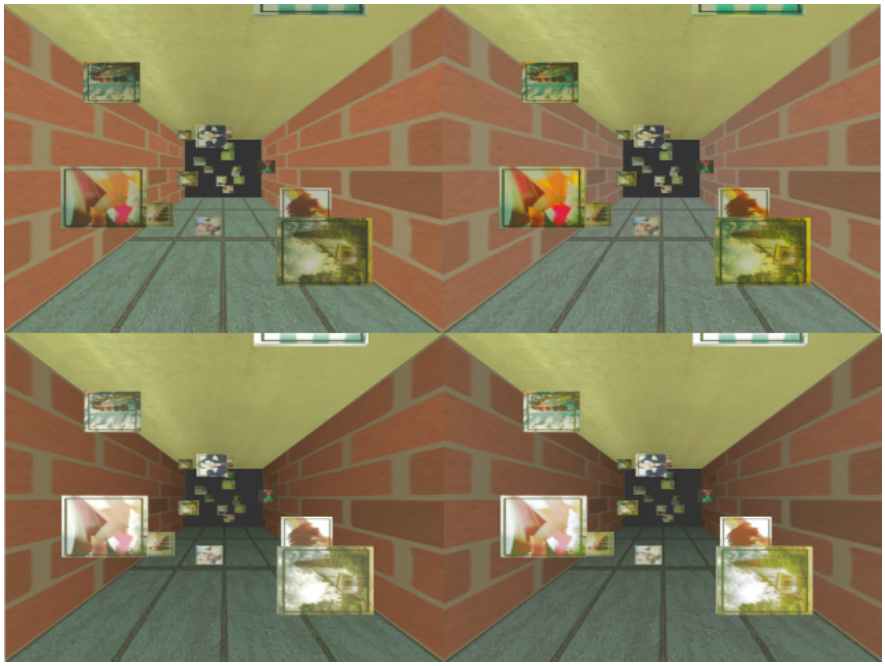

Figure 3.12: Effect of the proximity luminance method. (Top-left: original scene, top-right: only saturation is changed, bottom-left: only luminance is changed, bottom-right: both saturation and luminance are changed with the distance.)

Boundary Enhancement

So as to improve the perception of the shape of the objects, we enhance the important edges by increasing the color contrast on these areas, using the depth

buffer-based method proposed by Luft et al. [29]. According to this method, the depth information retrieved from the depth buffer is written to a texture to be processed in the pixel shader. The depth texture is read in the pixel shader and the derivative of the depth values is used to calculate the "spatial importance" function which indicates the spatially important areas in the scene. This function is calculated as the difference between the original and the Gaussian filtered depth buffer (Eq. 3.21).

$$\Delta D = G * D - D \tag{3.21}$$

where D is the original depth buffer, G is a Gaussian filter kernel, and ΔD is the spatial importance function. Here, the $*$ operator stands for convolution. For the Gaussian kernel size, we also use the percentage of the image diagonal (Eq. 3.22) as suggested in the work of Luft et al. [29]. 2% of the diagonal length generally suffices as the kernel size.

$$kernelSize = \frac{x}{100} \times \sqrt{W^2 + H^2} \tag{3.22}$$

After calculating the spatial importance function, the color contrast of the whole image is modified by adding the spatial importance value multiplied by a constant to each color channel (Eq. 3.23).

$$\forall p \in P, \quad \begin{aligned} R_p &= R_p + \Delta D_p \cdot \lambda \\ G_p &= G_p + \Delta D_p \cdot \lambda \\ B_p &= B_p + \Delta D_p \cdot \lambda \end{aligned} \tag{3.23}$$

where P is the set of all pixels, ΔD_p is the spatial importance value of pixel p, λ is a constant determining the strength of the modification, and R_p, G_p, and B_p are the red, green, and blue values of that pixel respectively.

Figure 3.13: Effect of the boundary enhancement method. (Left to right: original scene, spatial impotance function, contrast enhanced scene.)

Effect of the edge enhancement method is shown in Figure 3.13, along with the spatial importance map of the scene.

Face Tracking

In this framework, we use a face tracking algorithm which we previously developed for user interaction on mobile devices [6]. This is a color-based algorithm which employs the skin color properties of a human face to extract the facial regions. The algorithm generally operates on HSL color space since it decouples the lightness value from the color and heavily used in face localization, detection, and recognition problems [6]. The overall system architecture of the face tracking system is shown in Figure 3.14.

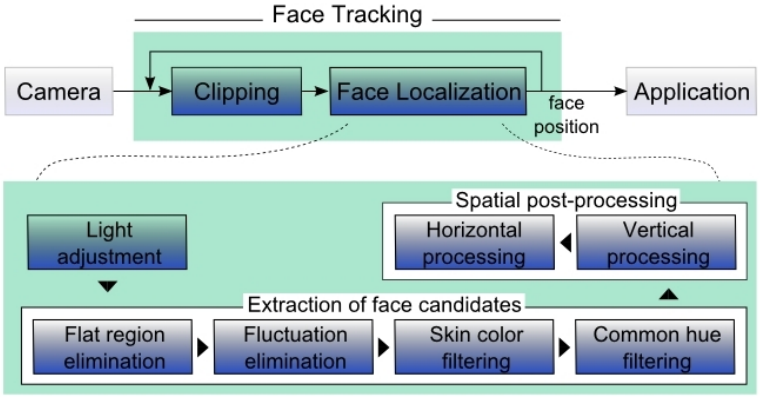

Figure 3.14: Face tracking: system architecture.

The algorithm does not operate on the whole image as it is costly. Instead,

it selects a suitable region of the image and applies the face localization part on the clipped region. The clipping phase of the algorithm selects a horizontal and a vertical slice of the image according to the previous y and x positions of the face, respectively. In addition, the previous z position, which indicates the closeness to the screen, determines the length of the portion of the scanline.

After selecting the vertical and horizontal scanlines in the clipping phase, the face localization algorithm is applied on each scanline. At the beginning, a light adjustment step, which sets the average light value of the image to a central value by widening the light distribution, is performed; since the lightness value is highly varying in a mobile environment. However, this step is optional for our system as it operates on desktop systems.

While extracting the face candidates, several steps are applied which eliminate the pixels that cannot be a face, considering the color properties of a skin. First of all, since the human face has a curvature, flat regions in the image cannot be facial regions. The flat regions are identified by searching for the lightness values not changing for a long sequence of pixels. In the next step, rapidly fluctuating regions on the image are eliminated since the color values of a face generally vary monotonically. Then, a skin color filter is applied which employs the red-green ratio of the skin color and eliminates the regions that cannot belong to a face. Lastly, the assumption that face covers a large region in the image is used to cluster the hue values in the image and select the cluster with the maximum number of pixels as the face hue. Note that this face tracking algorithm is originally designed to handle varying background conditions due to a mobile environment. However, if we ensure that the background is not changing, we can eliminate some of the steps in the face candidates extraction part to improve the performance. For instance, if the background is a flat wall, there is no need for eliminating the fluctuating regions.

After extracting the face candidates, the next step is to find the actual face among these candidate pixels. The purpose of the spatial post-processing phase is to analyze these pixels according to their spatial properties and localize the actual face using the coherency of a face shape.

As the result of the previous steps, some pixels are eliminated in the original image. The 3D face position is calculated by averaging the non-eliminated pixels positions for x and y separately. The z value is determined by the ratio of the number of non-eliminated pixels to the number of eliminated pixels.

We use this face tracking system in order to provide motion parallax effect in our architecture. The face tracker runs in a separate thread and the face position obtained from the face tracker is used in the application to determine the camera properties. Face position can be used in two different ways: In the first one, the face position is used as analogous to the viewpoint as shown in Figure 3.15. In the second one, face position is not directly used to determine the viewpoint; instead, it is used as helper to control the viewpoint. This helper functionality can be considered as the head movements of a user rather than the movement of the entire body. This way of interaction can be useful especially for controlling the head movements of a character in a first person shooter game which is demonstrated in Figure 3.16.

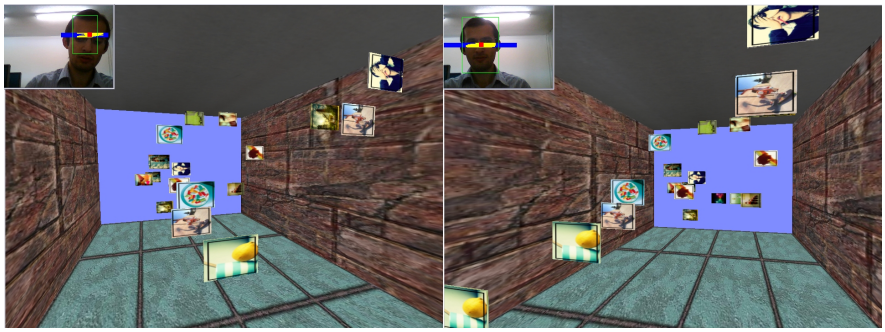

Figure 3.15: Demonstration of the face tracker in a camera application. (Left: The user looks at the scene from left. Right: The user looks at the scene from right.)

Figure 3.16: Demonstration of the face tracker in a game application. The face tracker is integrated into a sample game from Xith3D toolkit [18]. (Left: The user looks at the scene from left. Right: The user looks at the scene from right to see around the corner.)

Multi-view Rendering

In our system, to provide binocular disparity along with convergence and accommodation cues, we use multi-view rendering. During the multi-view rendering process, the original scene is rendered from several viewpoints according to the number of views and these separate images are combined depending on the multi-view display technology which is known as interlacing process.

We employ 9-view lenticular display produced by i-Art Corporation [21] for multi-view. Interlacing functions for n-view lenticular display are below.

$$\forall p \in P, \quad \begin{aligned} index_{R_p} &= (8 - (pov_{R_p} - 1) + 2) \bmod n_{view} \\ index_{G_p} &= (8 - (pov_{G_p} - 1) + 2) \bmod n_{view} \\ index_{B_p} &= (8 - (pov_{B_p} - 1) + 2) \bmod n_{view} \end{aligned} \tag{3.24}$$

For each pixel p in the image; indices of the red (R_p), green (G_p), and blue (B_p) values are calculated using Eq. 3.24. These indices indicate from which view the color value will be gathered. The *pov* values in this equation are substituted from the results of Eq. 3.25.

$$
\begin{aligned}
a &= 3 \times y_p \times slant \\
pov_{R_p} &= \begin{cases} ((x_p + k_{off} - a) \bmod sp) \times \frac{n_{view}}{sp} + 1, & x_p + k_{off} > a \\ (sp - ((-x_p - k_{off} + a) \bmod sp)) \times \frac{n_{view}}{sp} + 1, & otherwise \end{cases} \\
pov_{G_p} &= \begin{cases} ((x_p + 1 + k_{off} - a) \bmod sp) \times \frac{n_{view}}{sp} + 1, & x_p + 1 + k_{off} > a \\ (sp - ((-x_p - 1 - k_{off} + a) \bmod sp)) \times \frac{n_{view}}{sp} + 1, & otherwise \end{cases} \\
pov_{B_p} &= \begin{cases} ((x_p + 2 + k_{off} - a) \bmod sp) \times \frac{n_{view}}{sp} + 1, & x_p + 2 + k_{off} > a \\ (sp - ((-x_p - 2 - k_{off} + a) \bmod sp)) \times \frac{n_{view}}{sp} + 1, & otherwise \end{cases}
\end{aligned}
\tag{3.25}
$$

where x_p and y_p are the x and y coordinates of pixel p, n_{view} is the number of views, sp is the subpixels per lens horizontally, *slant* is the slant of the lenticular sheet. The values of these parameters are shown in Table 3.10.

Table 3.10: Parameters used in interlacing equations.

parameter	**value**
n_{view}	9
sp	4.5
$slant$	5/3
k_{off}	-3

In our system, the scenes rendered from different viewpoints are written to textures and sent to the pixel shader. Then, the interlacing operation is performed in the pixel shader using Eq. 3.24 and 3.25. To avoid the high rendering cost, we only generate four different views and use the same view for two adjacent views. The process is demonstrated in Figure 3.17.

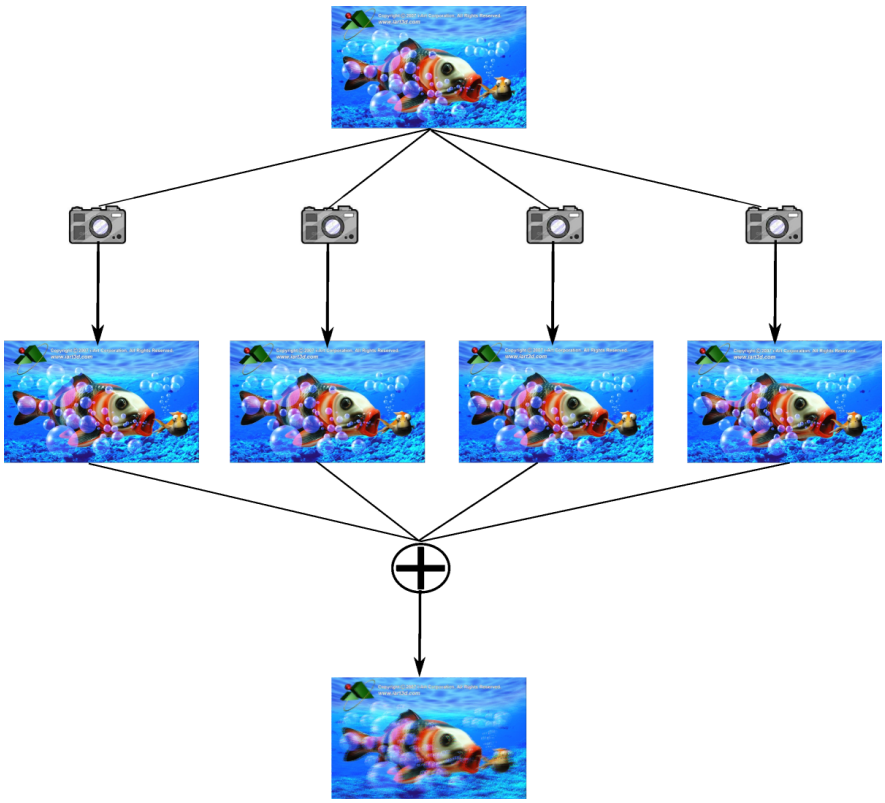

Figure 3.17: Multi-view rendering process. (The images are taken from the sample images of i-Art Auto3D Viewer tool.)

Chapter 4

Experiments and Evaluations

In order to evaluate the success of the proposed depth perception enhancement system, we have performed a number of objective and subjective experimental studies. In this chapter, we discuss these experimental studies and their results in detail.

In this study, we selected two important and common tasks among the ones described in Section 3.2.1: "judging relative positions" and "surface target detection (shape perception)". For the task "judging relative positions", we performed both a subjective and an objective experiment. On the other hand, we tested the "shape perception" task based on a subjective study. The following sections present detailed information about these objective and subjective experiments.

4.1 Objective Experiment

The purpose of this experiment is to test the effect of our system on depth perception, for the task "judging relative positions", compared to different conditions. The details of this experiment are explained in the following subsections.

Subjects

The objective experiment was performed on 14 subjects: 9 males and 5 females with a mean age of 24.4. All the subjects have self-reported normal or corrected vision. They were voluntary graduate and undergraduate students with computer science background. The purpose of the experiment was not explained to the subjects.

Procedure

In this experiment, an experimental setup similar to the one in Wanger's study [51] is used. Subjects were given a scene with a randomly positioned test object in a region whose boundaries were indicated visually and asked to estimate the z position of the given object (Figure 4.1). The subjects were informed that the minimum and the maximum z values were 0 and 50, respectively. There was no time limit and when they were ready, they entered their estimations using a slider-like widget (Figure 4.1).

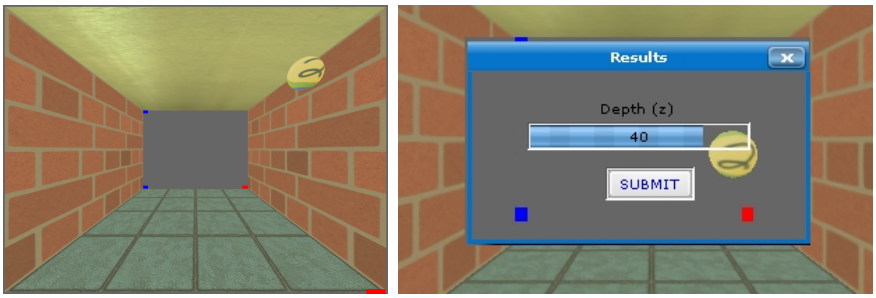

Figure 4.1: Left: The scene used in the objective experiment. (Red bars show the boundaries in z and blue bars show the boundaries in y.) Right: Submission of the results.

The above procedure was repeated for five different conditions. In the first case, there weren't any depth cues in the scene other than the perspective projection, which was indicated by the lengths of the limit indicator bars shown in Figure 4.1. In the second test case, some of the depth enhancement methods were chosen in a fully random fashion, at run time. The third case was also a random selection, but this time there was a cost limit. In other words, each method was decided to be applied randomly only if it did not decrease the frame rate under the given cost limit. The cost limit was the same as the one used in automatic

selection case. The fourth case was the application of all the depth enhancement methods and in the last case, the methods that will be applied to the scene were chosen using our automatic method selection algorithm.

Results and Discussion

According to our automatic method selection framework; linear perspective, relative brightness, texture gradient, shading, shadow, and relative height cues have the highest priorities for the test scene and task. To provide these cues, the method selection stage selected the methods shadow, room, keyboard control, Gooch shading, proximity luminance, bump mapping, reference objects, and dropping lines to the ground.

In order to measure the success of our system, we first calculated the RMS errors for each of the five test cases using Eq. 4.1.

$$RMS(T) = \sqrt{\frac{\Sigma_{i=1}^{|T|}(T_i - R_i)^2}{|T|}} \tag{4.1}$$

where T is the set of subjects' answers and R is the set of real positions.

Figure 4.2 shows the RMS errors for each test case. As the figure indicates, automatic method selection algorithm gives the best results with RMS error of only 3.1%. Hence, our algorithm is even better than applying all the methods. This indicates that, the disadvantage of applying all the methods is not only the high cost; it is also prone to more errors on depth judgments. A possible reason for this result is that applying all the methods may cause cue conflicts and confuse the subjects. Also, when all the methods are applied, the frame rate falls below 10 FPS and this situation distracts the user. Therefore, we consider the third case which is cost limited random selection as the strongest competitor of our case because of the cost limit. The results show that our method selection algorithm results in more than 2 times better estimations in depth, compared to the third case.

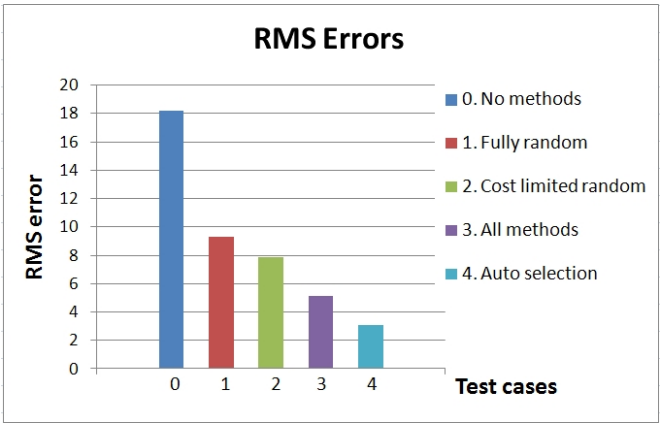

Figure 4.2: RMS errors for the objective depth judgement test.

We have also performed a reliability analysis on the experimental data, similar to the one in the study by Oruc et al. [40]. For each test case, reliability values were calculated using Eq. 4.2.

$$r_i = {\sigma_i}^{-2} \tag{4.2}$$

where r_i is the reliability of test case i and σ_i is the variance of the experimental results for test case i.

As shown in Figure 4.3, the behavior of the reliability graph is the inverse of the RMS error graph as expected; and the highest reliability value is for our case. Note that since the values in the experimental data set are between 0 -100, the variances of these values are high which result in low reliability values. However, the important thing in this graph is the relative change in the reliability values for each test case.

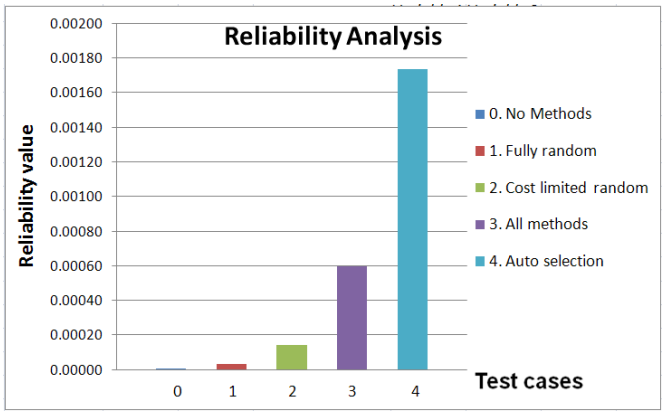

Figure 4.3: Results of the reliability analysis.

RMS errors and the reliability values show that our automatic method selection algorithm works better than all the other method selection techniques used in our test cases. In order to better indicate the statistically significant difference of our method, we have also performed a *paired samples t-test* on the experimental data. The mean error of each test case were compared to the mean error of our algorithm and it is shown that the difference between the automatic method selection algorithm and the other selection techniques is statistically significant with $p < 0.05$ (Table 4.1).

Table 4.1: Results of the t-test for objective depth judgement experiment.

test case	**mean difference**	**p value**
Auto selection (4) - no methods (0)	25	0.002
Auto selection (4) - fully random (1)	9.6	0.03
Auto selection (4) - cost limited random (2)	9	0.01
Auto selection (4) - all methods (3)	4.1	0.01

4.2 Subjective Experiments

We have also performed subjective experiments for the tasks “judging relative

positions" and "shape perception". In these experiments, the subjects were asked to evaluate the given scenes, subjectively.

Subjects

For the depth judgement task, 21 subjects (17 males, 4 females) with a mean age of 24.2 participated in the experiment. On the other hand, 11 subjects (9 males, 2 females) whose mean age is 24.1, evaluated the scenes for the shape judgement task. The subjects were among the voluntary graduate and undergraduate students who have self-reported normal or corrected vision. They were not informed about the purpose of the experiments.

Procedure

For each task, the subjects were shown a scene and asked to grade the given scene between 0 and 100. For the depth judgement task, the left scene, in which objects of the same size are randomly positioned in the space, in Figure 4.4 is shown to the subjects. Whereas, the right scene in Figure 4.4 is shown for the shape judgement task, since it contains more complex and detailed meshes which are more suitable for shape judgement.

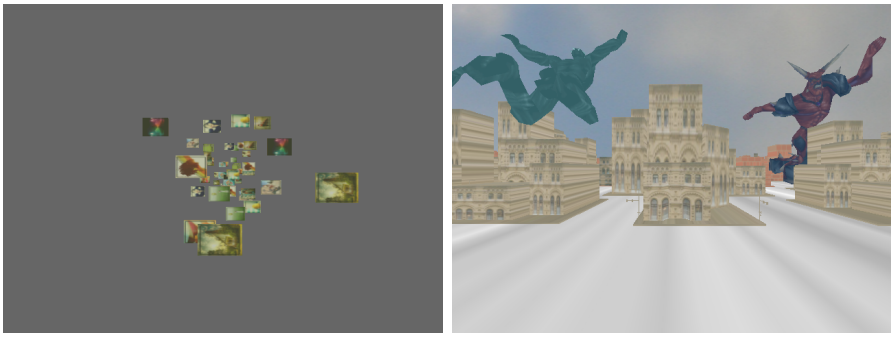

Figure 4.4: Left: The scene (without cues) for depth judgement task. Right: The scene (without cues) for shape judgement task.

At the beginning of the experiment, the subjects were informed about the grading criteria as follows:

- "For the depth judgement, you should consider how easy it is to understand the relative distances between the objects in the scene."
- "For the shape judgement task, you should evaluate the perception of the morphology of the objects. Is it easy to distinguish the boundaries and creases of the objects?"

At first, the original scene without any cues was shown to the subjects and they were told that the grade of this scene is 50 and they graded the other scenes by comparing them to the original scene. Test cases were the same with the objective experiment. However; this time the first case, the scene without cues, is not graded by the subjects since it is shown to the subjects as reference with a grade of 50. The subjects submitted their grades using the forms shown in Figure 4.5.

Figure 4.5: Submission of the results.

Results and Discussion

For the "depth judgement" test, our system calculated the relative brightness, linear perspective, depth-of-focus, motion parallax, and binocular disparity cues as the highest priority cues. In this respect, the methods keyboard control, room, multi-view rendering, and proximity luminance were suggested as suitable rendering methods for the given scene by the system. On the other hand, for the test "shape judgement", the highest priority cues were aerial perspective, linear perspective, shading, texture gradient, and motion parallax. The method selection stage suggested keyboard control, boundary enhancement, face tracking, bump

mapping, Gooch shading, shadow, and proximity luminance methods to provide these cues.

Figure 4.6 shows the average grades of each test case, for the depth judgement task. According to the results, the scene rendered with our automatic method selection algorithm has an average grade of 86.8. "All methods" and "fully random" cases follow the automatic selection case with a grade of about 77, although they have no cost limitation. The error bars in the graph show the 95% confidence interval of the mean, which corresponds to the range within which the mean is expected to fall with 95% certainty. Non-overlapping error bars indicate the statistical difference with $p < 0.05$. The error bar of the "automatic selection" case overlaps the error bars of "all methods" and "fully random" cases. Hence, we have performed a *paired samples t-test* to show the statistical difference between these cases and obtained a statistically significant difference with $p < 0.05$ for both cases (Table 4.2).

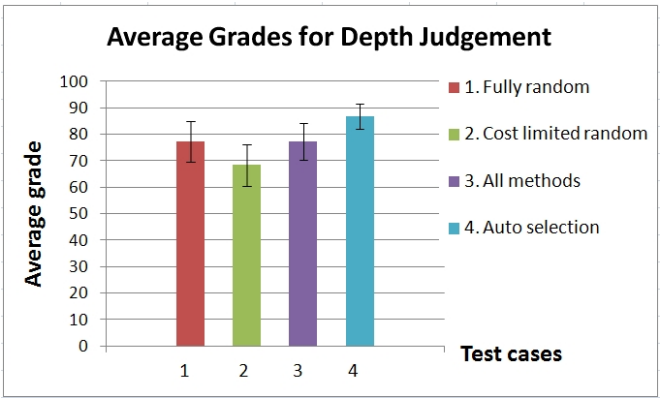

Figure 4.6: Experimental results for subjective depth judgement. (Error bars show the 95% confidence intervals.)

Table 4.2: Results of the t-test for subjective depth judgement experiment.

test case	**mean difference**	**p value**
Auto selection (4) - fully random (1)	9.5	0.01
Auto selection (4) - cost limited random (2)	18.38	0.00002
Auto selection (4) - all methods (3)	9.5	0.03

The average grades for the shape judgement task is shown in Figure 4.7. Our automatic method selection algorithm has the highest grade (72.7) for this task too. The error bar of the "auto selection" case overlaps with the "cost limited random" and "all methods" cases, this time. We have also performed a *paired samples t-test* for the results of the shape judgement task and showed that our algorithm is statistically ($p < 0.05$) better than the other selection techniques (Table 4.3).

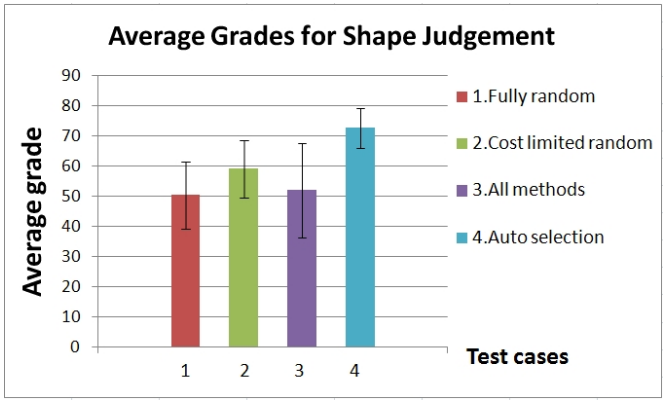

Figure 4.7: Experimental results for subjective shape judgement. (Error bars show the 95% confidence intervals.)

Table 4.3: Results of the t-test for subjective shape judgement experiment.

test case	**mean difference**	**p value**
Auto selection (4) - fully random (1)	22.3	0.006
Auto selection (4) - cost limited random (2)	13.54	0.03
Auto selection (4) - all methods (3)	20.6	0.04

4.3 General Discussion

We have performed both objective and subjective experiments for depth judgement task and only a subjective experiment for shape judgement task. Experimental results were analyzed statistically, using paired samples t-test. These

statistical evaluations indicate that our automatic depth enhancement algorithm works better than the other method selection techniques tested in our experiments. For instance, in objective experiment for depth judgement, subjects estimated the z position of the test object with RMS error of only 3.1%, in average. Also, the scene rendered with the proposed depth enhancement algorithm received a subjective grade of 87 out of 100 points in average, which is statistically better than the grades of the scenes rendered with other selection techniques. Besides, in terms of performance gains, applying all the methods results in very low frame rates (could be lower than 10 FPS) whereas, more effective results can be obtained with minimum 15-20 FPS using our framework.

Note that these results may slightly differ for different experimental conditions. For instance, adding new depth enhancement methods or changing the implementation of the existing methods may affect the results. Besides, only two tasks were experimented and the other tasks should also be tested. Likewise, different method selection techniques other than the ones used in the experiments may also be compared to the automatic selection algorithm. However, these different conditions would not affect the results drastically and current experimental results still give a strong idea about the success of the proposed algorithm.

Figure 4.8 shows several examples of the results of our automatic depth enhancement system. In the figure, the original scenes are shown on the left side and their depth enhanced versions are shown on the right side. However, in this figure some depth enhancement methods such as multi-view rendering and face tracking cannot be visualized. These methods also have significant effects on the depth perception.

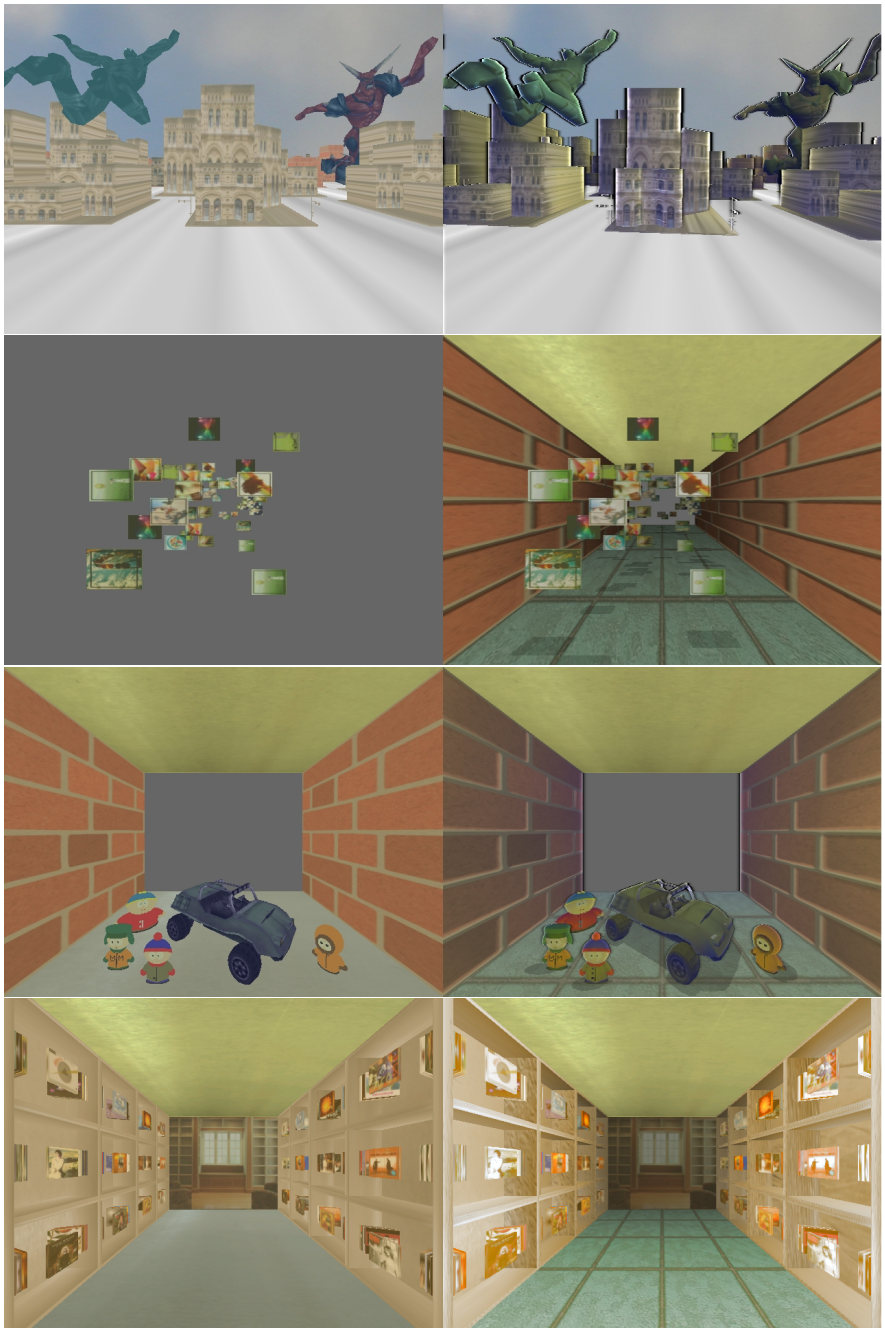

Figure 4.8: Left: Original scene. Right: The scene with enhanced depth perception.

Chapter 5

Conclusion

In this work, we have proposed a framework that enhances the depth perception in a given 3D graphical content. For this purpose, we have developed an algorithm that automatically decides on the important depth cues using fuzzy logic and selects the rendering methods which provide these cues based on the Knapsack problem. In this automatic depth perception enhancement framework, we consider several factors: the user's tasks in the application, spatial layout of the scene, and the costs of the rendering methods.

We tested our system by the help of objective and subjective experimental studies. According to the results of the objective experiment for depth judgement, average RMS error of our system is only 3.1% (Figure 4.2) and our system works statistically better than the other selection methods used in our tests. In addition, we have conducted a subjective user experiment to evaluate our system for shape judgement. In this experiment, the average grade for the scene enhanced using our algorithm is about 73 out of 100 which is the best score among other test cases (Figure 4.7). We have analyzed the results using paired samples t-test and observed the statistically significant ($p < 0.05$) difference between our selection algorithm and the other selection ways.

One possible future direction for our system is to perform more comprehensive experiments such as testing other tasks and comparing with different selection

methods. Moreover, the rule base should be extended and more rendering methods should be implemented. Another idea for the future work is to compare different multi-view technologies and analyze the behavior of our system under these technologies. Furthermore, the current system is designed to operate globally which means that it does not analyze the objects in the scene individually. Hence, it can be extended to consider each object in the scene individually and apply some of the methods to only the objects that need these methods. Lastly, the effects of animation should also be taken into account.

Bibliography

[1] Fuzzy control programming. Technical report, International Electrotechnical Commision, 1997.

[2] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-Time Rendering*, chapter 6. A. K. Peters, third edition, 2008.

[3] M. Brackstone. Examination of the use of fuzzy sets to describe relative speed perception. *Ergonomics*, 43(4):528–542, 2000.

[4] M. F. Bradshaw, A. D. Parton, and A. Glennerster. The task-dependent use of binocular disparity and motion parallax information. *Vision Research*, 40(27):3725 – 3734, 2000.

[5] S. Bruckner and E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351, 2007.

[6] A. Bulbul, Z. Cipiloglu, and T. Capin. A color-based face tracking algorithm for enhancing interaction with mobile devices. *The Visual Computer*, 26(5):311–323, 2010.

[7] H. H. Bulthoff and A. Yuille. *A Bayesian Framework for the Integration of Visual Modules*. 1996.

[8] M. Bunnel. *Dynamic Ambient Occlusion and Indirect Lighting*. Addison Wesley, 2004.

[9] J. J. Clark and A. L. Yuille. *Data Fusion for Sensory Information Processing Systems*. Springer, 1990.

[10] J. E. Cutting and P. M. Vishton. *Perception of Space and Motion. Handbook of Perception and Cognition*, pages 69–117. Academic Press, second edition, 1995.

[11] N. A. Dodgson. Autostereoscopic 3d displays. *Computer*, 38:31–36, 2005.

[12] I. Fine and R. A. Jacobs. Comparing perceptual learning across tasks: A review. *Journal of Vision*, pages 190–203, 2002.

[13] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452, New York, NY, USA, 1998. ACM.

[14] B. Gooch, P. J. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 31–38, New York, NY, USA, 1999. ACM.

[15] P. Haeberli and K. Akeley. The accumulation buffer: Hardware support for high-quality rendering. *SIGGRAPH Comput. Graph.*, 24(4):309–318, 1990.

[16] M. Halle. Autostereoscopic displays and computer graphics. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 104, New York, NY, USA, 2005. ACM.

[17] I. Howard and B. Rogers. *Seeing in Depth.* Oxford University Press, 2008.

[18] http://xith.org/. Home of the xith3d project, 2009.

[19] G. S. Hubona, P. N. Wheeler, G. W. Shirah, and M. Brandt. The relative contributions of stereo, lighting, and background scenes in promoting 3d depth visualization. *ACM Trans. Comput.-Hum. Interact.*, 6(3):214–242, 1999.

[20] J. Huttenlocher and L. Hedges. Combining graded categories: Membership and typicality. *Psychological Review*, 101:157–157, 1994.

[21] i Art Corporation. http://www.iart3d.com/, 2010.

[22] jFuzzyLogic. http://jfuzzylogic.sourceforge.net/html/index.html, 2010.

[23] T. Kaneko, T. Takahei, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda, and S. Tachi. Detailed shape representation with parallax mapping. In *Int Conf Artif Real Telexistence*, 2001.

[24] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[25] D. Kersten, D. C. Knill, P. Mamassian, and I. Bulthoff. Illusory motion from shadows. *Nature*, 1996.

[26] D. Kersten, P. Mamassian, and D. C. Knill. Moving cast shadows and the perception of relative depth. Technical report, Max Planck Institute for Biological Cybernetics, 1994.

[27] M. Kraus and M. Strengert. Depth-of-field rendering by pyramidal image processing. *Computer Graphics Forum*, 26:645–654, 2007.

[28] M. S. Landy, L. T. Maloney, E. B. Johnston, and M. J. Young. Measurement and modeling of depth cue combination: in defense of weak fusion. *Vision Research*, pages 389–412, 1995.

[29] T. Luft, C. Colditz, and O. Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph.*, 25(3):1206–1213, 2006.

[30] L. T. Maloney and M. S. Landy. A statistical framework for robust fusion of depth information. In *SPIE Visual Commmunications and Image Processing IV*, 1989.

[31] P. Mamassian, D. C. Knill, and D. Kersten. The perception of cast shadows. *Trends in Cognitive Sciences*, 2(8):288 – 295, 1998.

[32] D. Mao. Face tracking algorithms: A survey. Technical report, Panasonic Singapore Lab, 2009.

[33] L. Markosian, M. A. Kowalski, D. Goldstein, S. J. Trychin, J. F. Hughes, and L. D. Bourdev. Real-time nonphotorealistic rendering. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and*

interactive techniques, pages 415–420, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[34] G. Mather and D. R. R. Smith. Depth cue integration: Stereopsis and image blur. *Vision Research*, 40(25):3501 – 3506, 2000.

[35] M. McGuire and J. F. Hughes. Hardware-determined feature edges. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 35–47, New York, NY, USA, 2004. ACM.

[36] J. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377, Mar 1995.

[37] M. Nienhaus and J. Doellner. Edge-enhancement - an algorithm for real-time non-photorealistic rendering. *Journal of WSCG*, pages 346–353, 2003.

[38] M. M. Oliveira. *Relief Texture Mapping*. PhD thesis, University of North Carolina, 2000.

[39] M. M. Oliveira, G. Bishop, and D. McAllister. Relief texture mapping. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 359–368, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[40] I. Oruc, T. Maloney, and M. Landy. Weighted linear cue combination with possibly correlated error. *Vision Research*, 43:2451–2468, 2003.

[41] J. Pfautz. *Depth Perception in Computer Graphics*. PhD thesis, Cambridge University, 2000.

[42] P. Rheingans and D. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7:253–264, 2001.

[43] J. Russell. How shall an emotion be called. *Circumplex models of personality and emotions*, pages 205–220, 1997.

[44] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer*

graphics and interactive techniques, pages 197–206, New York, NY, USA, 1990. ACM.

[45] P. R. Schrater and D. Kersten. How optimal depth cue integration depends on the task. *International Journal of Computer Vision*, 40:73–91, 2000.

[46] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on GPUs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 73–80, New York, NY, USA, 2007. ACM.

[47] P. Shirley. *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., 2002.

[48] A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 47, Washington, DC, USA, 2003. IEEE Computer Society.

[49] C. T. Swain. Integration of monocular cues to improve depth perception, 2009.

[50] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):1237–1244, Sept.-Oct. 2006.

[51] L. Wanger. The effect of shadow quality on the perception of spatial relationships in computer generated imagery. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 39–42, New York, NY, USA, 1992. ACM.

[52] L. R. Wanger, J. A. Ferwerda, and D. A. Greenberg. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications*, 12:44–58, 1992.

[53] C. Ware. *Information Visualization: Perception for Design*, chapter 8. Elsevier, 2004.

[54] C. Ware, C. Gobrecht, and M. Paton. Dynamic adjustment of stereo display parameters. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(1):56–65, Jan 1998.

[55] C. Ware and P. Mitchell. Visualizing graphs in three dimensions. *ACM Trans. Appl. Percept.*, 5(1):1–15, 2008.

[56] D. Weiskopf and T. Ertl. A depth-cueing scheme based on linear transformations in tristimulus space. Technical report, Visualization and Interactive Systems Group University of Stuttgart, 2002.

[57] C. Zhang and R. Crawfis. Shadows and soft shadows with participating media using splatting. *IEEE Transactions on Visualization and Computer Graphics*, 9:139–149, 2003.

Appendix A

A Short Fuzzy Logic Tutorial

The purpose of this tutorial is to give a brief information about fuzzy logic systems. The tutorial is prepared based on the studies [36] and [1]. For further information on fuzzy logic, the reader is directed to these studies.

A fuzzy logic system (FLS) can be defined as the nonlinear mapping of an input data set to a scalar output data [36]. A FLS consists of four main parts: fuzzifier, rules, inference engine, and defuzzifier. These components and the general architecture of a FLS is shown in Figure A.1.

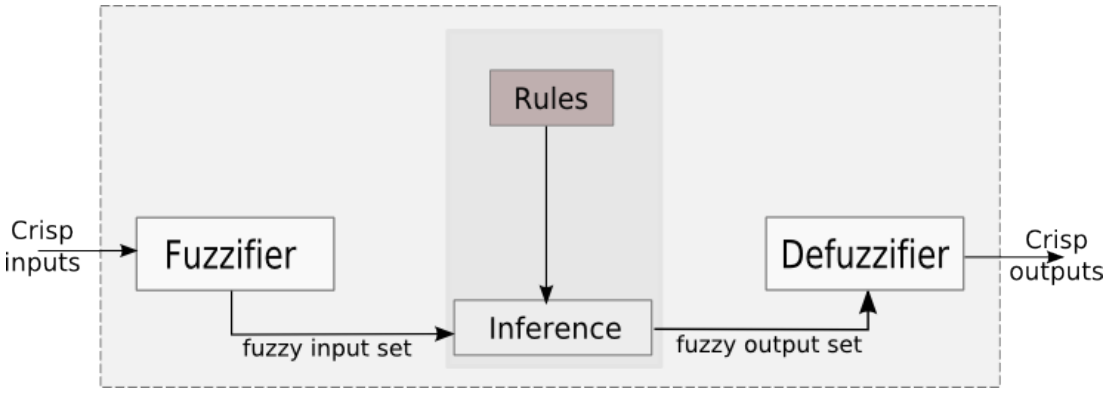

Figure A.1: A Fuzzy Logic System.

The process of fuzzy logic is explained in Algorithm 2: Firstly, a crisp set

of input data are gathered and converted to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions. This step is known as fuzzification. Afterwards, an inference is made based on a set of rules. Lastly, the resulting fuzzy output is mapped to a crisp output using the membership functions, in the defuzzification step.

Algorithm 2 Fuzzy logic algorithm

```
1. Define the linguistic variables and terms (initialization)
2. Construct the membership functions (initialization)
3. Construct the rule base (initialization)
4. Convert crisp input data to fuzzy values
   using the membership functions (fuzzification)
5. Evaluate the rules in the rule base (inference)
6. Combine the results of each rule (inference)
7. Convert the output data to non-fuzzy values (defuzzification)
```

In order to exemplify the usage of a FLS, consider an air conditioner system controlled by a FLS (Figure A.2). The system adjusts the temperature of the room according to the current temperature of the room and the target value. The fuzzy engine periodically compares the room temperature and the target temperature, and produces a command to heat or cool the room.

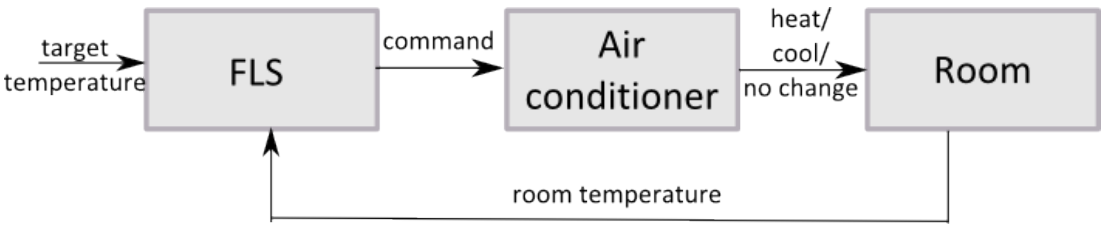

Figure A.2: A Simple FLS to Control an Air Conditioner.

Linguistic Variables

Linguistic variables are the input or output variables of the system whose values are words or sentences from a natural language, instead of numerical values. A linguistic variable is generally decomposed into a set of linguistic terms.

Example: Consider the air conditioner in Figure A.2. Let *temperature* (t) is the

linguistic variable which represents the temperature of a room. To qualify the temperature, terms such as "hot" and "cold" are used in real life. These are the linguistic values of the temperature. Then, *T(t) = {too-cold, cold, warm, hot, too-hot}* can be the set of decompositions for the linguistic variable temperature. Each member of this decomposition is called a linguistic term and can cover a portion of the overall values of the temperature.

Membership Functions

Membership functions are used in the fuzzification and defuzzification steps of a FLS, to map the non-fuzzy input values to fuzzy linguistic terms and vice versa. A membership function is used to quantify a linguistic term. For instance, in Figure A.3, membership functions for the linguistic terms of temperature variable are plotted. Note that an important characteristic of fuzzy logic is that a numerical value does not have to be fuzzified using only one membership function. In other words, a value can belong to multiple sets at the same time. For example, according to Figure A.3, a temperature value can be considered as "cold" and "too-cold" at the same time, with different degree of memberships.

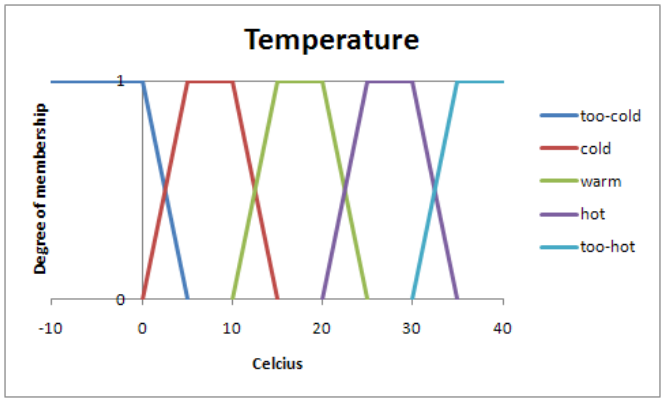

Figure A.3: Membership Functions for *T(temperature) = {too-cold, cold, warm, hot, too-hot}*.

There are different forms of membership functions such as triangular, trapezoidal, piecewise linear, Gaussian, or singleton (Figure A.4). The most common types of membership functions are triangular, trapezoidal, and Gaussian shapes.

The type of the membership function can be context dependent and it is generally chosen arbitrarily according to the user experience [36].

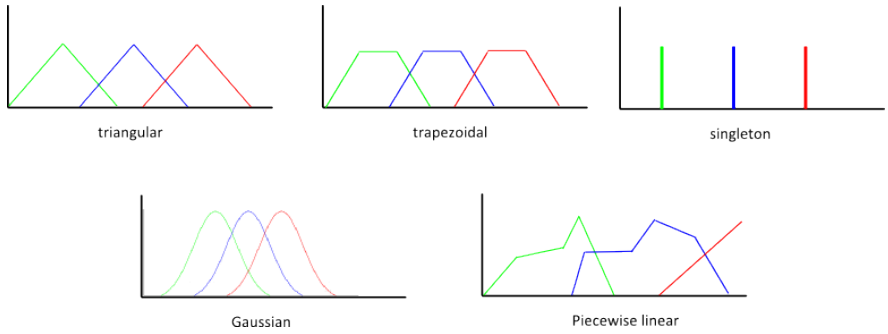

Figure A.4: Different Types of Membership Functions.

Fuzzy Rules

In a FLS, a rule base is constructed to control the output variable. A fuzzy rule is a simple IF-THEN rule with a condition and a conclusion. In Table A.1, sample fuzzy rules for the air conditioner system in Figure A.2 are listed. Table A.2 shows the matrix representation of the fuzzy rules for the said FLS. Row captions in the matrix contain the values that current room *temperature* can take, column captions contain the values for *target* temperature, and each cell is the resulting *command* when the input variables take the values in that row and column. For instance, the cell (3, 4) in the matrix can be read as follows: *If temperature is cold and target is warm then command is heat.*

Table A.1: Sample fuzzy rules for air conditioner system.

	Fuzzy Rules
1.	IF (temperature is *cold* OR *too-cold*) AND (target is *warm*) THEN command is *heat*
2.	IF (temperature is *hot* OR *too-hot*) AND (target is *warm*) THEN command is *cool*
3.	IF (temperature is *warm*) AND (target is *warm*) THEN command is *no-change*

Table A.2: Fuzzy matrix example.

temperature/target	**too-cold**	**cold**	**warm**	**hot**	**too-hot**
too-cold	no-change	heat	heat	heat	heat
cold	cool	no-change	heat	heat	heat
warm	cool	cool	no-change	heat	heat
hot	cool	cool	cool	no-change	heat
too-hot	cool	cool	cool	cool	no-change

Fuzzy Set Operations

The evaluations of the fuzzy rules and the combination of the results of the individual rules are performed using fuzzy set operations. The operations on fuzzy sets are different than the operations on non-fuzzy sets. Let μ_A and μ_B are the membership functions for fuzzy sets A and B. Table A.3 contains possible fuzzy operations for OR and AND operators on these sets, comparatively. The mostly-used operations for OR and AND operators are *max* and *min*, respectively. For complement (NOT) operation, Eq. A.1 is used for fuzzy sets.

Table A.3: Fuzzy set operations.

OR (Union)		**AND (intersection)**	
MAX	$Max\{\mu_A(x), \mu_B(x)\}$	MIN	$Min\{\mu_A(x), \mu_B(x)\}$
ASUM	$\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$	PROD	$\mu_A(x)\mu_B(x)$
BSUM	$Min\{1, \mu_A(x) + \mu_B(x)\}$	BDIF	$Max\{0, \mu_A(x) + \mu_B(x) - 1\}$

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{A.1}$$

After evaluating the result of each rule, these results should be combined to obtain a final result. This process is called inference. The results of individual rules can be combined in different ways. Table A.4 contains possible accumulation

methods that are used to combine the results of individual rules. The maximum algorithm is generally used for accumulation.

Table A.4: Accumulation methods.

Operation	**Formula**
Maximum	$Max\{\mu_A(x), \mu_B(x)\}$
Bounded sum	$Min\{1, \mu_A(x) + \mu_B(x)\}$
Normalized sum	$\frac{\mu_A(x)+\mu_B(x)}{Max\{1, Max\{\mu_A(x'), \mu_B(x')\}\}}$

Defuzzification

After the inference step, the overall result is a fuzzy value. This result should be defuzzified to obtain a final crisp output. This is the purpose of the defuzzifier component of a FLS. Defuzzification is performed according to the membership function of the output variable. For instance, assume that we have the result in Figure A.5 at the end of the inference. In this figure, the shaded areas all belong to the fuzzy result. The purpose is to obtain a crisp value, represented with a dot in the figure, from this fuzzy result.

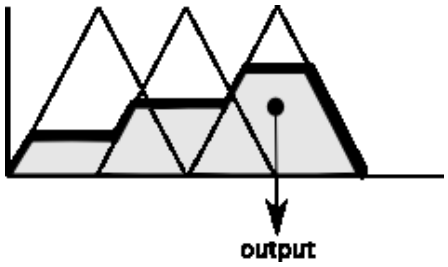

Figure A.5: Defuzzification step of a FLS.

There are different algorithms for defuzzification too. The mostly-used algorithms are listed in Table A.5. The meanings of the variables used in Table A.5 are explained in Teble A.6.

Table A.5: Defuzzification algorithms [1].

Operation	**Formula**
Center of Gravity	$U = \frac{\int_{min}^{max} u\,\mu(u)\,du}{\int_{min}^{max} \mu(u)\,du}$
Center of Gravity for Singletons	$\frac{\sum_{i=1}^{p} [u_i\ \mu_i]}{\sum_{i=1}^{p} [\mu_i]}$
Left Most Maximum	$U = inf(u')$, $\mu(u') = sup(\mu(u))$
Right Most Maximum	$U = sup(u')$, $\mu(u') = sup(\mu(u))$

Table A.6: The variables in Table A.5.

Variable	**Meaning**
U	result of defuzzification
u	output variable
p	number of singletons
μ	membership function after accumulation
i	index
min	lower limit for defuzzification
max	upper limit for defuzzification
sup	largest value
inf	smallest value

Appendix B

Fuzzy Rules

	Accommodation Rules
1.	IF scene is *poor* THEN accommodation is *unsuitable*
2.	IF scene is *fair* THEN accommodation is *fair*

	Aerial Perspective Rules
1.	IF scene is *poor* THEN aerial_perspective is *unsuitable*
2.	IF scene is *fair* THEN aerial_perspective is *fair*
3.	IF scene is *suitable* AND maxDistance is *far* THEN aerial_perspective is *strong*

	Binocular Disparity Rules
1.	IF scene is *poor* THEN binocular_disparity is *unsuitable*
2.	IF scene is *fair* THEN binocular_disparity is *fair*
3.	IF scene is *suitable* AND (minDistance is NOT *far* OR maxDistance is NOT *far*) AND asthetic_impression is *low_priority* THEN binocular_disparity is *strong*
4.	IF scene is *suitable* AND (minDistance is NOT *far* OR maxDistance is NOT *far*) AND asthetic_impression is *medium_priority* THEN binocular_disparity is *fair*
5.	IF scene is *suitable* AND (minDistance is NOT *far* OR maxDistance is NOT *far*) AND asthetic_impression is *high_priority* THEN binocular_disparity is *weak*
6.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND surface_target_detection is *low_priority* THEN binocular_disparity is *weak*
7.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND surface_target_detection is *medium_priority* THEN binocular_disparity is *fair*
8.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND surface_target_detection is *high_priority* THEN binocular_disparity is *strong*
9.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND patterns_of_points_in_3d is *low_priority* THEN binocular_disparity is *weak*
10.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND patterns_of_points_in_3d is *medium_priority* THEN binocular_disparity is *fair*
11.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND patterns_of_points_in_3d is *high_priority* THEN binocular_disparity is *strong*
12.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND judging_relative_positions is *low_priority* THEN binocular_disparity is *weak*
13.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND judging_relative_positions is *medium_priority* THEN binocular_disparity is *fair*
14.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND judging_relative_positions is *high_priority* THEN binocular_disparity is *strong*
15.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND reaching_for_objects is *low_priority)* THEN binocular_disparity is *weak*
16.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND reaching_for_objects is *medium_priority)* THEN binocular_disparity is *fair*
17.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close*) AND reaching_for_objects is *high_priority)* THEN binocular_disparity is *strong*
18.	IF scene is *suitable* AND (minDistance is NOT *far* OR maxDistance is NOT *far*) AND tracing_data_path_in_3d_graph is *low_priority* THEN binocular_disparity is *weak*
19.	IF scene is *suitable* AND (minDistance is NOT *far* OR maxDistance is NOT *far*) AND tracing_data_path_in_3d_graph is *medium_priority* THEN binocular_disparity is *fair*
20.	IF scene is *suitable* AND (minDistance is NOT *far* OR maxDistance is NOT *far*) AND tracing_data_path_in_3d_graph is *high_priority* THEN binocular_disparity is *strong*

Convergence Rules	
1.	IF scene is *poor* THEN convergence is *unsuitable*
2.	IF scene is *fair* THEN convergence is *fair*

Depth-of-focus Rules	
1.	IF scene is *poor* THEN dept_of_focus is *unsuitable*
2.	IF scene is *fair* THEN dept_of_focus is *fair*

Kinetic Depth Rules	
1.	IF scene is *poor* THEN kinetic_depth *unsuitable*
2.	IF scene is *fair* THEN kinetic_depth is *fair*
3.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *low_priority* THEN kinetic_depth is *weak*
4.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *medium_priority* THEN kinetic_depth is *fair*
5.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *high_priority* THEN kinetic_depth is *strong*
6.	IF scene is *suitable* AND patterns_of_points_in_3d is low_priority THEN kinetic_depth is *weak*
7.	IF scene is *suitable* AND patterns_of_points_in_3d is medium_priority THEN kinetic_depth is *fair*
8.	IF scene is *suitable* AND patterns_of_points_in_3d is high_priority THEN kinetic_depth is *strong*
9.	IF scene is *suitable* AND surface_target_detection is low_priority THEN kinetic_depth is *weak*
10.	IF scene is *suitable* AND surface_target_detection is medium_priority THEN kinetic_depth is *fair*
11.	IF scene is *suitable* AND surface_target_detection is high_priority THEN kinetic_depth is *strong*
12.	IF scene is *suitable* AND asthetic_impression is low_priority THEN kinetic_depth is *weak*
13.	IF scene is *suitable* AND asthetic_impression is medium_priority THEN kinetic_depth is *fair*
14.	IF scene is *suitable* AND asthetic_impression is high_priority THEN kinetic_depth is *strong*

	Linear Perspective Rules
1.	IF scene is *poor* THEN linear_perspective *unsuitable*
2.	IF scene is *fair* THEN linear_perspective is *fair*
3.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *low_priority* THEN linear_perspective is *strong*
4.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *medium_priority* THEN linear_perspective is *fair*
5.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *high_priority* THEN linear_perspective is *weak*
6.	IF scene is *suitable* AND patterns_of_points_in_3d is *low_priority* THEN linear_perspective is *strong*
7.	IF scene is *suitable* AND patterns_of_points_in_3d is *medium_priority* THEN linear_perspective is *fair*
8.	IF scene is *suitable* AND patterns_of_points_in_3d is *high_priority* THEN linear_perspective is *weak*

	Motion Parallax Rules
1.	IF scene is *poor* THEN motion_parallax *unsuitable*
2.	IF scene is *fair* THEN motion_parallax is *fair*
3.	IF scene is *suitable* AND minDistance is NOT *far* AND judging_relative_positions is *low_priority* THEN motion_parallax is *weak*
4.	IF scene is *suitable* AND minDistance is NOT *far* AND judging_relative_positions is *medium_priority* THEN motion_parallax is *fair*
5.	IF scene is *suitable* AND minDistance is NOT *far* AND judging_relative_positions is *high_priority* THEN motion_parallax is *strong*
6.	IF scene is *suitable* AND minDistance is NOT *far* AND reaching_for_objects is *low_priority* THEN motion_parallax is *weak*
7.	IF scene is *suitable* AND minDistance is NOT *far* AND reaching_for_objects is *medium_priority* THEN motion_parallax is *fair*
8.	IF scene is *suitable* AND minDistance is NOT *far* AND reaching_for_objects is *high_priority* THEN motion_parallax is *strong*
9.	IF scene is *suitable* AND minDistance is NOT *far* AND asthetic_impression is *low_priority* THEN motion_parallax is *weak*
10.	IF scene is *suitable* AND minDistance is NOT *far* AND asthetic_impression is *medium_priority* THEN motion_parallax is *fair*
11.	IF scene is *suitable* AND minDistance is NOT *far* AND asthetic_impression is *high_priority* THEN motion_parallax is *strong*

	Motion Perspective Rules
1.	IF scene is *poor* THEN motion_perspective *unsuitable*
2.	IF scene is *fair* THEN motion_perspective is *fair*
3.	IF scene is *suitable* AND aesthetic_impression is *low_priority* THEN motion_perspective is *weak*
4.	IF scene is *suitable* AND aesthetic_impression is *medium_priority* THEN motion_perspective is *fair*
5.	IF scene is *suitable* AND aesthetic_impression is *high_priority* THEN motion_perspective is *strong*

	Relative Brightness Rules
1.	IF scene is *poor* THEN relative_brightness is *unsuitable*
2.	IF scene is *fair* THEN relative_brightness is *fair*

	Relative Height Rules
1.	IF scene is *poor* THEN relative_height is *unsuitable*
2.	IF scene is *fair* THEN relative_height is *fair*

	Relative Size Rules
1.	IF scene is *poor* THEN relative_size is *unsuitable*
2.	IF scene is *fair* THEN relative_size is *fair*
3.	IF scene is *suitable* AND judging_relative_positions is *low_priority* THEN relative_size is *weak*
4.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close* OR minDistance is *near* OR maxDistance is *near*) AND judging_relative_positions is *medium_priority* THEN relative_size is *fair*
5.	IF scene is *suitable* AND (minDistance is *close* OR maxDistance is *close* OR minDistance is *near* OR maxDistance is *near*) AND judging_relative_positions is *high_priority* THEN relative_size is *strong*

	Shadow Rules
1.	IF scene is *poor* THEN shadow is *unsuitable*
2.	IF scene is *fair* THEN shadow is *fair*
3.	IF scene is *suitable* AND asthetic_impression is *low_priority* THEN shadow is *strong*
4.	IF scene is *suitable* AND asthetic_impression is *medium_priority* THEN shadow is *fair*
5.	IF scene is *suitable* AND asthetic_impression is *high_priority* THEN shadow is *weak*
6.	IF scene is *suitable* AND patterns_of_points_in_3d is *low_priority* THEN shadow is *strong*
7.	IF scene is *suitable* AND patterns_of_points_in_3d is *medium_priority* THEN shadow is *fair*
8.	IF scene is *suitable* AND patterns_of_points_in_3d is *high_priority* THEN shadow is *weak*
9.	IF scene is *suitable* AND surface_target_detection is *low_priority* THEN shadow is *strong*
10.	IF scene is *suitable* AND surface_target_detection is *medium_priority* THEN shadow is *fair*
11.	IF scene is *suitable* AND surface_target_detection is *high_priority* THEN shadow is *weak*
12.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *low_priority* THEN shadow is *strong*
13.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *medium_priority* THEN shadow is *fair*
14.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *high_priority* THEN shadow is *weak*

Shading Rules	
1.	IF scene is *poor* THEN shading is *unsuitable*
2.	IF scene is *fair* THEN shading is *fair*
3.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *low_priority* THEN shading is *strong*
4.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *medium_priority* THEN shading is *fair*
5.	IF scene is *suitable* AND tracing_data_path_in_3d_graph is *high_priority* THEN shading is *weak*
6.	IF scene is *suitable* AND surface_target_detection is *low_priority* THEN shading is *weak*
7.	IF scene is *suitable* AND surface_target_detection is *medium_priority* THEN shading is *fair*
8.	IF scene is *suitable* AND surface_target_detection is *high_priority* THEN shading is *strong*

Texture Gradient Rules	
1.	IF scene is *poor* THEN texture_gradient is *unsuitable*
2.	IF scene is *fair* THEN texture_gradient is *fair*
3.	IF scene is *suitable* AND surface_target_detection is low_priority THEN texture_gradient is *weak*
4.	IF scene is *suitable* AND surface_target_detection is medium_priority THEN texture_gradient is *fair*
5.	IF scene is *suitable* AND surface_target_detection is high_priority THEN texture_gradient is *strong*
6.	IF scene is *suitable* AND judging_relative_positions is low_priority THEN texture_gradient is *weak*
7.	IF scene is *suitable* AND judging_relative_positions is medium_priority THEN texture_gradient is *fair*
8.	IF scene is *suitable* AND judging_relative_positions is high_priority THEN texture_gradient is *strong*