# Computer Modeling and Simulation Techniques for Computer Vision Problems

A Dissertation Presented

by

Ming-Chin Lu

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

State University of New York
at
Stony Brook

May 1993

State University of New York
at Stony Brook

The Graduate School

Ming-Chin Lu

We the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of the dissertation.

Muralidhara Subbarao, Advisor, Associate Professor
Department of Electrical Engineering

John Murray, Associate Professor, Chairman
Department of Electrical Engineering

Dali Tao, Assistant Professor
Department of Electrical Engineering

Vaclav Dolezal, Professor
Department of Applied Mathematics & Statistics

This dissertation is accepted by the Graduate School.

Graduate School

# Abstract of the Dissertation

## Computer Modeling and Simulation Techniques for Computer Vision Problems


by

Ming-Chin Lu

Doctor of Philosophy

in

Electrical Engineering

State University of New York at Stony Brook

1993



Verification of computer vision theories is facilitated by the development and implementation of computer simulation systems. Computer simulation avoids the necessity of building actual systems; they are fast, flexible, and can be easily duplicated for use by others. Development and implementation of computational models in computer vision are both interesting and challenging. It involves research in diverse areas and requires integration of both science and technology.

This dissertation addresses the computer modeling and simulation techniques for two computer vision problems: object recognition and image sensing process. Image sensing process investigates how an image is sensed by specifying the input characteristics of the object and the imaging devices, while object recognition is a high level processing of the sensed image. We present a neural network model to solve the problem of *3-D* object identification and pose estimation. The network is divided into two stages, namely Feature Extraction Stage and Feature Detection Stage to extract the feature vectors and to identify the objects, respectively. *3-D* moments are used as input feature vectors to the network. Therefore, unoccluded objects are required. We also present an useful computational model to explore the image sensing process. This model decouples the photometric information and the geometric information of objects in the scene. Therefore, it is computationally tractable. Finally, we extend the proposed image sensing model to simulate the formation of moving objects and stereo imaging applications. All the models presented here have been implemented and the implementations are efficient, modular, extensible, and user-friendly so that others can easily reproduce and/or verify their experiments on a broader set of computer vision theories.

To my parents and my wife

# Contents

# List of Figures

# List of Tables

## Acknowledgements

I would first like to express my sincere thanks to my thesis advisor Prof. M. Subbarao for his guidance and inspiration. I am indebted to Prof. J. Murray, Prof. D. Tao, and Prof. V. Dolezal for helpful discussions and being on my thesis committee.

I am grateful for guidance from Prof. H.-S. Don and Dr. C.-H. Lo in the early stages of my research on *3-D* moment and neural network. I would also like to thank Mr. H.-C. Mike Lee for his assistance with computing facilities and helpful discussions. Special thanks to Ms. M. Krause for her kindness and help during my graduate studies.

This work was supported in part by the National Science Foundation under Grant IRI-8821923 and the Olympus Corporation.

# Chapter 1

# Introduction

## 1.1 Motivation

Obtaining information about three-dimensional scenes is a central problem in computer vision. The scene information that can be sensed are of two types: photometric and geometric. Photometric information consists of color and radiance of objects in the scene. Geometric information consists of shapes and distances of objects in the scene. The sensed data is in the form of digital images. Higher level processing of sensed data is necessary in tasks such as object recognition.

Many theories have been developed in computer vision during the past three decades for recovering scene information. Verification of such computer vision theories often require expensive and accurate camera systems, and laboratory facilities for calibration and experimentation. As an alternative, it is possible to develop computational models of the camera system, and simulate the system on a computer. This is not only faster and cheaper than building actual camera systems and setting up expensive laboratories, it also provides flexibility and accuracy. The physical parameters of the camera system (*e.g.* focal length, sampling rate, quantization level, noise characteristics, optical aberrations, etc.) are easily changed and they can be set to desired values to very high accuracy. The only major limitation is the amount of computational resources required for simulation.

The motivation for the research presented here is precisely the one mentioned above – that is to verify a set of computer vision theories. During the last five years, the computer vision research group in our department has developed new theories for object recognition, determining the distance of objects in a scene, and restoration of defocused images. In particular, verification of theories related to determining the distance of objects required an expensive camera system and elaborate laboratory set up. Even after a modest camera system and a laboratory became available, the necessity for a simulation system was acutely felt for the purpose of debugging the implementation of the theory on the actual camera system.

One goal of this research is to develop and implement computational models for verifying a set of vision theories. Another important goal is to make the implementations efficient, modular, extensible, portable and user-friendly so that other researchers can easily use and/or extend our developed systems to verify a much broader set of computer vision theories.

A valuable byproduct of the development of a computational model for the verification of a theory and its implementation is a better understanding of the theory itself. For example, consider our model (presented later) of the image sensing and digitization of a defocused image in a CCD camera. This model was developed to verify a method of determining distance and rapid autofocusing of camera systems. We were surprised to discover that over 15 (as opposed to only a few) intermediate steps had to be considered in the transformation of the light energy incident on the camera's entrance pupil into a digital image. Our attempt to develop these steps led us to detailed investigations into many areas: the effect of diffraction on the formation of defocused images, the technology of the Charge Coupled Devices (CCDs), the standards for video signal transmission and digitization, just to name a few. In contrast to the complexity of the actual image formation in a camera system, researchers in computer vision and computer graphics routinely over-simplify it and adopt a pinhole camera model. This over-simplification is especially unacceptable in the case of many computer vision applications.

The challenge of computer modeling and simulation lies not only in researching diverse areas and integrating scientific and technological information, but also in the development of computationally efficient algorithms and their implementation. A further challenge is to develop a system which many others can, and are willing to use it in their research. This requires the system to be user-friendly, portable, and easily extensible.

During the last three years of research, we have developed and implemented computational models for the following two problems: (i) object recognition based on *3-D* moment invariants using neural networks, and (ii) sensing and digitization of defocused images in a CCD camera system. This research is not being done in isolation for the purpose of doing research, but it is part of a larger project involving several researchers, and the results of this research are intended to be used by other researchers in our group. This goal has already been accomplished. Others have used our results to verify theories for object recognition, determining distance of objects, and autofocusing of camera systems.

## 1.2 Literature Review

### 1.2.1 Object Recognition

Bolle and Cooper [3] present a Bayesian classifier for local *3-D* shape recognition. They partition an image into many small windows within which surfaces can be locally approximated by one or two quadric surfaces in the *3-D* space, and then process these windows in parallel using a Bayesian minimum-probability-of-error recognition scheme. A detailed descriptions of other classifiers can be found in the book by Duda and Hart [10].

Reeves and Taylor [36] propose a model based recognition technique for the shape classification of three-dimensional objects using global features. They developed a method to generate an exhaustive set of library views and worst case views of an object. Their system also requires unoccluded views of objects, but no restrictions are placed on object position and orientation. They compute simple statistics for the library and balance the individual feature vector elements by dividing by the associated deviation of the database. Then, they store the balanced feature vectors in the balanced library. When there is an input test object, an algorithm is used to compute up to 252 test view (the viewing angles are taken from the sampling vertices – the worst case). They match each test view, by means of an Euclidean distance in feature space, to its closest entry in the library. This best match library entry is used to identify the type and orientation of the test object.

Lo and Don [27, 28] propose a *3-D* object identification method using *3-D* moments. They use a group-theoretic method to derive *3-D* moment invariants. Objects are then recognized by their shapes via moment invariants. They first compute the *3-D* moment invariants, and then use the computed moments as feature vectors to identify the *3-D* objects in the scene. A prototype 3-layer neural classification network is proposed to identify the object in the scene. Their network consists of 5 input nodes (corresponding to three second-order and two quadratic third-order moment invariants), 40 hidden nodes, and 2 output nodes to perform a two-class non-convex curved object classification problem.

Darwish and Jain [9] use *a priori* knowledge about the scene to coordinate and control bilevel image segmentation for visual pattern inspection. Jain and Hoffman [21] propose an evidence-based recognition algorithm to identify *3-D* objects by looking for notable features of objects. *3-D* range image is used as the database. Eight stages are involved in the object recognition process. The first six stages are used to retrieve an initial object representation. They are: 1) image acquisition stage to obtain range image; 2) image preprocessing stage to remove background and enhance the range image; 3) segmentation stage to get surface patches; 4) classification stage to get the classified surface patches; 5) merging stage to reconstruct the surfaces; 6) information retrieval stage to derive the initial object representation from the morphological properties of the range image, surface patches, and jump edges. The last two stages, modified representation stage and recognition stage, use an evidence rulebase which provides salient information about surface patches and pairs of patches for various objects in a database. Evidence rules support or refute hypotheses about the identity of an observed object. A measure of similarity between observed features derived from the range image and supporting features present in the evidence feature rules is developed for each object in the database. Then, the maximum similarity value is used to identify an object in the range image.

### 1.2.2 Camera Modeling and Image Sensing

Previous literature [13, 24, 37] in computer vision and computer graphics areas on camera system modeling are mostly either descriptive in nature or adopt a pin-hole camera model. This over-simplification is especially unacceptable in the case of many computer vision applications such as depth from defocus.

Potmesil and Chakravarty [35] extended camera modeling from the traditional pinhole projection model to a lens and aperture camera model. They add an aperture function to the basic lens model used in geometrical optics:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f}$$

where $u$ is the object distance, $v$ is the image distance, and $f$ is the focal length. They primarily dealt with only the effects of image defocus due to the lens and the aperture. Other characteristics of a CCD camera such as vignetting, CCD sensor geometry and response, CCD noise, etc. were not considered.

Chen [7] revised the algorithm proposed by Potmesil and Chakravarty [35] using simple light particle theory instead of the wave theory to avoid complicated computations and the huge memory consumption. A new algorithm was presented by him to deal with the highly defocused scenes due to the lens and aperture effects.

Shafer's work [42] is perhaps the first to deal with a detailed description of the image sensing process. Many important aspects of the image sensing process which are routinely ignored by computer vision researchers are discussed in detail by him. He has presented a twelve-parameter model for a robot imaging system – six parameters in camera position and orientation, three in the optical system, and three in sensitivity. In addition, he provides valuable information on the state of the art in the imaging system technology and calibration. However, Shafer's model is incomplete for the purpose of computer implementation. Further, it couples the effects of geometric and photometric aspects of a scene, thus making any modified version of it to be computationally intractable.

Recently, Subbarao and Nikzad [47] propose a mathematical model for a typical CCD camera system used in machine vision applications. Their model is based on a precise definition of the input to a camera system. This definition decouples the geometric properties of the scene from the photometric properties of the scene in the input to the camera system. They presented an ordered sequence of about 20 operations to transform the input properties to a digital image as sensed by the camera system. In order for their contribution to be used by other researchers, this mathematical model needs to be modified to obtain a computational model for implementation purpose. Furthermore, a friendly, user-controllable, flexible, and extensible user interface needs to be provided so that others are willing to use the simulation system.

### 1.2.3 Motion and Stereo Simulation

Algorithms for computing the images of moving objects are found in the Computer Graphics literature. Motion involves geometric transformations of images. Such transformations include translation, rotation, scaling, and other nonlinear operations. Most of the simulation algorithms adapt a pin-hole camera model to compute the new images. No defocus effect due to lens is modeled.

Weiman [50] has developed algorithms for performing scaling and shearing of images by rational amounts. Feibush *et al* [12] give a somewhat more sophisticated mechanism for image transformation using filtering techniques. A more detailed description of various transformations on motion simulation can be found in the book by Foley *et al* [13].

Stereo simulation can be considered as motion simulation on both the left and the right camera with the motion displacements correspond to the camera locations. Hodges and McAllister [17] propose a method for presenting stereo pairs to the eye. Neilson and Olsen [33] develop an algorithm to construct *2-D* images based on constrained *3-D* movement.

Krotkov's dissertation [23] addresses a set of visual perception problems called the spatial layout problems. In one of the problems, he uses a pair of verging cameras to develop and analyze a practical system for computing range from stereo. Other issues on camera calibration, focus ranging, and cooperative focusing and stereo are also discussed in his dissertation. The text book *Robot Vision* by Horn [19] also provides an excellent introduction to image sensing and formation, motion, and stereo vision issues.

### 1.2.4 Neural Network

Neural net models have brought many researchers attention for many years in the hope of achieving human-like performance on pattern recognition, speech recognition, and image understanding applications where many hypotheses are pursued in parallel and high computation speed are required. So far, the best systems are far behind the human performance. For these applications, neural net seems to be a good approach to the solution because it has the advantage of 1) highly parallel structure, 2) parallel distributed processing capability, 3) higher degree of fault tolerance capability than von Neumann sequential computers, and 4) the learning capability.

Lippmann [25] gives an excellent introduction to the field of neural network by reviewing six important neural net models that can be used for pattern classification. He divides these models into 4 categories as 1) binary input, supervised learning (*e.g.*, Hopfield net and Hamming net); 2) binary input, unsupervised learning (Carpenter/Grossberg classifier); 3) continuous-valued input, supervised learning (perception model and multi-layer perception model); and 4) continuous-valued input, unsupervised learning (Kohonen self-organizing map). Both the algorithms and the examples for these models are presented in his paper.

Hopfield and Tank [20] present a highly-interconnected network of non-linear analog neurons to solve the optimization problems. They model the processing elements (called *neurons*) as amplifiers in conjunction with feedback circuits comprised of resistors and capacitors. The transfer function of the amplifier is the sigmoid monotonic function. They define an energy function of the network and try to minimize this function. A traveling-salesman problem is used as a demonstration of their model.

Rumelhart *et al* present a multilayer network model and the back-propagation training algorithm in [38]. They extend the delta rule to the *generalized delta rule*. In this generalized rule, a *momentum* term is included to increase the learning rate without leading to oscillation. Several examples are used as illustrations on how their algorithms can be applied. We use their algorithm in the training process of the object recognition network to be presented later.

## 1.3 Thesis Overview

This dissertation is organized as follows: Chapter 2 presents the design of a neural network model to solve the problem of *3-D* object recognition. Both the object identification and pose estimation problems are solved concurrently. We use *3-D* moment invariants as feature vectors to the proposed network. The network itself is a multi-stage feed-forward neural network using back-propagation algorithm as the training rule.

A complete simulation program called OPEN (Object identification and Pose Estimation Network simulator) is developed to solve the problem of *3-D* object identification and pose estimation concurrently.

Chapter 3 discusses the image sensing model for CCD camera systems. We start from the image sensing process for a single camera system by specifying the input object parameters and the CCD camera parameters. The proposed model decouples the photometric information and the geometric information. Therefore, the computations involved in the image sensing process become tractable and implementable. This model is then extended to the simulation of moving objects and the simulation of stereo image pairs. The diffraction effect is also included.

Chapter 4 presents the IDS (Image Defocus Simulator) computer simulation system. This simulation system consists of a simulation engine and three user interfaces. The simulation engine implements the image sensing process starting from the object information, passing through the imaging device, and ending at the image detector to form the sensed image. In order for others to use our model to verify their computer vision theories/implementations, three user interfaces are provided – Sunview Graphical Interface (SGI), X-window Graphical Interface (XGI), and Dummy Terminal Interface (DTI) – so that users can easily enter the camera/object parameters to conduct the experiments. Some simulation results on the verification of the theories on Depth From Defocus (DFD) are included.

Chapter 5 describes the AVS (Active Vision Simulator) computer simulation system. The simulation of curved objects, moving objects, and stereo system are emphasized here. The user interface, simulation results, and some applications are also included.

This dissertation concludes with Chapter 6, which reviews what has been learned from this work, describes some natural extensions of it, shows how others can use our model and simulation systems in their work, and presents a final summary and future research topics.

# Chapter 2

# *3-D* Object Identification and Pose Estimation

## 2.1 Introduction

*3-D* Object identification and pose estimation are two of the major computer vision tasks in many applications such as robotics and mission planning systems. The goal of object identification is to identify an unknown object from the given model database; while that of pose estimation is to determine in real time what orientation the space shuttle, for example, is facing a *3-D* object in order to aim antennas, launch scientific instruments, and so on. As described in Chapter 1, some approaches use a feature extractor to extract presumably relevant information from the input data and then design a classifier to minimize the probability of error [10]. The other approaches generate a large set of model library

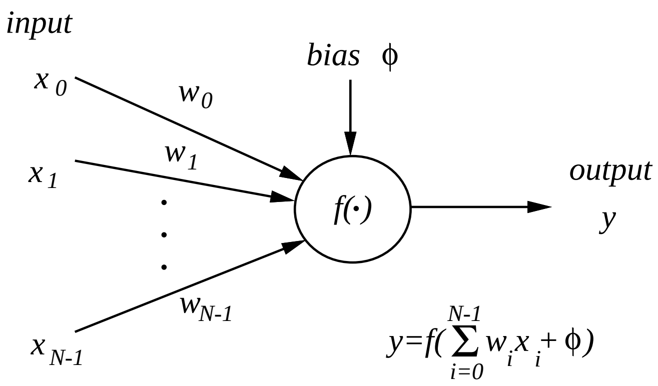



Figure 2-1: Computational elements in a neural network.

features and apply their algorithms to find the best match [8, 36]. These approaches require, in general, large amount of computation time and/or database.

The recently proposed 3-layer neural network has the capability of classifying patterns with arbitrarily complex decision regions [25, 39]. The network consists of individual computational elements or nodes, called *neurons*, connected by links with variable or fixed weights. Each element simply sums its weighted inputs and passes the result through a linear or nonlinear transfer function to generate its output as shown in Figure 2-1. The network has the advantages of 1) the simplicity of the computational elements, and 2) the potentially parallel structure. Our goal is to find a model to decompose the problem of *3-D* object identification and pose estimation into several parallelly realizable layers so that we can take the advantages of the neural network.

In this Chapter, we present a multi-stage neural network model

[29, 30] to identify *3-D* unoccluded objects from arbitrary viewing angles and to estimate their poses. We use *3-D* moment invariants as feature vectors to the network. Therefore, unoccluded views of objects are required. The invariants and vector moment functions are constructed using the complex moment derived from [27]. It is based on the Clebsch-Gordon expansion in group representation theory [11]. Moments are first expressed in the basis of spherical harmonic polynomials. They are called complex moments. Vectors and scalars are extracted from the compounds of complex moments via Clebsch-Gordon expansion. Higher order moment invariants are also derived in this way. They represent the fine spatial details on the objects.

The neural network model presented here is divided into two major stages: the *F*eature *E*xtraction *S*tage (FES) and the *F*eature *D*etection *S*tage (FDS). FES is a fixed-weight, biased neural network used to extract moment invariant features from the input image — range image which is commonly used in robotic vision applications. They are multiview representations of *3-D* objects with the *3-D* coordinates of points on the visible surfaces obtained from, *e.g.*, the laser range finders. After the feature vectors are extracted, they are fedforward to the FDS which consists of an object identification subnetwork and $c$ pose estimation subnetworks to handle $c$-class problems. They are all variable-weight, biased neural networks. A network arbitrator is also designed to dispatch the input vectors and to choose one of the output poses during training and normal operation processes, respectively. Based on this approach, object identification and pose estimation can be done concurrently. We have developed a simulation package called Object identification and Pose Estimation Network simulator (OPEN) and conducted simulations on both convex and non-convex curved objects. The *3-D* moment features of objects were extracted from the FES stage and used as the input features to the networks in the FDS stage. After the network is trained, some randomly generated testing images are fed into the network to evaluate the performance of the system. The performance is evaluated using different number of hidden units, number of training iterations, and the computational complexity. After a reasonable amount of training cycles, the results show that (1) a high percentage of objects can be correctly identified even though some of them look similar; and (2) the orientation can be successfully estimated which could be used for robot applications.

This Chapter is organized as follows: Section 2 presents the computation of *3-D* geometric moments and moment invariants for *3-D* range-data; Section 3 describes the computational model of the network; Section 4 discusses the training process; Section 5 discusses the normal operation processes; Section 6 presents the computer simulation; Section 7 presents the simulation results; and finally, Section 8 concludes this Chapter with a summary of results and a few remarks.

## 2.2 Feature Extraction Using *3-D* Moment Invariants

Lo and Don[27] propose a procedure to compute *3-D* moment and apply the computed moment invariants to object identification and positioning. Objects are recognized by their shapes via moment invariants. The work on object identification and pose estimation we are going to present in this Chapter is an extension of the work proposed by them. Here we present a neural network approach to compute, in FES, the moment invariants proposed by them and then feed the parallelly computed moments to FDS to identify the object and estimate its pose. In this section, we will summarize how the moment invariants are computed based on their theoretical contribution.

The *3-D* moment of order $p = l + m + n$ of a *3-D* density function $\rho(x, y, z)$ is defined by the Riemann integrals:

$$M_{lmn} = \frac{1}{N} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^l \, y^m \, z^n \, \rho(x, y, z) \, dx \, dy \, dz \tag{2.1}$$

where the constant $N$ is used to normalize the integral so that $M_{000} = 1$. If the density function is piecewisely continuous and bounded in a finite region in $R^3$ space, then moments of all order exist. In this case, the characteristic function of $\rho(x, y, z)$ can be defined as

$$M(u_1, u_2, u_3) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp^{j(u_1 x + u_2 y + u_3 z)} \; \rho(x, y, z) \, dx \, dy \, dz \tag{2.2}$$

where $u_1$, $u_2$, and $u_3$ are spatial frequency components. Equation (2.2) can be expanded into a power series

$$M(u_1,u_2,u_3) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\sum_{p=0}^{\infty}\frac{j^p}{p!}(u_1x+u_2y+u_3z)^p\,\rho(x,y,z)\,dx\,dy\,dz \quad (2.3)$$

By interchanging the integration and summation in Equation (2.3), the characteristic function can be expressed as an infinite series of homogeneous polynomials of $u_1$, $u_2$, and $u_3$ as

$$M(u_1,u_2,u_3) = \sum_{p=0}^{\infty}\frac{j^p}{p!}H_p(u_1,u_2,u_3) \quad (2.4)$$

where

$$H_p(u_1,u_2,u_3) = \sum_{l,m,n=0}^{p}\frac{p!}{l!m!n!}M_{lmn}u_1^l u_2^m u_3^n \quad (2.5)$$

Introducing a vector $\vec{u}$ whose components are the monomial $u_1^l u_2^m u_3^n$ and a vector $\vec{m}$ whose components are the coefficients of $u_1^l u_2^m u_3^n$ in $H_p$, that is,

$$\begin{aligned} \vec{u} &= [u_1^p \quad u_2^p \quad \cdots \quad u_1^l u_2^m u_3^n \quad \cdots] \\ \vec{m} &= [M_{p00} \quad M_{0p0} \quad \cdots \quad \frac{p!}{l!m!n!}M_{lmn} \quad \cdots] \end{aligned} \quad (2.6)$$

so that $\vec{u}\cdot\vec{m} = H_p$. The components of $\vec{u}$ and $\vec{m}$ may be arranged in different order as long as their scalar product remains to be $H_p$.

In the *3-D* space, the moments of a *2-D* surface patch, parameterized by $u_1$ and $u_2$, can be computed by the following equation:

$$M_{lmn} = \frac{1}{A}\int\!\!\int x^l(u_1,u_2)\,y^m(u_1,u_2)\,z^n(u_1,u_2)\sqrt{g}\,du_1\,du_2 \quad (2.7)$$

where $A$ is the area of the patch and $g$ is the determinant of the metric tensor of the *2-D* surface. In range data applications, the surface in the range image is represented by a monge patch $z = z(x, y)$ and the image plane is parameterized by the coordinates $u_1 = x$ and $u_2 = y$. In such case, Equation (2.7) becomes:

$$M_{lmn} = \frac{1}{A} \int\int_R x^l\, y^m\, z^n(x, y) \sqrt{1 + (\frac{\partial z}{\partial x})^2 + (\frac{\partial z}{\partial y})^2}\, dx\, dy \tag{2.8}$$

where $R$ is the projection of the surface patch on the image plane.

Moments with their origin at the centroid of the density function are called central moments. The central moments have the nice property of invariance under translation. We use second and third order moment invariants derived from the vectors and scalars which are extracted from the compounds of complex moments as shown in [27]. Moment vectors constructed from higher order moments are too expensive to compute and are very sensitive to noise.

For second order moment invariants,

$$\vec{u} = [u_1^2 \quad u_2^2 \quad u_3^2 \quad u_1u_2 \quad u_1u_3 \quad u_2u_3]$$

the components of $\vec{u}$ can be expressed as a linear combination of the symmetric tensor space of rank 2 [11]. This results in five second-order complex moments $\vec{\nu} = [\nu_2^2 \quad \nu_2^1 \quad \nu_2^0 \quad \nu_2^{-1} \quad \nu_2^{-2}]$:

$$\begin{aligned}
\nu_2^2 &= \sqrt{\frac{2\pi}{15}}(M_{200} - M_{020} + j2M_{110}), \\
\nu_2^1 &= \sqrt{\frac{2\pi}{15}}(-2M_{101} - j2M_{011}), \\
\nu_2^0 &= \frac{2\sqrt{\pi}}{3\sqrt{5}}(2M_{002} - M_{200} - M_{020}),
\end{aligned} \tag{2.9}$$

$$\begin{aligned}
\nu_2^{-1} &= \sqrt{\frac{2\pi}{15}}(2M_{101} - j2M_{011}), \\
\nu_2^{-2} &= \sqrt{\frac{2\pi}{15}}(M_{200} - M_{020} - j2M_{110}).
\end{aligned}$$

The ten third-order moments

$$\vec{u} = [u_1^3 \quad u_2^3 \quad u_3^3 \quad u_1u_2^2 \quad u_1^2u_2 \quad u_1u_3^2 \quad u_1^2u_3 \quad u_2u_3^2 \quad u_2^2u_3 \quad u_1u_2u_3]$$

are decomposed into two complex moment multiples. Seven of the third-order complex moments, denoted by

$$\vec{\nu_3} = [\nu_3^3 \quad \nu_3^2 \quad \cdots \quad \nu_3^{-2} \quad \nu_3^{-3}]$$

are:

$$\begin{aligned}
\nu_3^3 &= \sqrt{\frac{\pi}{35}}[(-M_{300} + 3M_{120}) + j(M_{030} - 3M_{210})], \\
\nu_3^2 &= \sqrt{\frac{6\pi}{35}}[(M_{201} - M_{021}) + j2M_{111}], \\
\nu_3^1 &= \frac{\sqrt{3\pi}}{5\sqrt{7}}[(M_{300} + M_{120} - 4M_{120}) + j(M_{030} + M_{210} - 4M_{012})], \\
\nu_3^0 &= \frac{2\sqrt{\pi}}{5\sqrt{7}}[2M_{003} - 3M_{201} - 3M_{021}], \qquad (2.10) \\
\nu_3^{-1} &= \frac{\sqrt{3\pi}}{5\sqrt{7}}[(-M_{300} - M_{120} + 4M_{120}) + j(M_{030} + M_{210} - 4M_{012})], \\
\nu_3^{-2} &= \sqrt{\frac{6\pi}{35}}[(M_{201} - M_{021}) - j2M_{111}], \\
\nu_3^{-3} &= \sqrt{\frac{\pi}{35}}[(M_{300} - 3M_{120}) + j(M_{030} - 3M_{210})].
\end{aligned}$$

Another three third-order complex moments, denoted by

$$\vec{\nu_1} = [\nu_1^1 \quad \nu_1^0 \quad \nu_1^{-1}]$$

are:

$$\begin{aligned}
\nu_1^1 &= \frac{\sqrt{6\pi}}{5}[(-M_{300} - M_{120} - M_{102}) - j(M_{030} + M_{210} + M_{012})], \\
\nu_1^0 &= \frac{2\sqrt{3\pi}}{5}[M_{003} + M_{201} + M_{021}], \qquad (2.11) \\
\nu_1^{-1} &= \frac{\sqrt{6\pi}}{5}[(M_{300} + M_{120} + M_{102}) - j(M_{030} + M_{210} + M_{012})],
\end{aligned}$$

These equations give a relationship between geometric moments and complex moments. The moment invariants can be computed as [27]:

$$\begin{aligned}
\nu_0^0 &= \frac{2}{3}\sqrt{\pi}(M_{200} + M_{020} + M_{002}), \\
\nu(2,2)_0^0 &= (5)^{\frac{-1}{2}}[2\nu_2^2\nu_2^{-2} - 2\nu_2^1\nu_2^{-1} + (\nu_2^0)^2], \qquad (2.12) \\
\eta\nu &= (5)^{\frac{-1}{2}}(\eta_2^2\nu_2^{-2} - \eta_2^1\nu_2^{-1} + \eta_2^0\nu_2^0 - \eta_2^{-1}\nu_2^1 + \eta_2^{-2}\nu_2^2.
\end{aligned}$$

where $\eta$ is a second rank spherical tensor and

$$\eta_2^m = \sum_{i=-2}^{i=2} \langle 2, i, 2, m-i|2, 2, 2, m\rangle \, \nu_2^i \nu_2^{m-i}$$

Note that $\eta\nu$ is a cubic polynomial of second order moments obtained by Sadjadi and Hall [41]. Two quadratic moment invariants containing the third-order moments are given below

$$\begin{aligned}
\nu(3,3)_0^0 &= (7)^{\frac{-1}{2}}[2\nu_3^3\nu_3^{-3} - 2\nu_3^2\nu_3^{-2} + 2\nu_3^1\nu_3^{-1} - (\nu_3^0)^2)], \\
\nu(1,1)_0^0 &= (3)^{\frac{-1}{2}}[2\nu_1^1\nu_1^{-1} - (\nu_1^0)^2]. \qquad (2.13)
\end{aligned}$$

When each invariant moment feature is divided by suitable power of $\nu_0^0$, it becomes invariant under change of size. Those similitude-invariant quantities derived from the moment invariants in the above equations are given below

$$
\begin{aligned}
I_{22}^{2} &= \frac{\nu(2,2)_0^0}{(\nu_0^0)^2}, && (2.14)\\
I_{222}^{2} &= \frac{\eta\nu}{(\nu_0^0)^3}, && (2.15)\\
I_{33}^{3} &= \frac{\nu(3,3)_0^0}{(\nu_0^0)^{\frac{12}{5}}}, && (2.16)\\
I_{11}^{3} &= \frac{\nu(1,1)_0^0}{(\nu_0^0)^{\frac{12}{5}}}. && (2.17)
\end{aligned}
$$

Other moment invariants can be similarly derived. The results are:

$$
\begin{aligned}
I_{233}^{2,3} &= \frac{\nu(3,3)_2\nu_2}{(\nu_0^0)^{\frac{17}{5}}}, && (2.18)\\
I_{123}^{2,3} &= \frac{\nu(3,1)_2\nu_2}{(\nu_0^0)^{\frac{17}{5}}}, && (2.19)\\
I_{112}^{2,3} &= \frac{\nu(1,1)_2\nu_2}{(\nu_0^0)^{\frac{17}{5}}}, && (2.20)\\
I_{3333}^{3} &= \frac{\nu^2(3,3)_2}{(\nu_0^0)^{\frac{24}{5}}}, && (2.21)\\
I_{1333}^{3} &= \frac{\nu(3,3)_2\nu(3,1)_2}{(\nu_0^0)^{\frac{24}{5}}}, && (2.22)\\
I_{1133}^{3} &= \frac{\nu^2(3,1)_2}{(\nu_0^0)^{\frac{24}{5}}}, && (2.23)\\
I_{1113}^{3} &= \frac{\nu(3,1)_2\nu(1,1)_2}{(\nu_0^0)^{\frac{24}{5}}}. && (2.24)
\end{aligned}
$$

## 2.3 The Network Architecture

The block diagram of the proposed network is shown in Figure 2-2. It is divided into two major stages: *F*eature *E*xtraction *S*tage (FES)

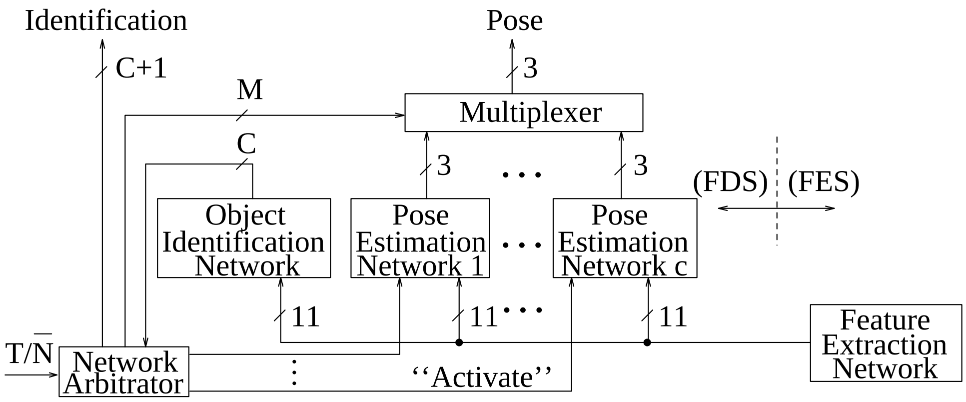



Figure 2-2: Block diagram of the proposed object identification and pose estimation network.

and *F*eature *D*etection *S*tage (FDS). FES is a fixed-weight, biased neural network to extract moment invariant features from the input image. FDS consists of $c$+1 three-layer, variable-weight, biased neural networks to detect input features and to map them into one of $c$ target classes and one of its orientations. A network arbitrator is also included to dispatch the input vectors and to choose one of the output poses during training and normal operation processes, respectively.

### 2.3.1 Feature Extraction Stage

Consider a digitized $N \times N$ range image, the *3-D* geometric central moment can be computed as:

$$M_{lmn} = \frac{1}{M_{000}} \sum_x \sum_y (x - M_{100})^l (y - M_{010})^m (z(x, y) - M_{001})^n \rho(x, y, z) \quad (2.25)$$

where

$$\rho(x,y,z) = \begin{cases} \sqrt{1+(\frac{\triangle z}{\triangle x})^2+(\frac{\triangle z}{\triangle y})^2}, & \text{if } (x,y) \text{ is not a background pixel} \\ \\ 1, & \text{otherwise} \end{cases} \tag{2.26}$$

Note that central moments are used such that the resulting moment invariants are invariant under translation.

Knowing that $x^l y^m z^n = \exp(l \ln x + m \ln y + n \ln z)$, we can rewrite the summation kernel in Equation (2.25) as

$$\begin{aligned} & exp[l \cdot \ln(x - M_{100}) + m \cdot (y - M_{010}) + n \cdot (z(x,y) - M_{001}) + \\ & \frac{1}{2} \ln \rho^2(x,y,z)] \\ = \; & exp\{l \cdot \ln(x - M_{100}) + m \cdot (y - M_{010}) + n \cdot (z(x,y) - M_{001}) + \\ & \frac{1}{2} \ln[1 + (\frac{\triangle z}{\triangle x})^2 + (\frac{\triangle z}{\triangle y})^2\} \end{aligned} \tag{2.27}$$

As one can see in Equation (2.27), the complicated computation of the product-of-power terms can be simplified as the summation of weighted linear terms passing through a transfer function. Therefore, it can be efficiently and rapidly computed by using multi-layer perceptron networks with different transfer functions in each layer. The resulting network is shown in Figure 2-3 where solid lines represent pre-calculated fixed weight connections. Our approach consists of the following steps:

(S.1) *Compute the square of the density function* $\rho(x,y)$.

Design two $N \times N$-neuron planes with the transfer

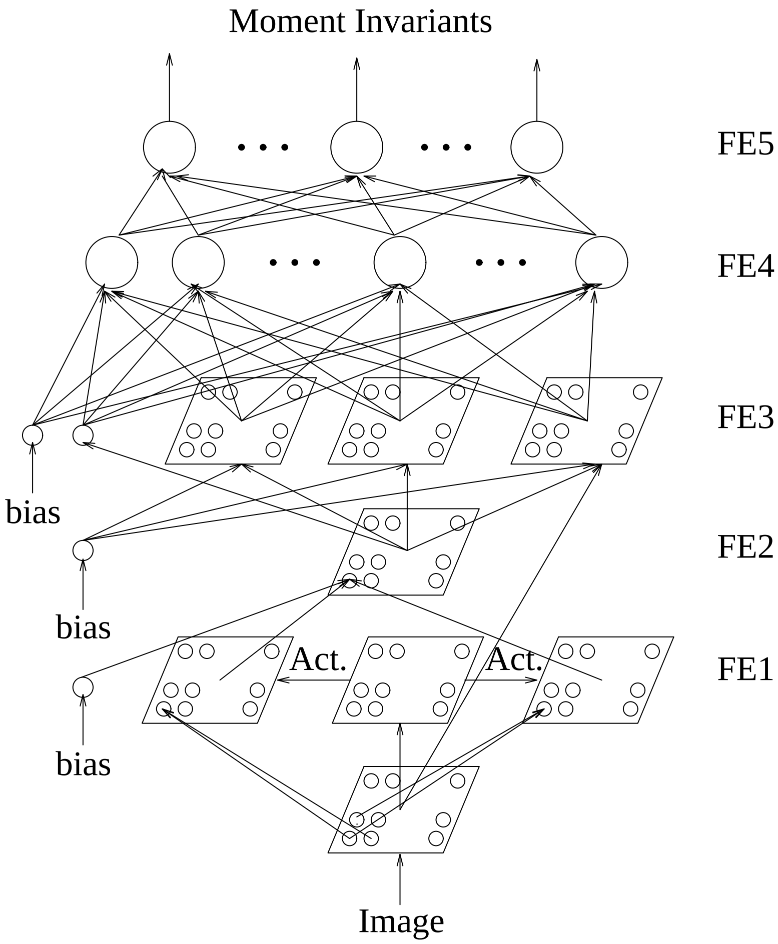



Figure 2-3: Feature extraction network.

function

$$f_{FE1}(x) = x^2.$$

The weights connecting from the image plane, $N \times N$ neurons, to those two planes are pre-calculated fixed values such that the output values in $FE1$ layers are $(\frac{\triangle z}{\triangle x})^2$ and $(\frac{\triangle z}{\triangle y})^2$ in the left and right plane, respectively. The decision of background/nonbackground pixels is made by the middle plane which has a transfer function, *e.g.*, of

$$f_{FE1M}(x) = U(-x + t).$$

where $U(\cdot)$ is the unit step function and $t$ is the threshold. This plane provides "activation" inputs to the other two planes. The output of this step is $\rho^2(x, y)$.

(S.2) *Compute* $\ln \rho^2(x, y)$ *using the following transfer function in* $FE2$ *layer:*

$$f_{FE2}(x) = \ln(x).$$

(S.3) *Compute* $\ln(x - M_{100})$, $\ln(y - M_{010})$, *and* $\ln(z(x, y) - M_{001})$ *in* $FE3$ *layer.*

The terms $x$, $y$ are encoded in the fixed weight connections coming from the bias terms; while the $z(x, y)$ term comes from the input image. The transfer function used in this layer is

$$f_{FE3}(x) = \ln|x|.$$

except that the neuron next to the left of the leftmost neuron plane uses a function of $f_{FE3L}(x) = 1/x$ to compute the scaling coefficients $1/M_{000}$.

(S.4) *Compute $M_{lmn}$.*

Construct a row of neurons in $FE4$ layer with the transfer function

$$f_{FE4}(x) = \exp(x).$$

The order $p = l + m + n$ is encoded in the weights between $FE3$ and $FE4$ layers.

(S.5) *Compute the moment invariants based on the coefficients derived in [27]*

Using the above procedure, we can generate moment invariant feature vector $\vec{I}$

$$\vec{I} = [I^2_{22} \quad I^2_{222} \quad I^3_{33} \quad I^3_{11} \quad I^{2,3}_{233} \quad I^{2,3}_{123} \quad I^{2,3}_{112} \quad I^3_{3333} \quad I^3_{1333} \quad I^3_{1133} \quad I^3_{1113}]$$

Note that, some coefficient arrangements must be done to avoid any singularity in the above transfer functions.

### 2.3.2 Feature Detection Stage

This stage consists of $c + 1$ networks – one *O*bject *I*dentification *N*etwork (OIN) and $c$ *P*ose *E*stimation *N*etworks (PEN). The structure of each network is shown in Figure 2-4 where dashed lines are used to

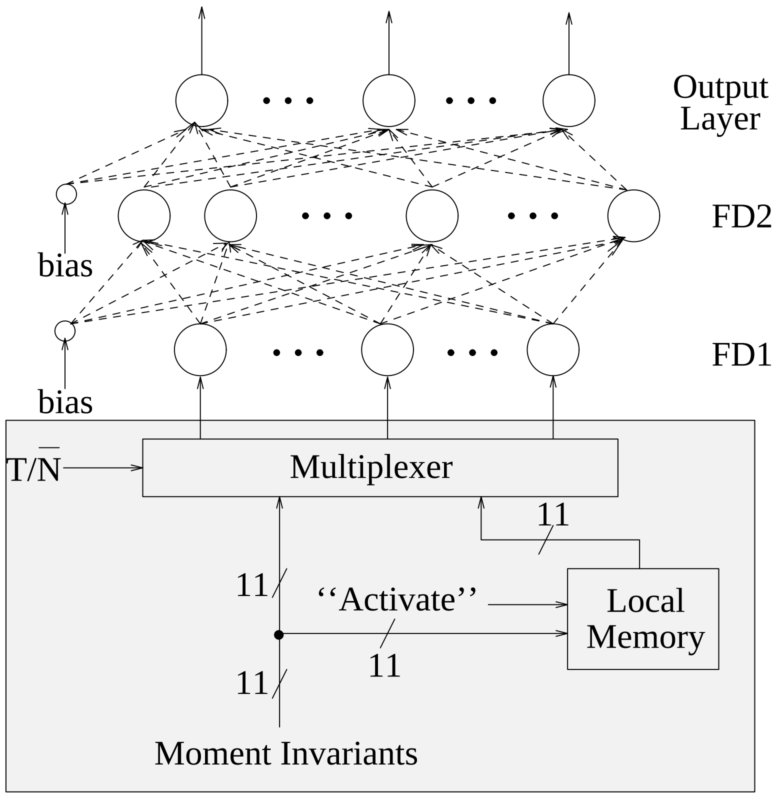



Figure 2-4: Network structure in FDS.

represent adaptable weight connections. The output of the OIN is either logic zero or one to represent the corresponding object; while the outputs of the PEN are continuous values between $0$ to $\pi$ to represent the angles $\alpha$, $\beta$, $\gamma$ between the principle axis of the object and the positive $x$-, $y$-, and $z$-axis, respectively.

Since analog output is required to represent the angles, we will need a large amount of training cycles for PENs to converge. This is accomplished by associating a local memory with each network to increase the bandwidth of the PENs. When the network is in the off-line training process, identified by the signal $T/\overline{N} = 1$, the feature vector $\vec{I}$ extracted from the previous stage is *dispatched* to both the local memory of OIN and one of the local memories of PEN, depending on the *activation* status of the PEN which is determined by the network arbitrator. After the memories are filled with proper training samples, every networks start their training cycles by *disabling* the write ability to their local memories and *enabling* the multiplexors. These $c$+1 networks will then function independently and concurrently afterwards. By doing so, the training cycles used for PENs are $c$ times those used for OIN. This is to trade network complexity for flexibility and computational efficiency.

Basically, the networks in this stage are 3-layer perceptrons. The learning rule used is the back-propagation training algorithm which uses sigmoidal threshold function and iterative gradient descent method to adapt the weights in the network[39]. To speed up the convergence, we implement the bias units and use momentum term in the generalized delta rule.

### 2.3.3 The Network Arbitrator

The *N*etwork *A*rbitrator (NA) is a combinational circuit used to control the operations of the proposed network. It takes as input the control signal $T/\overline{N}$ and the output vector of OIN. The output signals are $c$-bit control lines and $M$-bit ($M = \lceil \log_2 c \rceil$) selection lines to handle the operations of the training and normal operation processes, respectively. The detailed operations of these two processes will be discussed in the following two sections.

During training operation process, NA dispatches in a pre-specified order the input feature vectors into the proper local memory locations within OIN and PENs such that the local memory of OIN is filled with all the input feature vectors and that of PEN is filled with all the input feature vectors belonging to that class. When an image is sent to the trained network, NA selects the "*most likely*" object as the target and guarantees the output of the corresponding pose if the outputs of OIN exceed the given threshold. Should none of the outputs of OIN exceed that threshold, NA will reject the input image and classify it as "*undecided*". In case of a tie, NA selects the "*most important*" one to minimize the *risk* [10] that might happen.

## 2.4 Training Operation Processes

The operation of the proposed network is divided into two processes, namely, training process and normal operation process. During training process, feature vectors will be sent to the corresponding local memories first. After that, local memories are write-protected and used as the input sources to their corresponding network. Finally, each network in FDS starts training its weights by using the local memory inside that network. The training of each network is summarized below:

(T.1) *Compute the output at node i, $o_{h_i}^n$ and $o_{o_i}^n$, of the hidden and the output layer when the $n$-th input vector is presented.*

$$\begin{aligned} o_{h_i}^n &= f(\sum_k w_{ik}^{n-1} I_k^n + \phi_i^n) \\ o_{o_i}^n &= f(\sum_k w_{ik}^{n-1} o_{h_k}^n + \phi_i^n) \end{aligned}$$

(T.2) *Compute the delta weight at node i, $\delta_{o_i}^n$ and $\delta_{h_i}^n$, of the output and the hidden layer.*

$$\begin{aligned} \delta_{o_i}^n &= (t_i^n - o_{o_i}^n) o_{o_i}^n (1 - o_{o_i}^n) \\ \delta_{h_i}^n &= o_{o_i}^n (1 - o_{o_i}^n) f(\sum_k \delta_{o_k}^n w_{ki}^{n-1}) \end{aligned}$$

(T.3) *Update the weight.*

$$w_{ij}^n = \begin{cases} w_{ij}^{n-1} + \xi \delta_{o_i}^n o_{h_j}^n + \theta(w_{ij}^{n-1} - w_{ij}^{n-2}), \\ \qquad\qquad \text{if } i\text{-th unit is in the output layer.} \\ w_{ij}^{n-1} + \xi \delta_{h_i}^n I_j^n + \theta(w_{ij}^{n-1} - w_{ij}^{n-2}), \\ \qquad\qquad \text{if } i\text{-th unit is in the hidden layer.} \end{cases}$$

where $w_{ij}^n$ is the weight from the $j$-th to the $i$-th unit at the $n$-th input presentation number; $\phi_i^n$ is the biased weight connected to node $i$; $\xi$ is the learning rate; $t_i^n$ is the $i$-th component of the $n$-th target vector; $\theta$ is the momentum; and $f(\cdot)$ is the sigmoid function.

## 2.5 Normal Operation Processes

Once the network is trained, local memories are no longer needed. This is done by removing the shaded block in Figure 2-4 and by combining the FE5 layer in Figure 2-3 and the FD1 layer in Figure 2-4. Since then, the network is operated in normal operation process. When a range image is presented in the network input, its feature vector will be extracted by FES and sent to OIN and all the PENs. Based on the trained weights, FDS can identify the object and estimate $c$ poses in the $c$ PENs. The correct pose will then be selected by the network arbitrator.

## 2.6 Computer Simulation

Based on the proposed neural network model, we have developed and implemented a computer simulation program called OPEN (*O*bject identification and *P*ose *E*stimation *N*etwork simulator). The objects are generated using ray casting algorithm [40]. In this section, we are going to describe the *3-D* scene generator and the OPEN simulation program.

### 2.6.1 Object Generation using Ray Casting Algorithm

Roth [40] models solid objects by combining primitive solids, such as blocks and cylinders, using the set operators union, intersection, and difference. We have implemented his model in ANSI C and the implemented program can be easily ported to various platforms (*e.g.*, VAX, SUN, and PCs).

The program generates *3-D* scenes consisting of primitives of blocks, spheres (ellipsoids), cylinders, and cones. These primitives can be arbitrarily scaled, rotated, and translated. The input data to the program is organized as a node-based tree structure. Each node contains 14 fields:

```
struct _NODE {
    char *label;
    int address;
    char *node_type;  /*"composite" or
                        "primitive" */
    char *op;         /*"union", "intersection",
                         or "difference" */
```

```
    char *primitive;  /*"block", "sphere",
                         "cylinder" or "cone" */
    float a, b, r;    /* rotation */
    float sx, sy, sz; /* scaling */
    float tx, ty, tz; /* translation */
};
```

The leaf nodes of the tree are the primitive nodes while the internal nodes are the composite nodes. The *3-D* transform is associated with each node in the tree. This program generates *3-D* range image with hidden surface removed. Figure 2-5 is the composite tree of the “depth stop” machine part. The corresponding input description file can be found in Appendix A.

### 2.6.2 OPEN: The Object Identification and Pose Estimation Network Simulator

Figure 2-6 is the functional block diagram of the OPEN simulator. The OPEN kernel consists of a FDS stage, a network arbitrator, and I/O interface routines. When the FDS has never been trained (we call this the initial training process), OPEN takes the range image as input and generates the feature vector $\vec{I}$ via the FES stage. The users can specify the number of training cycles used to train the FDS. After the number of training cycles has been reached, OPEN outputs (i) the internal states (weights) of the FDS stage to a file, (ii) the generated moment invariant vectors, and (iii) the actual number of training cycles used if the network

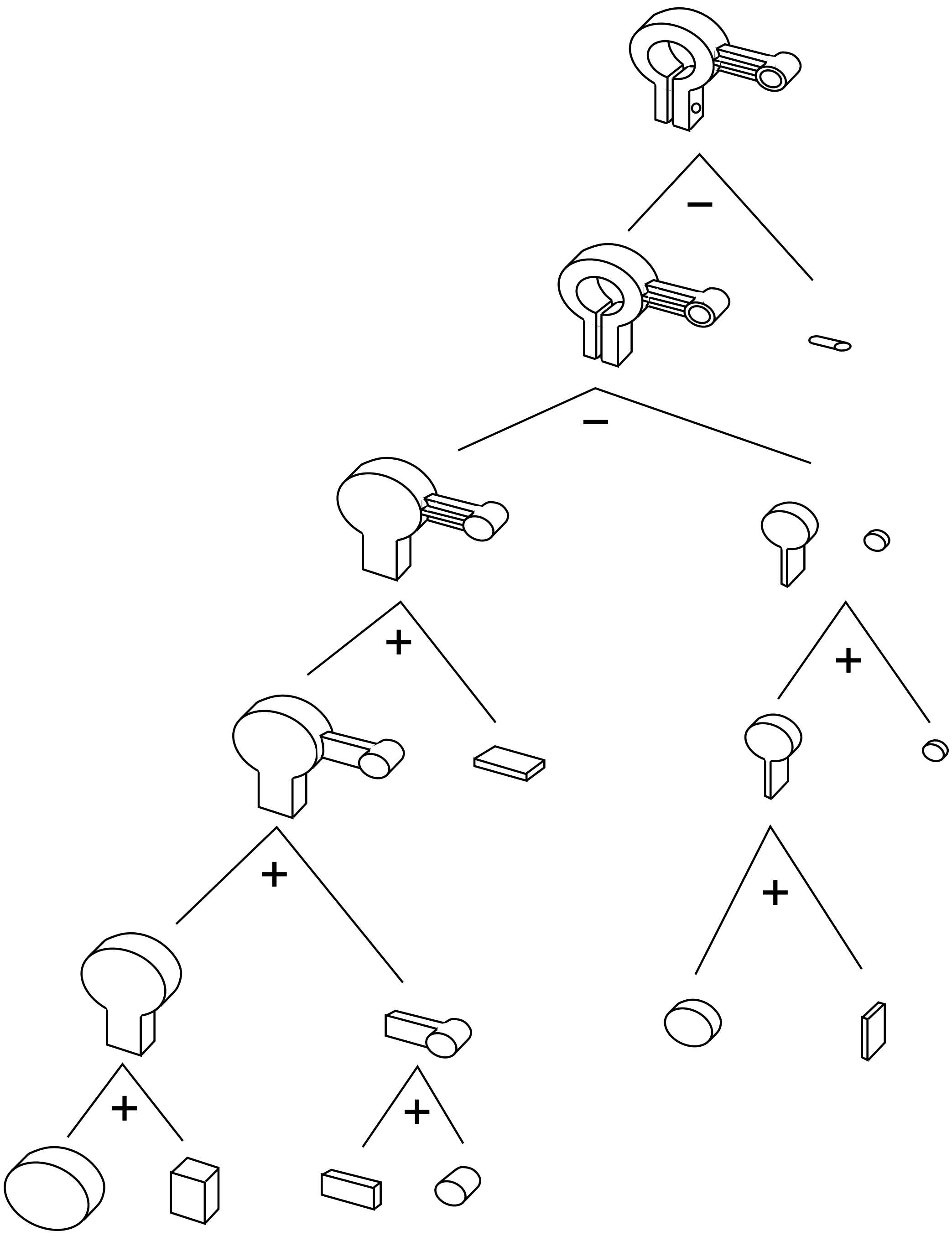
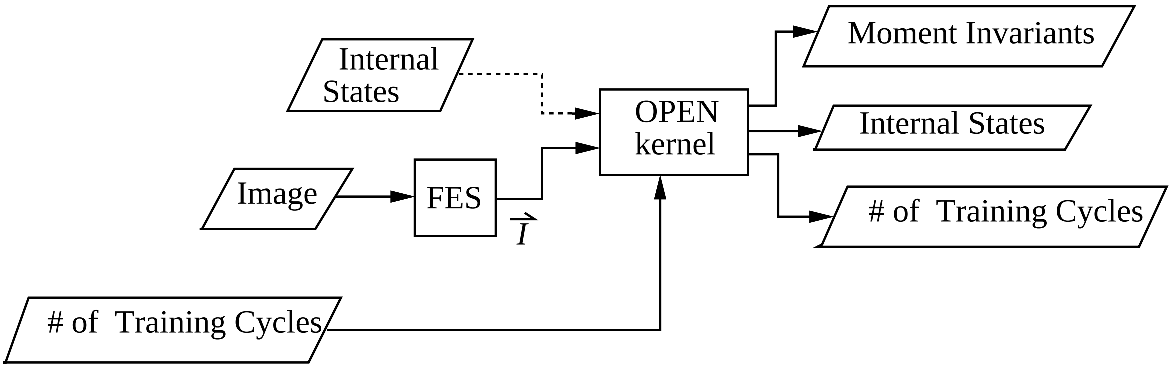



Figure 2-5: The composite tree of a machine part.

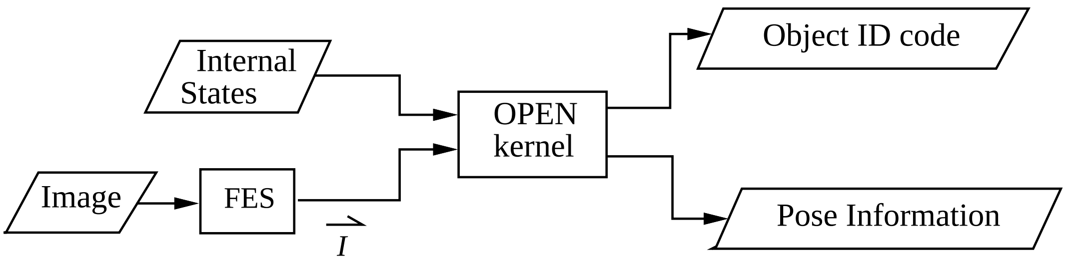
converges before the user-specified number of training cycles has been reached. If the network is not properly trained, the user can continue training the network by specifying the name of the internal state file as shown in the dashed line in Figure 2-6(a) (we call this the continued training process). Using this methodology, user does not have to waste time in re-training the network from the beginning. Instead, he/she can load the previously trained FDS states and keep training the network until the results are satisfactory.

When the training process is finished, user can feed the *3-D* range image to the normal operation process to test the performance of the trained network. The operation status of the OPEN simulator is determined by a command line switch which instructs the OPEN simulator to perform (i) the initial training process, (ii) the continued training process, or (iii) the normal operation process.

The OPEN simulator, as the name suggests, provides a open system methodology in the sense that user can easily modify/adapt the modules of the simulator to use different training algorithms and/or feature vector extraction methods. The proposed network is highly parallel and suitable for hardware implementation.




(a) Training operation process.




(b) Normal operation process.

Figure 2-6: The functional block diagram of the OPEN simulator.

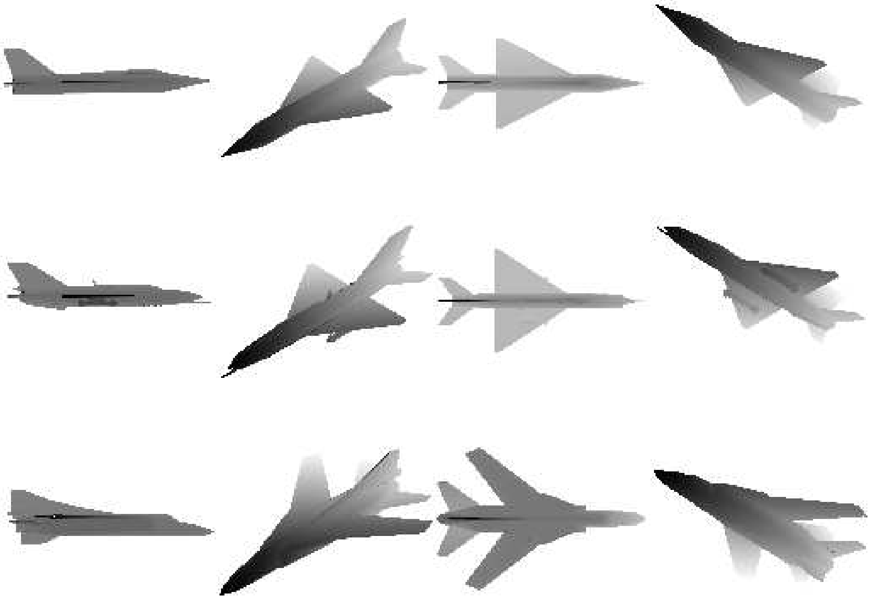


Figure 2-7: Sample airplanes X, Y, Z.

## 2.7 Simulation Results

Two simulations were conducted. The first simulation uses three airplanes $X$, $Y$, and $Z$ as shown in the first, second, and third row of Figure 2-7, respectively. The only difference between $X$ and $Y$ is that $Y$ is armed with one missile under each wing, while $X$ is not. The second simulation uses three computer-generated industrial machine parts: *depth stop, column base,* and *wedge lift* as shown in Figure 2-8.

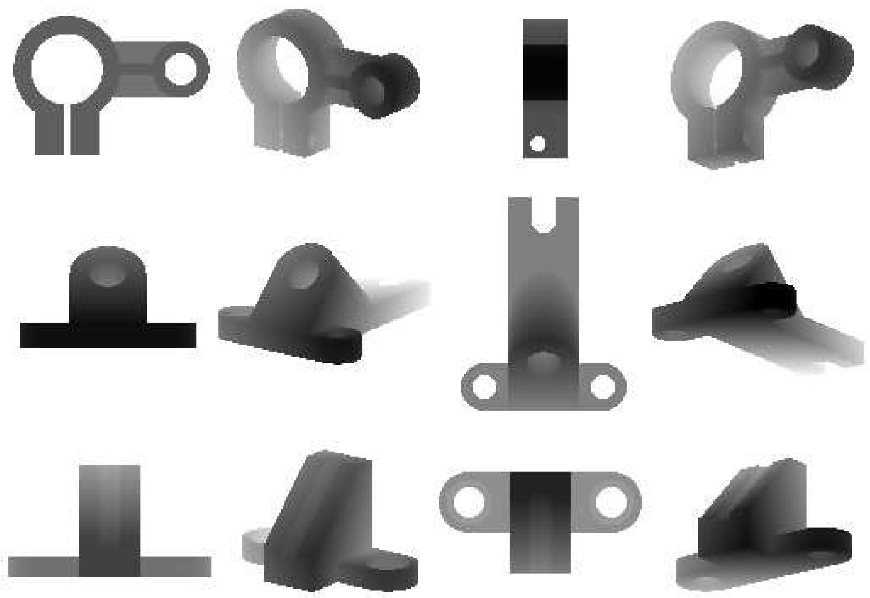


Figure 2-8: Sample machine parts.

### 2.7.1 The Simulations

In order to generate the training database, we put each object in the center of a sphere and generate the range images from the viewpoints which equally sample the intersection circle of the sphere and a horizontal plane. The angles between the $x$-$y$ plane and the sample points are chosen to be $\pm 90°$, $\pm 72°$, $\pm 54°$, $\pm 36°$, $\pm 18°$, and $0°$ to generate 1, 8, 12, 16, 20, and 24 images, respectively (refer to Figure 2-9 and Table 2-1). We also randomly generate 118 images as testing database for each object. This results in a total of 768 sample images in each simulation.

In Figure 2-9, for a point on a unit sphere with angle $\theta_x$, $\theta_y$, and $\theta_z$

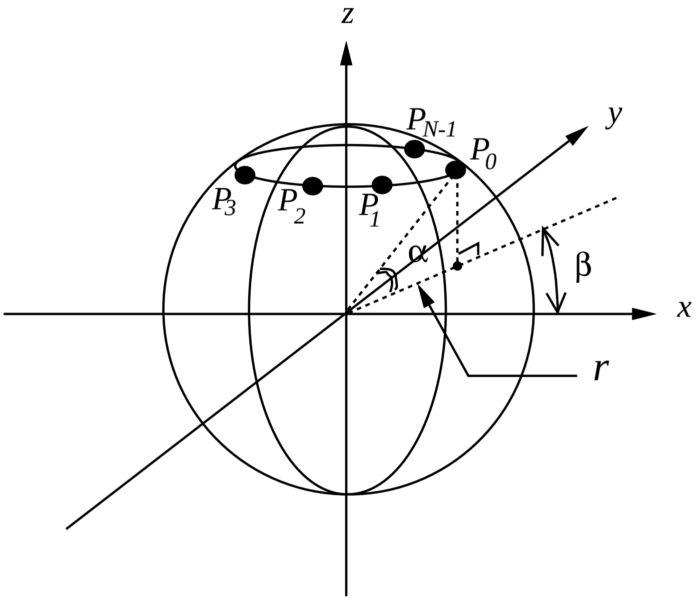



Figure 2-9: Viewing angle in pose estimation.
.

| $\alpha(degree)$ | 90 | 72 | 54 | 36 | 18 | 0 | -18 | -36 | -54 | -72 | -90 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | 1 | 8 | 12 | 16 | 20 | 24 | 20 | 16 | 12 | 8 | 1 |
| $\theta_z(degree)$ | 0 | 18 | 36 | 54 | 72 | 90 | 108 | 126 | 144 | 162 | 180 |

Table 2-1: Training database configuration

with respect to the positive x-, y-, and z-axis, we have

$$\begin{aligned} x &= \cos\theta_x \\ y &= \cos\theta_y \\ z &= \cos\theta_z \end{aligned} \tag{2.28}$$

From the geometrical relationship in Figure 2-9, we have

$$\vec{\theta} = [\theta_x \quad \theta_y \quad \theta_z] = [2\pi \pm \cos^{-1}(\cos\alpha\ \cos\beta) \quad 2\pi \pm \cos^{-1}(\cos\alpha\ \sin\beta) \quad \frac{\pi}{2} - \alpha]$$

which is the pose information vector to be estimated.

The convergence of the network is measured by the root-mean-square (RMS) error value, computed by the equation:

$$RMS = \sqrt{\frac{\sum_n \parallel \vec{t^n} - \vec{I^n} \parallel^2}{n_p n_o}} \tag{2.29}$$

where $n_p$ is the number of patterns in the training set; $n_o$ is the number of nodes in the output layer; and $\parallel \cdot \parallel$ is the Euclidean norm. We consider the network as trained when the *RMS* error is less than 1 percent.

### 2.7.2 Performance Analysis

When the network is trained, the testing samples are fed into the proposed system. The performance is evaluated by the percentage correctness. The output is claimed correct if the object is identified and the error of the estimated angle is within $5^\circ$. The results are shown in Table 2-2 and Table 2-3 for airplane and machine parts, respectively. The

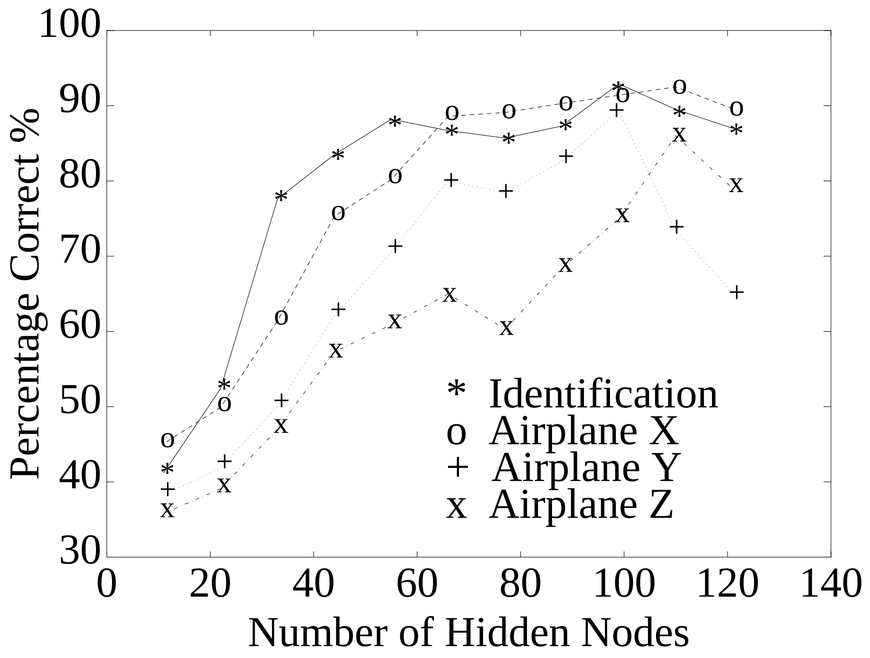



Figure 2-10: Simulation results for airplanes.

relationship between the number of hidden nodes used and the percentage correctness are plotted in Figure 2-10 and Figure 2-11 for airplane and machine part simulations. In the airplane simulation, the best results occur when the number of hidden nodes are 99, 110, 99, and 110 for the identification, airplane $X$ estimation, airplane $Y$ estimation, and airplane $Z$ estimation network, respectively; while in the machine parts simulation, they are 33, 99, 99, and 66 for the identification, depth stop estimation, column base estimation, and wedge lift estimation network, respectively.

Based on the proposed architecture, the training cycles used in PEN is $c$-1 times more than those used in OIN (in our simulations, $c = 3$) to deal with the analog outputs of PENs. The overall training cycles are

| number of hidden nodes | identification (%) | plane X P.E. (%) | plane Y P.E. (%) | plane Z P.E. (%) |
|---|---|---|---|---|
| 11 | 41.2 | 45.1 | 38.2 | 35.8 |
| 22 | 52.4 | 49.9 | 41.9 | 39.1 |
| 33 | 77.6 | 61.4 | 50.0 | 47.0 |
| 44 | 83.5 | 75.3 | 62.1 | 57.3 |
| 55 | 88.2 | 80.2 | 70.5 | 60.9 |
| 66 | 86.7 | 88.6 | 80.0 | 65.0 |
| 77 | 85.7 | 89.1 | 78.4 | 60.3 |
| 88 | 87.3 | 90.3 | 82.6 | 68.4 |
| 99 | 92.9 | 91.4 | 89.3 | 75.0 |
| 110 | 89.5 | 92.5 | 73.5 | 86.0 |
| 121 | 87.0 | 89.6 | 65.1 | 79.0 |

Table 2-2: Simulation results for airplanes.

| number of hidden nodes | identification (%) | depth stop P.E. (%) | column base P.E. (%) | wedge lift P.E. (%) |
|---|---|---|---|---|
| 11 | 51.3 | 40.2 | 42.5 | 47.3 |
| 22 | 62.3 | 45.3 | 55.2 | 55.2 |
| 33 | 97.6 | 55.7 | 67.1 | 64.3 |
| 44 | 94.9 | 62.1 | 77.3 | 75.3 |
| 55 | 95.7 | 73.5 | 85.9 | 87.5 |
| 66 | 94.8 | 84.3 | 97.5 | 98.7 |
| 77 | 92.5 | 85.5 | 95.1 | 97.4 |
| 88 | 93.5 | 87.3 | 96.3 | 85.7 |
| 99 | 92.9 | 89.9 | 98.7 | 90.1 |
| 110 | 87.1 | 73.1 | 94.3 | 71.4 |
| 121 | 85.9 | 73.5 | 93.2 | 72.5 |

Table 2-3: Simulation results for machine parts.

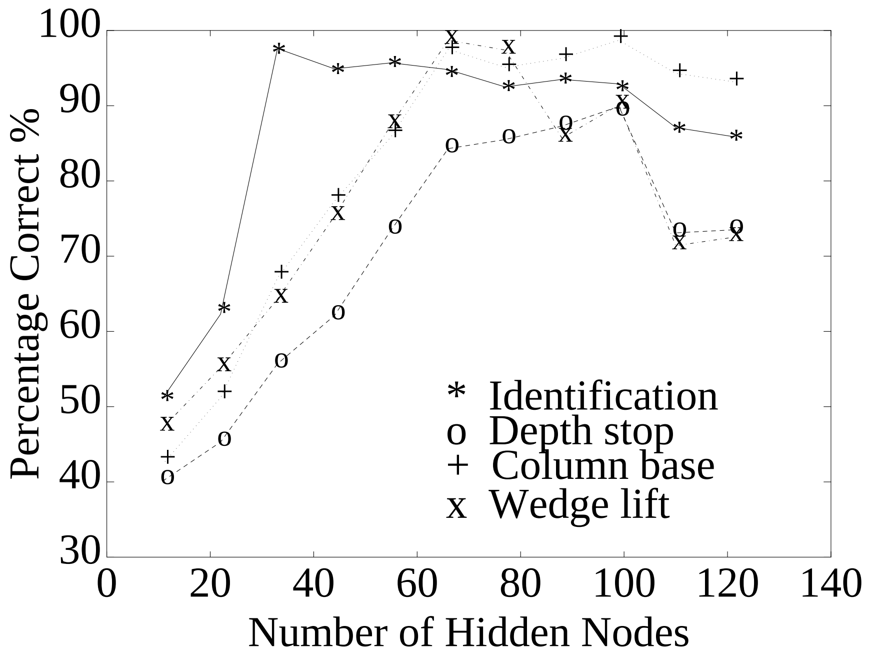



Figure 2-11: Simulation results for machine parts.

25,000 and 12,000 for the results shown in Figure 2-10 and Figure 2-11, respectively. Note that, the training cycles used for the first simulation are about twice more than those used for the second one. This is expected because of the similarity between the airplanes, we will need a much more complex network and use more training cycles to get a better result. Besides, the time complexity of the calculation of the moment invariants is $O(N^2p^2)$ for $N \times N$ image and an order $p$ moment set. This large amount of computation time can be greatly reduced by using the parallel architecture of the multi-layer networks.

## 2.8 Conclusion

In this Chapter, we have shown that *3-D* moment invariants are useful to encode *3-D* information. The moment feature extraction does not require the sophisticated range image segmentation. The large computation of the moment invariants can be done by five simple neuron layers. The network complexity is traded for better results and greater flexibility. Even though the off-line training of the network is time consuming, the identification of the unknown object and its orientation can be obtained simultaneously when the network is trained.

# Chapter 3

# Image Sensing Model for CCD Camera Systems

## 3.1 Introduction

Verification of many computer vision theories require expensive and accurate camera systems, and laboratory facilities for calibration and experimentation. As an alternative, it is possible to develop computational models of the camera system, and simulate the system on a computer. This is not only faster and cheaper than building actual camera systems and setting up expensive laboratories, it also provides flexibility and accuracy. The physical parameters of the camera system (*e.g.* focal length, sampling rate, quantization level, noise characteristics, optical aberrations, etc.) can be easily changed and they can be set to desired values to very high accuracy.

The motivation for this work arose from our need for the experimental verification of a set of new theories on measuring object distances using image defocus information (depth from defocus) [44, 48, 49]. The experiments required a precision camera system whose parameters could be controlled and measured accurately. We have used the computer model and the simulation system described in this and the following Chapters to synthesize experimental data and validate the new methods of measuring distance.

The camera model in this Chapter is derived from the mathematical model presented by Subbarao and Nikzad [47]. Their mathematical model has been modified and extended to obtain a *computational model*. Further, we have implemented the computational model and demonstrated its practical application in research on two problems in machine vision. Our implementation is nearly optimal in terms of the computational resources. One reason for this is that the model decouples the photometric properties of a scene from the geometric properties of the scene in the input to the camera system.

A camera system (or any physical system) can be modeled at many levels of detail and abstraction. Here we have attempted to focus on developing what we believe is a *useful* model rather than a *detailed* model. Our model involves seven major stages. Each of these stages can be extended (or condensed) to make it more (or less) detailed than what we have presented. For example, the stage involving the CCD sensor can be made more detailed using the CCD model presented by Healey and Kondepudy [15].

We also extended the proposed computational model to simulate the image formation of moving objects (motion) and stereo vision system. All the models presented in this Chapter are implemented and described in the following two Chapters – the Image Defocus Simulator (IDS) simulates the image formation process, while the Active Vision System (AVS) simulates the moving objects and stereo imaging.

This Chapter is organized as follows: Section 2 presents the computational model for a CCD camera system; Section 3 extends the computational model to motion and stereo simulation; and finally, Section 4 concludes this Chapter with a few remarks.

## 3.2 Computational Model

A block diagram of a typical CCD video camera system used in machine vision applications is shown in Figure 3-1. Inside the optical system stage, there are in general many lenses and apertures. However, we are concerned only with the terminal properties of the aggregate [14]. The terminals of this stage are an *entrance pupil* (effective or real) representing a finite aperture through which the light must pass to reach the imaging elements and an *exit pupil* (again effective or real) representing a finite aperture through which light must pass as it leaves the imaging element on its way to the image detector plane. It is also assumed that

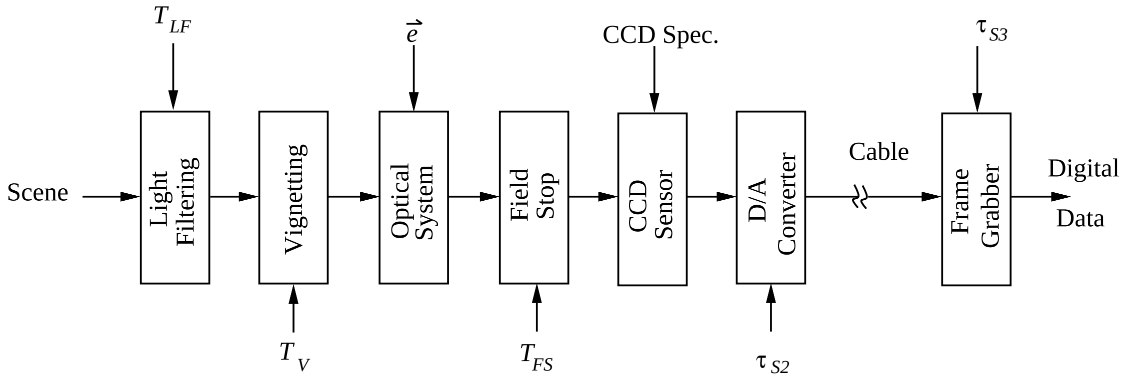



Figure 3-1: Block diagram of a video camera system.

the passage of light between the entrance and exit pupil planes can be adequately described by geometric optics [16].

We define a spherical coordinate system with its $z$-axis along the optical axis and the origin fixed in the plane of the entrance pupil as shown in Figure 3-2 (called Entrance Pupil Coordinate System, or EPCS for short). All the distance measurements are made with respect to $O$. The *external parameters* of the camera system are the position (three parameters) and orientation (three parameters) of the EPCS with respect to some reference coordinate system in the scene.

We consider the scene to contain only objects with opaque surfaces. In such case, for every direction defined by $(\theta, \phi)$ in the EPCS, there is a *unique* point $P$ on a visible object in the scene as shown in Figure 3-2. If $r$ is the distance from $P$ to the origin $O$, then the geometry of visible surfaces in the scene can be completely defined by a function $r(\theta, \phi)$ which gives *geometric information* about visible surfaces in the scene.

What the camera "observes" due to the point source $P$ is the electro-

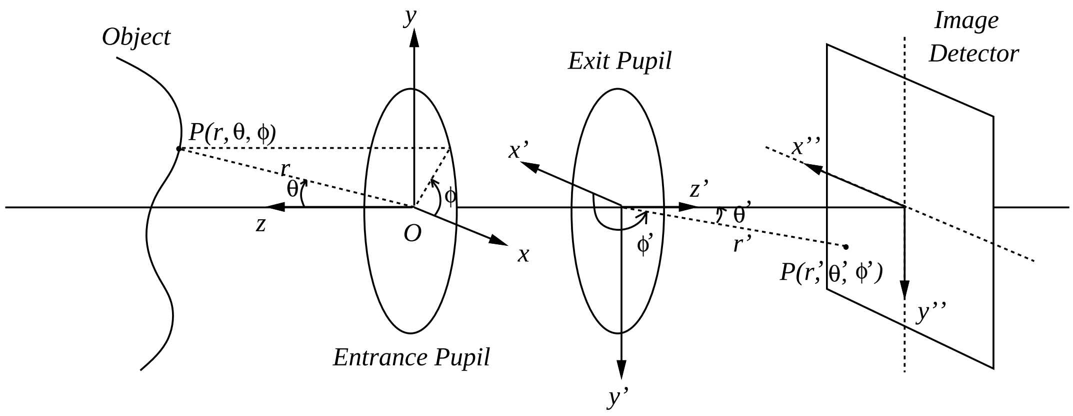



Figure 3-2: Entrance pupil coordinate system.

magnetic field distribution produced by the source at the entrance pupil. This field distribution can be specified by a very general form; however, we shall restrict ourselves to a simpler case. We will only consider incoherent and unpolarized light. The quantities we are interested are the radius $r$ of the wavefront incident on the entrance pupil and the energy strength of the wavefront as a function of its wavelength $\lambda$. These quantities are assumed to remain the same during one exposure period of the camera.

We define the *focused image* $f(\theta, \phi, \lambda, t)$ of the scene to be the power of light of wavelength $\lambda$ incident on the entrance pupil from the direction $(\theta, \phi)$ at time $t$. The function $f(\cdot)$ encodes purely *photometric* information. We define the *inputs* to the camera system to be $r(\theta, \phi)$ which describes the geometry of the scene, and the focused image $f(\theta, \phi, \lambda, t)$ which describes the appearance of the scene. These two functions are quantities which are directly observable by the camera system. There is no interdependence between these two quantities. This should be compared to the traditional convention of characterizing the photometric aspects of the scene in terms of scene radiance. The radiance of a small surface patch in the scene is the light power emitted by the surface patch into a unit solid angle per unit area, and has units of *Watts*$/m^2/$*Steradian*. Since the area of a surface cannot be estimated without a knowledge of the distance $r$ and orientation of the surface patch with respect to the camera system, the radiance of the surface patch cannot be observed by the camera system independently. Therefore, characterization of scene appearance in terms of the radiance of surface patches in the scene invariably *couples* the photometric and geometric properties. In contrast, our characterization of scene properties in terms of $f(\theta, \phi, \lambda, t)$ and $r(\theta, \phi)$ decouples the photometric and geometric properties.

Having defined the input to the camera, we will now define a sequence of transformations which transforms the input signal to the output digital image. The blocks in Figure 3-1 are numbered from left to right as stage $i$ for $i = 1, 2, 3, ..., 7$. The output of the $i$th stage is denoted by a function of the form $f_i$. Each stage $i$ may have one or more steps denoted by $f_{ij}$ for $j = a, b, c, ...$. Each step in the sequence of transformations typically corresponds to the effect of one component of the camera system on the input signal.

(TR.1) *Light Filtering:* Light filter is used to control the spectral content of light entering the camera system. Its characteristic can be specified by a transmittance function $T_{LF}(\lambda)$ where $0 \leq T_{LF}(\lambda) \leq 1.0$. The output of this stage is:

$$f_1(\theta, \phi, \lambda, t) = f(\theta, \phi, \lambda, t) \cdot T_{LF}(\lambda) \tag{3.1}$$

(TR.2) *Vignetting:* When there are multiple apertures in the optical system along the optical axis displaced with respect to each other, the effective light energy transmitted by the system decreases with increasing inclination of light rays with respect to the optical axis [16, 19]. This effect can be specified by a vignetting function $T_V(\theta, \phi)$ where $0 \leq T_V(\theta, \phi) \leq 1.0$. The output is:

$$f_2(\theta, \phi, \lambda, t) = f_1(\theta, \phi, \lambda, t) \cdot T_V(\theta, \phi) \tag{3.2}$$

(TR.3) *Optical System:* An image forming optical system can be characterized in terms of the image (or light energy distribution on the image detector) produced by the system when the scene contains a single point light source. The image of a point light source corresponds to the *point spread function* (PSF) of the camera system (when the light energy incident from the point source onto the entrance pupil is one unit). The scene can be considered to be an aggregate of point light sources each corresponding to one point on the visible surfaces in the scene. When the point sources in the scene are *incoherent*, the light intensity distribution produced on the image detector by each of the point sources can be simply summed to obtain the overall image. (If the point sources are coherent, then the electromagnetic field distribution will have to be summed instead of their intensities; we shall not consider this case here.) In this case, the imaging system acts as a linear system with its characteristics specified by a PSF $h(\theta, \phi, \theta', \phi', r(\theta, \phi), \vec{e})$ where (see Figure 3-2) $(\theta, \phi)$ is the direction of the point source $P$, $(\theta', \phi')$ is the direction of a point on the image detector in the image plane, and $\vec{e}$ is a vector specifying the parameters of the imaging system such as its focal length, aperture diameter, etc.

After standard transformation from spherical to a normalized Cartesian coordinate system (taking $z = z' = 1$), $f_2(\theta, \phi, \lambda, t)$ and $h(\theta, \phi, \theta', \phi', r(\theta, \phi), \vec{e})$ can be equivalently represented as $f_2'(x, y, \lambda, t)$ and $h'(x, y, x', y', r(x, y), \vec{e})$, respectively. Hence, the output is:

$$f_3(x', y', \lambda, t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h'(x, y, x', y', r(x, y), \vec{e}) \times f_2'(x, y, \lambda, t)\, dx\, dy \tag{3.3}$$

If the PSF is spatially invariant in the region of interest (*i.e.* isoplanetic region [4]), then the above integral becomes a convolution operation:

$$f_3(x', y', \lambda, t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h'(x'-x, y'-y, r(x,y), \vec{e}) \times$$
$$f_2'(x, y, \lambda, t)\, dx\, dy \qquad (3.4)$$

(TR.4) *Field Stop:* The extent of the image detector determines the field of view of the imaging system which can be specified by the transmittance function of the field stop $T_{FS}(x', y')$ where $T_{FS}(x', y')$ has a value of 1 inside the image detector region and a value of 0 outside. Therefore, the output will be:

$$f_4(x', y', \lambda, t) = f_3(x', y', \lambda, t)\; T_{FS}(x', y') \qquad (3.5)$$

(TR.5) *CCD Sensor:* The transducer on the image detector which converts light energy to electrical energy is not uniformly sensitive with respect to wavelength $\lambda$. To take this effect into account, we model the sensitivity by $T_s(\lambda)$ with $0 \leq T_s(\lambda) \leq 1.0$. This will transform $f_4$ to:

$$f_{5a}(x', y', \lambda, t) = f_4(x', y', \lambda, t)\; T_s(\lambda) \qquad (3.6)$$

Since the output of the photosensor on the image detector depends on the total light energy incident on the detectors, the light energy has to be integrated with respect to the wavelength $\lambda$, *i.e.*

$$f_{5b}(x', y', t) = \int_{-\infty}^{\infty} f_{5a}(x', y', \lambda, t)\, d\lambda \qquad (3.7)$$

The image sensor is exposed to incident light for a finite duration of time. During the period when the sensor is exposed, the strength of the incident light may vary because of the changing area of the aperture stop with time $T_{AS}(t)$. In a CCD camera, this is equivalent to measuring, periodically, the charge collected by the CCD elements and then clearing the charge in the elements. The effect of the exposure function is:

$$f_{5c}(x', y', t) = \int_{-\infty}^{\infty} f_{5b}(x', y', \tau)\, T_{AS}(\tau - t)\, d\tau \qquad (3.8)$$

If $T_{AS}(t)$ is symmetric, then equation (3.8) becomes a convolution operation. Next, we take into account the physical shape and size of the photosensor elements (pixels) and the sensor noise. Let $R(x, y)$ be a function whose value is 1 inside the surface area of a photosensor element and 0 outside, and $n_s(x', y', t)$ be the CCD sensor noise. We model each sensor element as producing an output proportional to the total light energy incident on its surface. Noise is then added to this output. The following expression results:

$$\begin{aligned} f_{5d}(x', y', t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} & f_{5c}(\alpha, \beta, t)\, R(\alpha - x', \beta - y')\, d\alpha\, d\beta \\ & + n_s(x', y', t) \end{aligned} \qquad (3.9)$$

If $R(x, y)$ is symmetric, i.e. $R(x, y) = R(-x, -y)$, then the integration in equation (3.9) is a convolution.

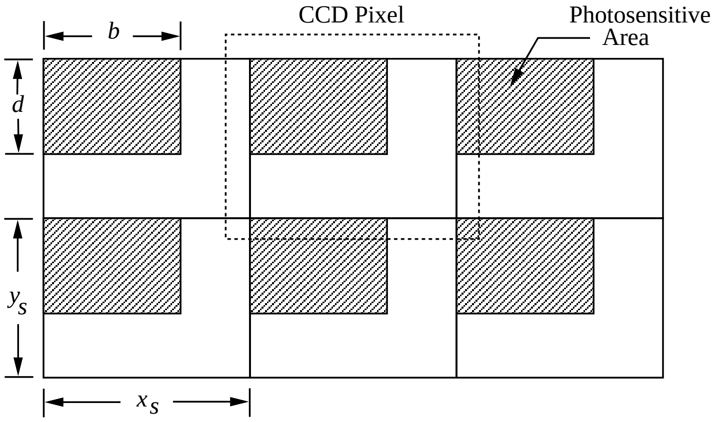



Figure 3-3: Typical geometry of pixels on a CCD image detector.

We will assume that the continuous signal $f_{5d}(x', y', t)$ is sampled in time periodically at fixed intervals of $\tau_{s1}$ and sampled in space on a discrete rectangular grid of points separated by a distance of $x_s$ along the horizontal direction and by a distance of $y_s$ along the vertical direction as shown in Figure 3-3. The resulting sampled output will be:

$$f_{5e}(x', y', t) = f_{5d}(x', y', t)\,\frac{1}{|\tau_{s1}||x_s||y_s|}\,\mathrm{comb}\left(\frac{x'}{x_s}, \frac{y'}{y_s}, \frac{t}{\tau_{s1}}\right) \tag{3.10}$$

which is converted to a discrete function, represented as a three-dimensional matrix $f_{5f}[i, j, k]$, as:

$$f_{5f}[i, j, k] = f_{5e}(i \cdot x_s,\ j \cdot y_s,\ k \cdot \tau_{s1}) \tag{3.11}$$

for $i = 0, 1, \cdots, M-1$; $j = 0, 1, \cdots, N-1$; $k = 0, 1, \cdots, K-1$ where $M$, $N$, and $K$ are the number of columns, rows, and image frames, respectively.

Ideally, we require the output of the photosensitive elements to be proportional to the light energy incident on them. However, in practice, their output is not proportional but some other function of light energy $S(I)$. Therefore, the output will be

$$f_{5g}[i,j,k] = S(f_{5f}[i,j,k]) \tag{3.12}$$

If the sensor response function is different for different photosensitive elements indexed by $(i,j)$, then $S(I)$ will become $S(i,j,I)$. Several models have been proposed for $S(I)$ in the literature such as the *gamma model, linear model,* etc.

Finally, the three-dimensional array is converted to one-dimensional sequence of numbers which can be converted to analog signals and transmitted over a cable. The resulting output is:

$$f_5[i + j \cdot M + k \cdot M \cdot N] = f_{5g}[i,j,k] \tag{3.13}$$

where we assume, for simplicity, that all rows are scanned in sequence one by one and that all the synchronization pulses for video monitor are ignored.

(TR.6) *D/A Converter:* The discrete sequence of numbers represented by $f_5[i]$ is converted to an analog signal using a sample-and-hold circuit. The time interval $\tau_{s2}$ between two numbers is an input parameter to this stage. Note that $\tau_{s2} \leq \frac{\tau_{s1}}{M \cdot N}$. The operation of the reconstruction circuit can be thought of as interpolating the sequence of values $f_5[i]$ defined at $i \cdot \tau_{s2}$ to get an analog signal:

$$f_{6a}(t) = h_{sh}(t) \star \left[ \sum_{l=0}^{K \cdot M \cdot N - 1} f_5[l] \; \delta(t \; - \; l \cdot \tau_{s2}) \right] \qquad (3.14)$$

where $h_{sh}(t)$ is the effective impulse response of the sample-and-hold interpolation circuit and $\star$ is the convolution operator.

Finally, the video signal $f_{6a}(t)$ is amplified before it is transmitted over a video cable so that any noise introduced in the cable will not dominate the signal. The resulting output signal is:

$$f_6(t) = f_{6a}(t) \star h_a(t) + n_a(t) \qquad (3.15)$$

where $h_a(t)$ and $n_a(t)$ are the impulse response and the additive noise of the amplifier, respectively.

(TR.7) *Frame Grabber:* The input signal to this stage is

$$f_{7a}(t) = f_6(t) \star h_c(t) \; + \; n_c(t) \qquad (3.16)$$

where $h_c(t)$ and $n_c(t)$ are the impulse response and additive noise of the video cable, respectively. This analog signal is sampled at intervals of $\tau_{s3}$ by multiplying it with $\sum_{n=-\infty}^{\infty} \delta(t - n\tau_{s3})$. If $\tau_{s3} \neq \tau_{s2}$, this will cause geometric distortion of the picture. This phenomenon has been called *mismatched electronics* in Shafer [42]. The sampled values are interpolated by an $n$-th order (usually $n = 0$) sample-and-hold filter. The resulting signal is again sampled by a slightly shifted sampling function to get an impulse train which is converted to a sequence of numbers as follows:

$$
\begin{aligned}
f_{7b}(t) &= \left\{ \left[ \sum_{n=-\infty}^{\infty} \delta(t - n\tau_{s3}) \right] \cdot f_{7a}(t) \right\} \star h_{sh}(t) \\
&\quad + n_{sh}(t) && (3.17) \\
f_{7c}(t) &= \frac{1}{\tau_{s3}} \mathrm{comb} \left( \frac{t - k\tau_{s3}}{\tau_{s3}} \right) \cdot f_{7b}(t) && (3.18) \\
f_{7d}[i] &= \int_{(i+k)\tau_{s3}^{-}}^{(i+k)\tau_{s3}^{+}} f_{7c}(t)\, dt && (3.19)
\end{aligned}
$$

where $h_{sh}(t)$ and $n_{sh}(t)$ are the impulse response and additive noise of the sample-and-hold filter, respectively; $0 < k < 1$; and the following equation is used in the derivation.

$$
\sum_{n=-\infty}^{\infty} \delta(t - n\tau) = \frac{1}{\tau} \mathrm{comb} \left( \frac{t}{\tau} \right) \qquad (3.20)
$$

Finally, the sequence of numbers are quantized and sent to frame buffer for further processing.

## 3.3 Extensions to Motion and Stereo Simulation

In this section, we will extend the computational model presented in the previous section to simulate the image sensing process for moving objects and binocular stereo camera systems.

### 3.3.1 Motion Simulation

When objects move in front of a camera, or when a camera moves through a fixed environment, there are corresponding changes in the images. The displacement of a point in the environment will cause a displacement of the corresponding image point. In motion simulation, we assume that all the objects in the scene are rigid objects. Therefore, the shape of the objects will not change during motion.

Figure 3-4 shows the relationship between an object motion vector $\vec{m_o} = P_o\vec{P}'_o = [V_{x0} \quad V_{y0} \quad V_{z0} \quad \Delta t]$ and the image motion vector $\vec{m_i} = P_i\vec{P}'_i$. For simplifying the discussion, the image plane is placed at the focused position and is perpendicular to the optical axis ($z$-axis). The vector $\vec{m_o}$ can be decomposed into two components, one parallel to the x-y plane ($P_o\vec{P}''_o$) which shifts the object, and another parallel to the $z$-axis which changes the size of the object.

Consider the translation vector $P_o\vec{P}''_o$. Let $P_o\vec{P}''_o = [\vec{V_t} \quad \Delta t] = [V_{xo} \quad V_{yo} \quad 0 \quad \Delta t]$ for a fixed time interval $\Delta t$. This corresponds to a motion vector $\vec{m_i} = [\vec{V_i} \quad \Delta t] = [V_{xi} \quad V_{yi} \quad 0 \quad \Delta t]$ in the image plane. The amount of displacement is $||P_o\vec{P}''_o|| = ||\vec{V_t}\Delta t||$ in the scene which corresponds to a displacement of $||P_i\vec{P}'_i|| = ||\vec{V_i}\Delta t||$ in the image plane.

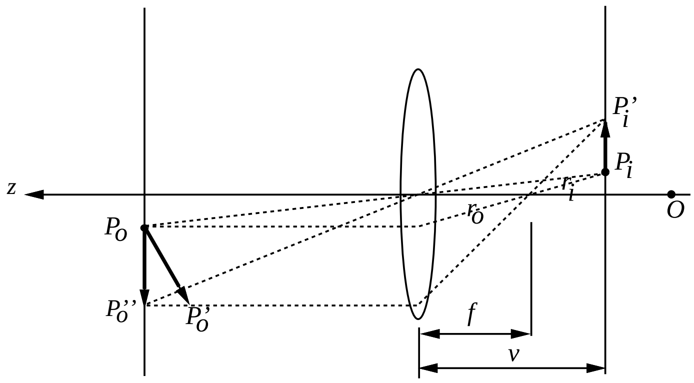



Figure 3-4: Relationship between the displacement of a point in the scene and the corresponding point in the image plane.

From the geometry in Figure 3-4, we have

$$\frac{||\vec{V}_i \Delta t||}{||\vec{V}_t \Delta t||} = \frac{r_i}{r_o} = \frac{v - f}{f} \tag{3.21}$$

The displacement of points in the image plane can be computed using Equation (3.21). For $z$-axis movement, *i.e.* $V_{z0} \neq 0$, there will be a change in the size of the objects in the scene. This results in image magnification or shrinking which requires image interpolation and resampling.

### 3.3.2 Stereo Vision System

A general stereo system model is shown in Figure 3-5 where $O$ is the global origin, $O'$ and $O''$ are the entrance pupil origin of the left and the right cameras, respectively. The left and the right cameras can be treated as monocular camera systems similar to that in Figure 3-2. The

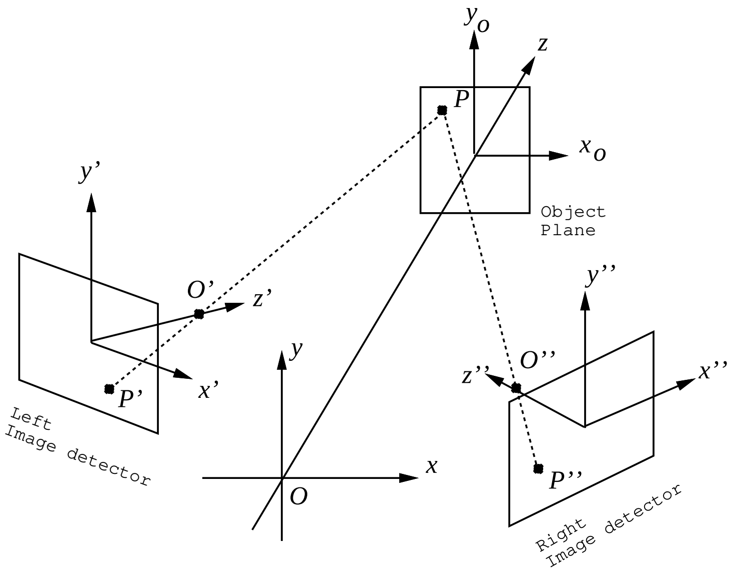



Figure 3-5: A general stereo system model.

global origin $O$ is introduced as a reference point for the positions of the object, the left camera, and the right camera.

In this generalized stereo system, the optical axes of the two cameras are not parallel, but intersect at some point in the scene. In order to simplify computations in our simulation, we restrict the optical axes of the cameras to be parallel. Therefore, in this model, there is a relative translation between the two cameras, but no relative rotation. This restriction can be removed at the expense of more computation.

Figure 3-6 is the global coordinate system used in our current stereo simulation where $z$-, $z'$-, and $z''$-axes are parallel to each other. Based on this configuration, the stereo vision system can be modeled as shown in Figure 3-7. The scene information is first translated, and scaled with respect to the origin of each camera. After this transformation, the

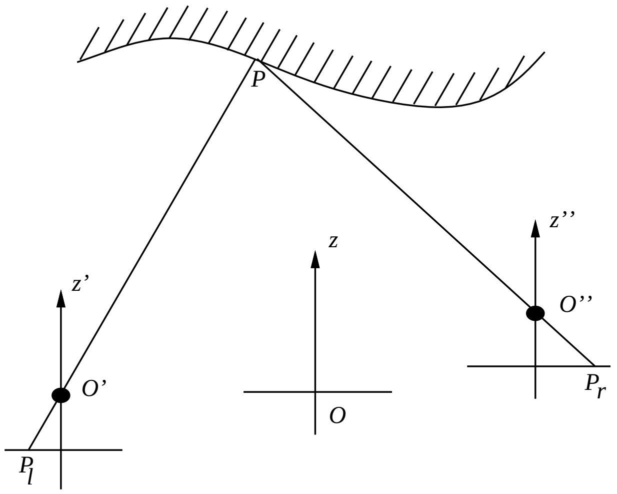



Figure 3-6: Global coordinate system used in stereo simulation.

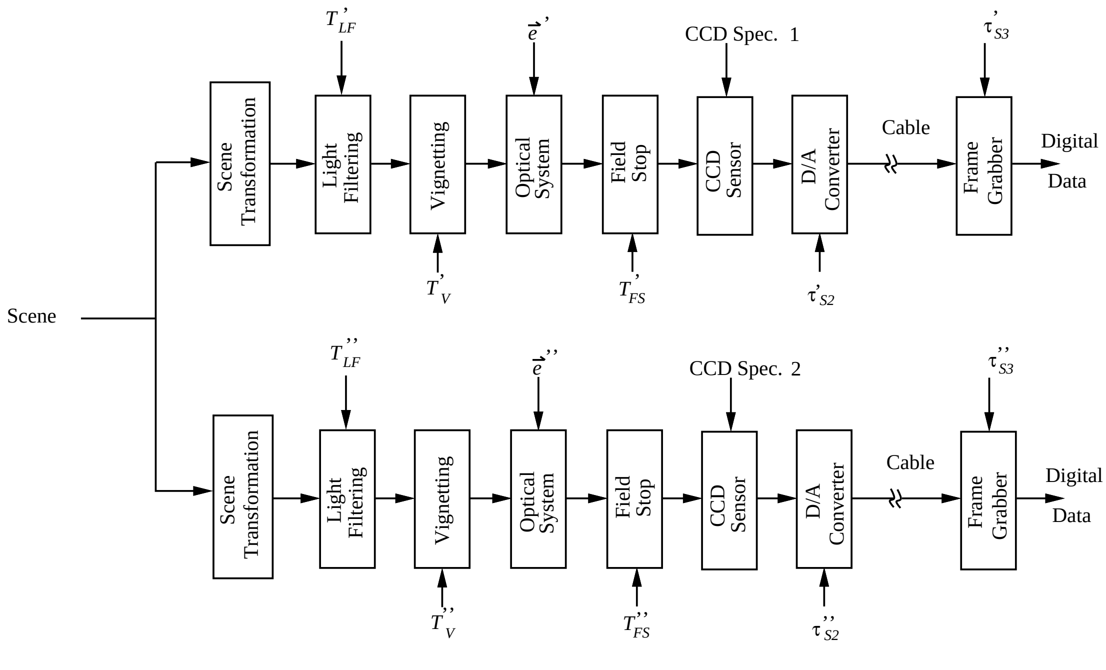



Figure 3-7: Block diagram of a stereo vision system.

photometric information $f(\theta, \phi, \lambda, t)$ and the geometric/depth information $r(\theta, \phi)$ are transformed to $f_l(\theta, \phi, \lambda, t)$, $f_r(\theta, \phi, \lambda, t)$ and $r_l(\theta, \phi)$, $r_r(\theta, \phi)$ for the left and the right cameras. These functions are the input to the camera system. The remaining functional blocks are the same as those presented in Section 3.2.

## 3.4 Conclusion

A computational model has been developed for a CCD camera used in computer vision applications. This model consists of 7 functional blocks, namely, light filtering, vignetting, optical system, field stop, CCD sensor, D/A converter, and the frame grabber. Each functional block is refined to one or more computational steps toward the image sensing process. There are 27 user controllable parameters in these 16 computational steps for single camera model. With the addition of the motion information parameters and the positional parameters of two cameras, the proposed model is extended to cover the simulation of moving objects and the stereo imaging system.

The model presented in this Chapter decouples the geometric and photometric information of the scene. Each functional block in this model is modularized and independent. Therefore, it is computationally tractable and can be easily extended and/or modified for other imaging applications.

# Chapter 4

# The IDS Computer Simulation System

## 4.1 Introduction

Based on the model presented in the previous chapter, a computer simulation system called Image Defocus Simulator (IDS) has been developed. It can simulate a typical CCD camera system used in machine vision applications.

IDS is a menu-driven simulation system which takes as input the camera parameters and scene information. The output of IDS is a digital image of the scene as sensed by the camera. IDS consists of a number of distinct software modules each implementing one step in the computational model. The modules are independent and can be easily modified to enhance the simulation system. From the functional point of view, IDS consists of a kernel engine and three user interfaces, namely, Sunview Graphical Interface (SGI), X window Graphical Interface (XGI), and Dummy TTY Interface (DTI). The simulation engine is a machine-independent module to carry out all the computations involved while the user interfaces are used to provide a menu- or command-driven I/O interface. The kernel engine and the DTI are machine-independent and hence portable. The SGI and XGI can be easily ported to SUN workstations and any machine with standard X11/R4 or newer distribution, respectively. Therefore it is very easy to extend the model and the simulation system to cover different types of imaging systems. A detailed description of the user interfaces and the reference manual of IDS can be found in [31, 46].

This Chapter is organized as follows: Section 2 presents the simulation engine of IDS; Section 3 describes the user interfaces; Section 4 presents the simulation result using both geometric optics and wave-optics; Section 5 describes the applications of IDS to the verification of the implementations of a theory of depth-from-defocus; and finally, Section 6 summarizes this Chapter.

## 4.2 The Simulation Engine

Consider the scene parameters $f(\theta, \phi, \lambda, t)$ and $r(\theta, \phi)$. If the profile of the scene in a small field-of-view is smooth, then we can approximate $r(\theta, \phi)$ by a constant $u$ which specifies the distance between the scene and the origin $O$ in the EPCS. Under these circumstances, the point spread

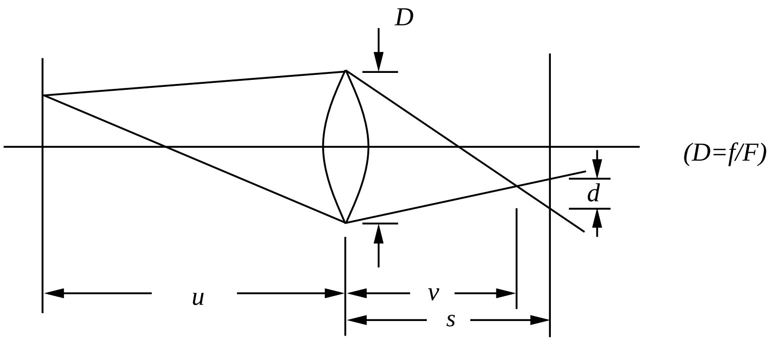



Figure 4-1: Blur circle calculation.

function will be spatially invariant. If geometric optics is assumed, then the diameter of the blur circle can be computed using the lens equation $1/f = 1/u + 1/v$ and Figure 4-1. The resulting diameter of the blur circle $d$ is:

$$d_m = \frac{f}{v \cdot F}|s - v| \tag{4.1}$$

$$d_p = \frac{d_m}{\rho} \tag{4.2}$$

where $f$ is the effective focal length; $F$ is the F-number (defined as $f/D$); $d_m$ is the diameter of blur circle in millimeter; $\rho$ is the CCD pixel element size in millimeter; $d_p$ is the diameter of the blur circle in pixels; $v$ is the distance between the exit pupil and the plane where the object is focused; and $s$ is the distance between the exit pupil and the photosensor plane. The point spread function according to paraxial geometric optics is:

$$h(x, y) = \begin{cases} \frac{4}{\pi d_m^2} & \text{if } x^2 + y^2 \leq \frac{d_m^2}{4} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

If $r(\theta, \phi)$ cannot be approximated by a constant distance $u$, then we must calculate the blur circle for each point in the scene and sum them to synthesize the image. Sometimes the paraxial model of the point spread function is not a satisfactory approximation to the actual point spread function. In this case, the actual point spread function of the optical system is measured for various distances $u$ and pre-stored in a file. The IDS will then use this file in the image synthesizing process. The choice of the point spread function is determined by the parameter "psf" which can be "**psf=cylinder**" for geometric optics, "**psf=file** `filename` [**size**] `width height`" for pre-calibrated point spread function stored in the file named `filename` of size `width` by `height`, or "**psf=wave_optics**" for wave optics. Therefore, one can choose geometric optics as well as physical optics as the point spread function to synthesize images. Note that, we use **boldface** to represent keywords in parameter specifications and use `typewriter` font to represent user-specified input. Keywords enclosed by square brackets are optional.

The effect of light filtering and vignetting are specified by the functions $T_{LF}(\cdot)$ and $T_V(\cdot)$, respectively. They can be: (i) a constant, (ii) a Gaussian mask **gaussian**$(\sigma_x, \sigma_y)$, or (iii) a function tabulated and stored in a file. Based on this information, Equations (3.1)-(3.4) can be computed with the information $\vec{e} = (s, f, F)$.

The parameter $T_{FS}$ controls the field-of-view of the photosensor devices. For a CCD camera, the field stop is rectangular in shape with width $A$ and height $B$. Therefore,

$$T_{FS}(x,y) = \text{rect}\left(\frac{x}{A}, \frac{y}{B}\right) \tag{4.4}$$

This is implemented by restricting the calculations to a rectangular region as specified above.

In most cases, the exposure function can be approximated by a rectangular function $T_{AS}(t) = \text{rect}(\frac{t}{T})$ where $T$ is the duration of exposure which is typically $\frac{1}{30}$ second. If the object is not moving, then timing information can be discarded. Further, from the geometry of CCD pixels in Figure 3-3, we have $R(x,y) = \text{rect}(\frac{x}{b}, \frac{y}{d})$ for Equation (3.9). The sensor response function $S(I)$ can be either $I^{\gamma}$ (for standard NTSC TV, $\gamma \approx \frac{1}{2.2}$ [43]), $aI+b$ ($a$, $b$ are constants), or a table read from a file.

Combining all the above information and adding (i) the impulse response functions for sample-and-hold circuit (using $h_{sh}$), for amplifier (using $h_a$), and for cable connections (using $h_c$); (ii) the corresponding noise functions (using $n_{sh}$, $n_a$, and $n_c$); (iii) the CCD noise (using $n_s$); (iv) the sampling information (using $\tau_{s1}, \tau_{s2}, \tau_{s3}$); and (v) the CCD geometry (using $x_s, y_s$), we can carry out Equations (3.5)-(3.19) directly to complete the simulation.

All the parameters mentioned above can be changed by using "Edt Param" command in the user interface to be discussed later. Furthermore, to deal with the tremendous amount of data storage expected and the variable size of the resulting output images, a built-in dynamic memory manager is used to achieve the greatest flexibility.

As one can expect, the convolution operation is a critical part in the simulation. Consider the convolution of the $M \times N$ image $f[m,n]$ and the $P \times Q$ point spread function $h[p,q]$:

$$g[\alpha,\beta] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n]h[\alpha-m,\beta-n] \tag{4.5}$$

where $m = 0,1,\cdots,M-1$; $n = 0,1,\cdots,N-1$; $p = 0,1,\cdots,P-1$; $q = 0,1,\cdots,Q-1$; $\alpha = 0,1,\cdots,(M+P-2)$; and $\beta = 0,1,\cdots,(N+Q-2)$. This equation can be rewritten as:

$$g[\alpha,\beta] = \sum_{m=0}^{min\{\alpha,M-1\}} \sum_{n=0}^{min\{\beta,N-1\}} f[m,n]h[\alpha-m,\beta-n] \tag{4.6}$$

In this equation, we simply assume that $h[p,q] = 0$ for $p \geq P$, $p < 0$, $q \geq Q$, or $q < 0$. The computational complexity to carry out the convolution directly will be $O((M+P) \times (N+Q) \times M \times N)$ which is a huge number if the image size is large. On the contrary, if we take the $2$-$D$ Discrete Fourier Transform of $g[\alpha,\beta]$:

$$G[k,l] = \sum_{\alpha=0}^{K-1} \sum_{\beta=0}^{L-1} g[\alpha,\beta] W_K^{k\alpha} W_L^{l\beta} \tag{4.7}$$

as $K$ DFTs of the form:

$$A[\alpha,l] = \sum_{\beta=0}^{L-1} g[\alpha,\beta] W_L^{l\beta} \tag{4.8}$$

followed by $L$ DFTs of the form:

$$G[k,l] = \sum_{\alpha=0}^{K-1} A[\alpha,l] W_K^{k\alpha} \tag{4.9}$$

where $k = 0,1,\cdots,K-1$; $l = 0,1,\cdots,L-1$; $K = M+P-1$; and $L = N+Q-1$ [34], then the computational complexity for using FFT algorithm will be $O((M+P) \times (N+Q) \times \log_2[(M+P) \times (N+Q)])$. For small values of $P$, $Q$, it will be more efficient to use direct convolution. Therefore, we provide three options in computing convolutions: direct, FFT, and *smart* convolution. In "direct" mode, no matter how big (or small) the image/psf size is, direct convolution is carried out in the spatial domain. Similarly, in "FFT" mode, irrespective of image/psf size, convolution is carried out in the Fourier domain using FFT algorithm. In "smart" mode, the expected number of operations is calculated for direct and FFT modes, and the mode which requires lower number of operations is chosen. Equations (4.8) and (4.9) suggest a method for parallelization to speed up the computations.

## 4.3 The User Interfaces

IDS provides three user interfaces – Sunview Graphical Interface (SGI), X-window Graphical Interface (XGI), and Dummy TTY Interface (DTI). The only difference between SGI and XGI is that SGI runs under Sunview environment while XGI runs under X window environment. They both provide a graphical, menu-driven, and mouse-oriented interactive user interface. The DTI interface is designed for users to run their simulations on a dummy terminal. It provides all the functions of IDS except the capability of displaying images and menus. A typical screen for SGI, XGI, and DTI are shown in Figure 4-2, Figure 4-3, and

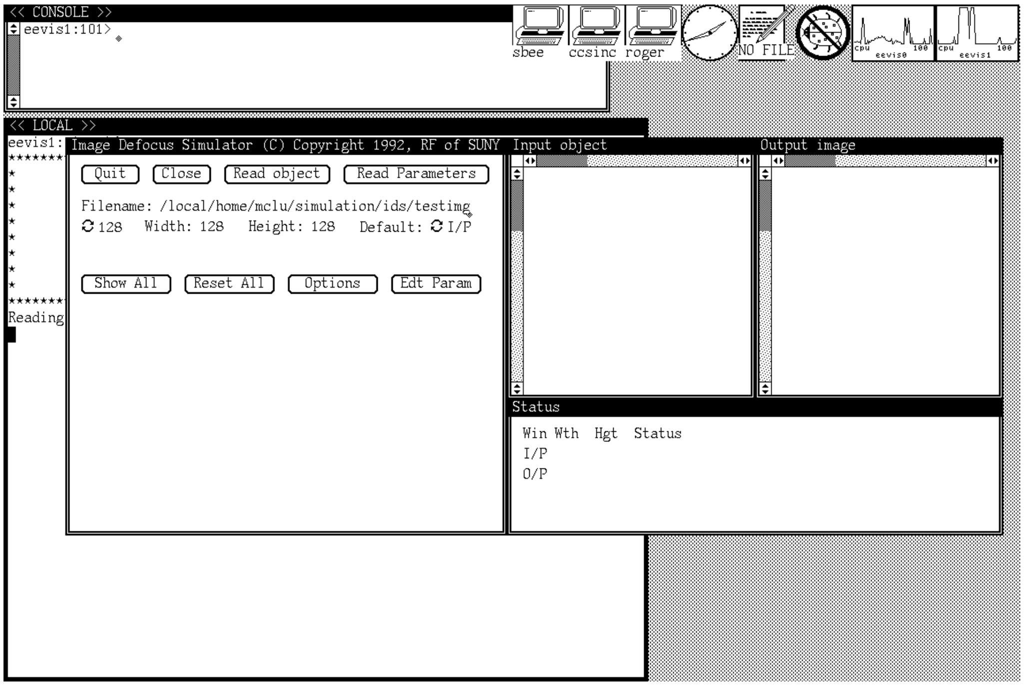



Figure 4-2: Startup screen of the simulation system (SGI user interface).

Figure 4-4, respectively. The SGI is used in the following illustrations.

Under SGI, users are provided with eight commands, three text-input fields ("Filename", "Width", and "Height"), and two choices ("Size" and "Default") as shown in Figure 4-2. All the I/O commands and the "Read Parameters" command take the string in the "Filename" field as the target filename. The image size is determined by the "Width" and "Height" fields which can be changed by either the "Size" choice (for square image) or the user input (for arbitrary size image). The command "Read Parameters" reads a file which contains all the user-controllable parameters. These parameters are parsed through a built-in LR(1) parser [1] to generate tokens and detect possible syntax errors.

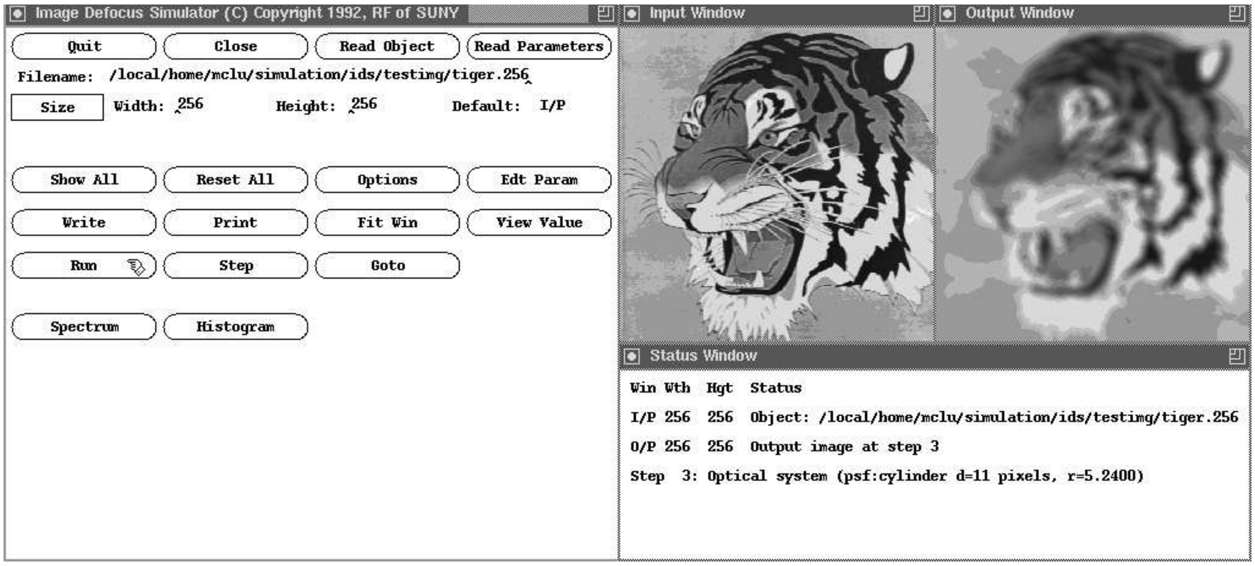



Figure 4-3: XGI user interface.

The parsing results are then interpreted by an interpreter to generate parameter values. These values can be modified by the "Edt Param" command which pops up a window as shown in Figure 4-5. The default parameter values can also be found in Figure 4-5.

The "Options" command controls the system-wide options such as input/output image format, convolution method, and how to handle the image border. The input/output image format can be binary/ASCII integers, binary/ASCII floating numbers, or even a $n$-th order polynomial (input only) specified by the order of the polynomial and the corresponding coefficients. The convolution method can be direct, FFT, or *smart* mode implementation as described in the previous section. The image

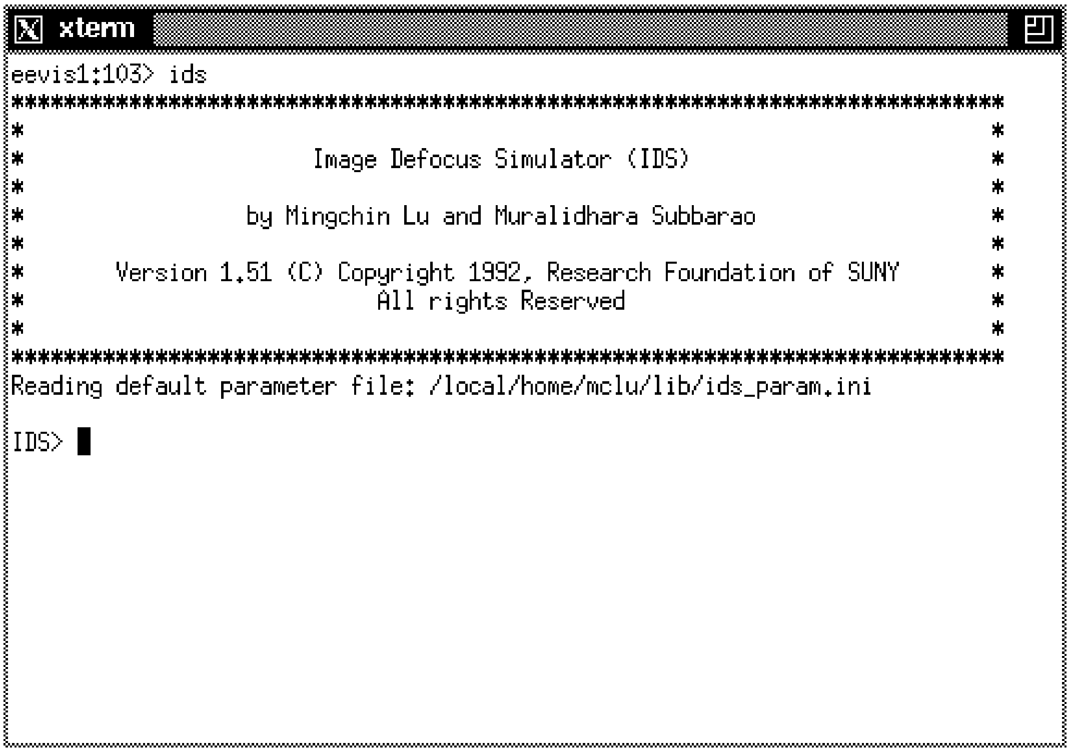



Figure 4-4: DTI user interface.

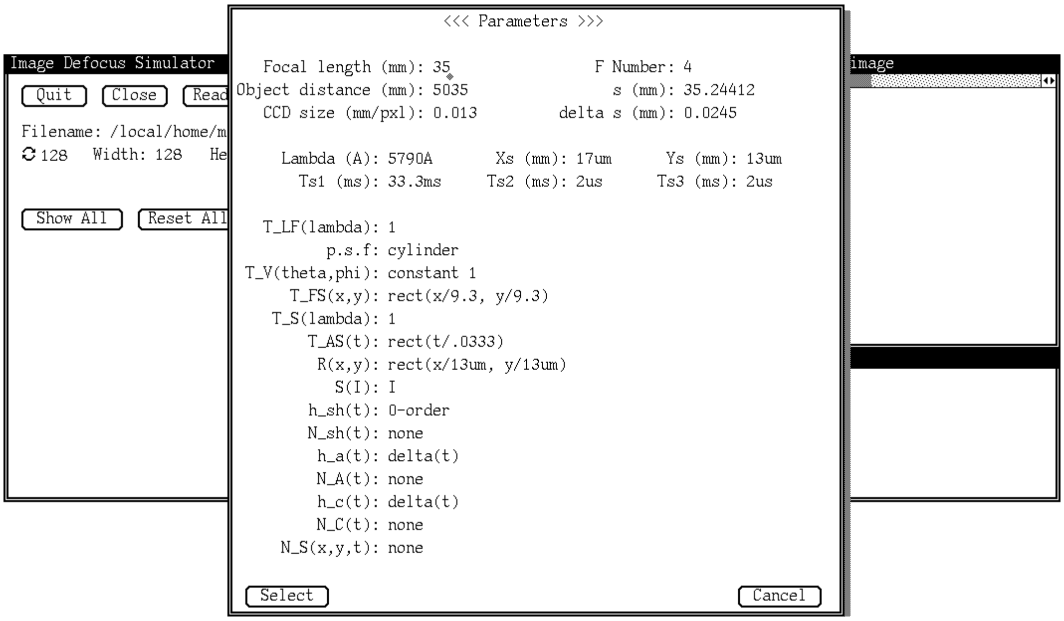



Figure 4-5: Popup window for "Edt Param" command.

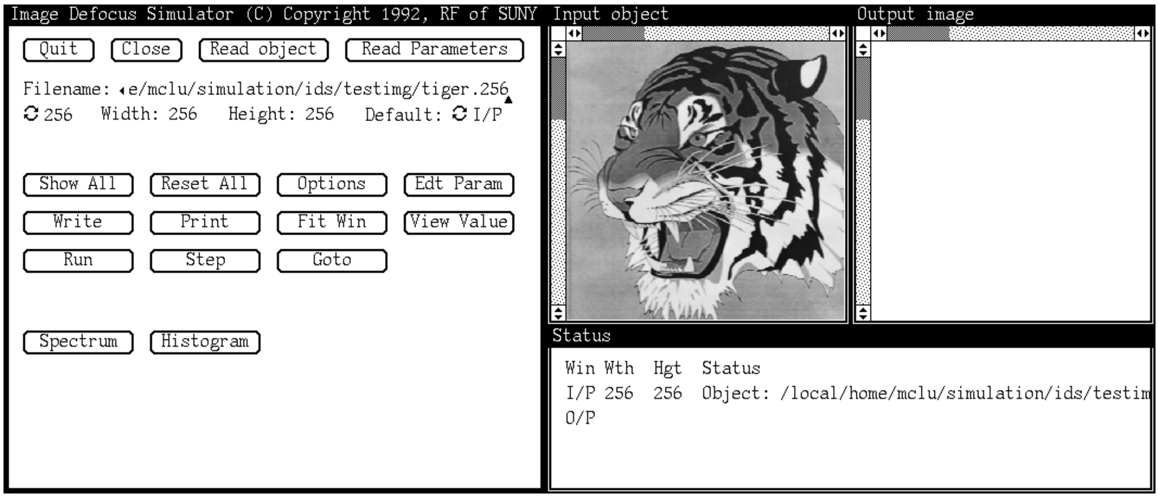



Figure 4-6: Full operation menu.

border can be treated as a zero-padded, mirrored, or periodic image during convolution. For "zero-padded" option, the image outside the field of view is simply treated as a dark area, *i.e.*, an area with zero values. For "periodic" option, the image is considered as $f(aM+m, bN+n) = f(m, n)$ for an $M \times N$ image where $a, b \in \{0, \pm 1, \pm 2, \cdots\}$. And for "mirrored" option, the image is first reflected along its right side border, then the resulting image is reflected along its top border; this gives an image which is four times larger than the original image. This four times larger image is then taken to be wrapped around at its borders. Note that, the "mirrored" option gives a periodic image whose period is twice that of the "periodic" option. After the input image and the parameters are loaded, additional command buttons are available as shown in Figure 4-6.

The "Run", "Step", and "Goto" commands control the execution of the simulation. Users can step or go to any particular stage in Figure 3-1 or any transformation step to examine its output. The value of the pixels of the image can be viewed via "View Value" command. We also provide Fourier spectrum and histogram analysis of a given image by using the "Spectrum" and the "Histogram" commands. The "Default" choice specifies the window (object or image) to which the command should be applied.

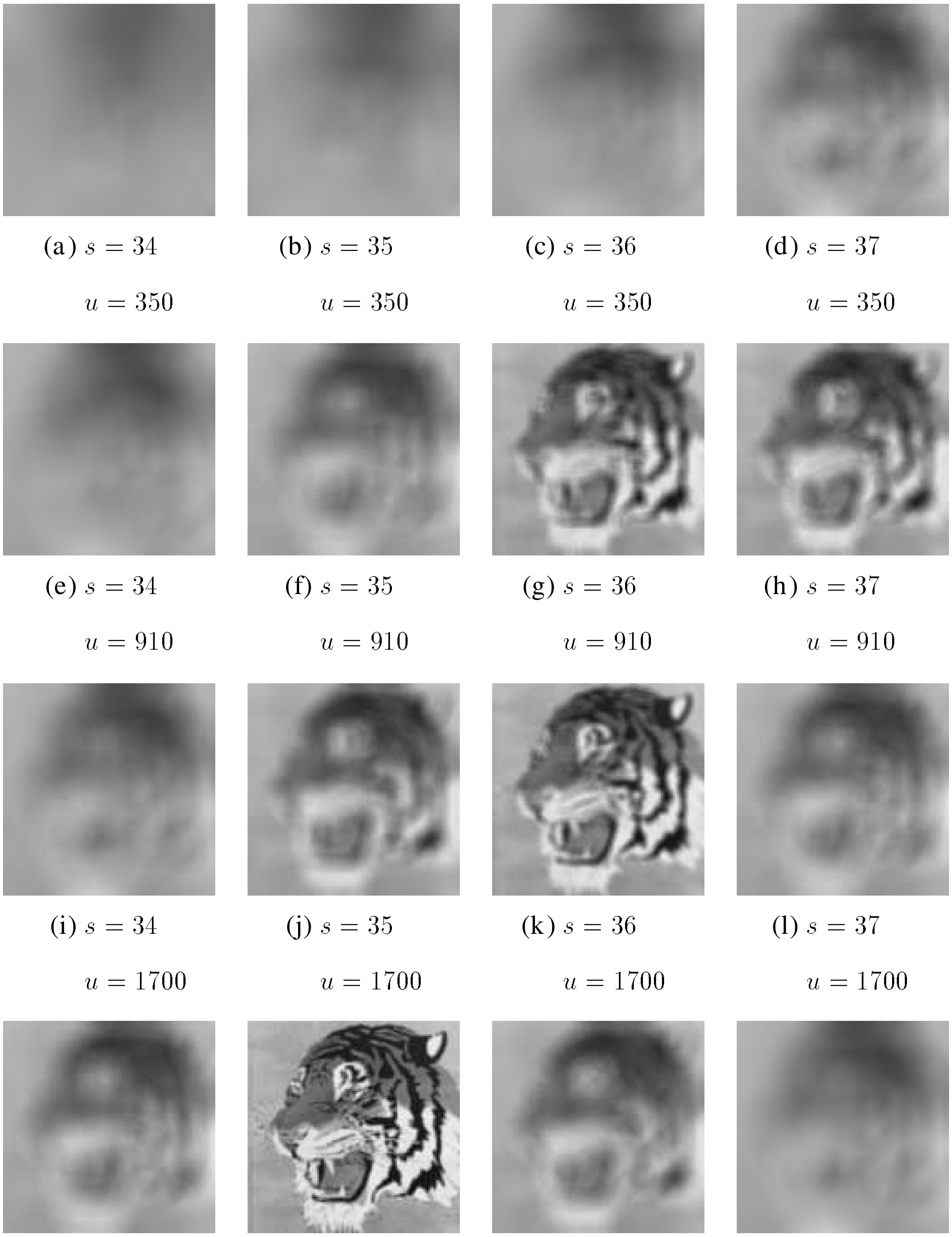
Finally, the synthesized image can be saved to a file and/or sent to printer for printing using a halftone algorithm. All the images shown in Section 4.4 are processed this way.

## 4.4 Simulation Results

### 4.4.1 Geometric Optics

Now we present the simulation results of IDS under the paraxial geometric optics model. The original image is shown in Figure 4-6 and the simulation results are shown in Figure 4-7.

Figure 4-7 shows 16 pictures of *Tiger* arranged in 4 rows and 4 columns. The distance of the object increases (350mm-8943mm) row-wise from top to bottom whereas the distance $s$ between the lens and the image detector increases (34mm-37mm) column-wise from left to right.



(a) $s = 34$ $u = 350$
(b) $s = 35$ $u = 350$
(c) $s = 36$ $u = 350$
(d) $s = 37$ $u = 350$

(e) $s = 34$ $u = 910$
(f) $s = 35$ $u = 910$
(g) $s = 36$ $u = 910$
(h) $s = 37$ $u = 910$

(i) $s = 34$ $u = 1700$
(j) $s = 35$ $u = 1700$
(k) $s = 36$ $u = 1700$
(l) $s = 37$ $u = 1700$

(m) $s = 34$ $u = 8943$
(n) $s = 35$ $u = 8943$
(o) $s = 36$ $u = 8943$
(p) $s = 37$ $u = 8943$

Figure 4-7: Simulated images for *Tiger*.

The focal length and the F-number are fixed at 35mm and 4, respectively. In Figure 4-7, for convolution operation, the "mirror" option mentioned earlier was used.

In Figure 4-7, somewhere along a direction parallel to the left-bottom to top-right diagonal, the pictures are focused, whereas, on either side of the this, the image defocus increases. This is consistent with the fact that image defocus should increase when either the object is moved farther or closer from its focused position, or when the image detector is moved farther or closer from its focused position.

### 4.4.2 PSF From Diffraction Model

The model of point spread function derived from Fraunhofer Diffraction principle is more accurate than the one based on geometric optics. In this subsection, we present the simulation results for a point spread function derived from Fraunhofer diffraction principle. The derivation for the point spread function in this case can be found in Appendix B where Equation (B.21) is the point spread function, and Equations (B.27)-(B.30) are the relationships between the PSF and the camera parameters.

The point spread function in this case is the well-known *Airy pattern*. The result of the point spread function generated by the simulation is shown in Figure 4-8 where the parameters used are $\lambda = 5790\ \mathring{A}$, $f = 35\,mm$, $F = 4$, $u = 1500\,mm$, and $s = 35.245\,mm$. Figure 4-8 is magnified

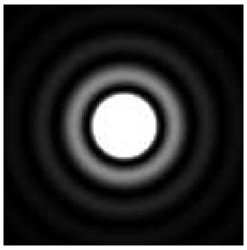


Figure 4-8: Airy pattern.

by about 2.6 times and over-exposed in the center by 10,000 so that the outer rings become visible.

One example of simulation for this case is shown in Figure 4-9 (compare this with Figure 4-3 which is based on geometric optics).

## 4.5 Applications

IDS has been used in our laboratory to test and debug the implementations of two new methods for finding distance of objects [48, 49] from their defocused images. IDS was used to generate a large number of test images in the range of interest for different camera parameter settings. Then, the programs implementing the methods of finding distance were tested using the generated images as input.

Figure 4-10 and Figure 4-11 show the results of testing one of the methods described in Subbarao and Wei [49]. Synthetic images were generated for 94 distinct distances of an object for three different lens positions. Other camera parameters were fixed. For each object distance,

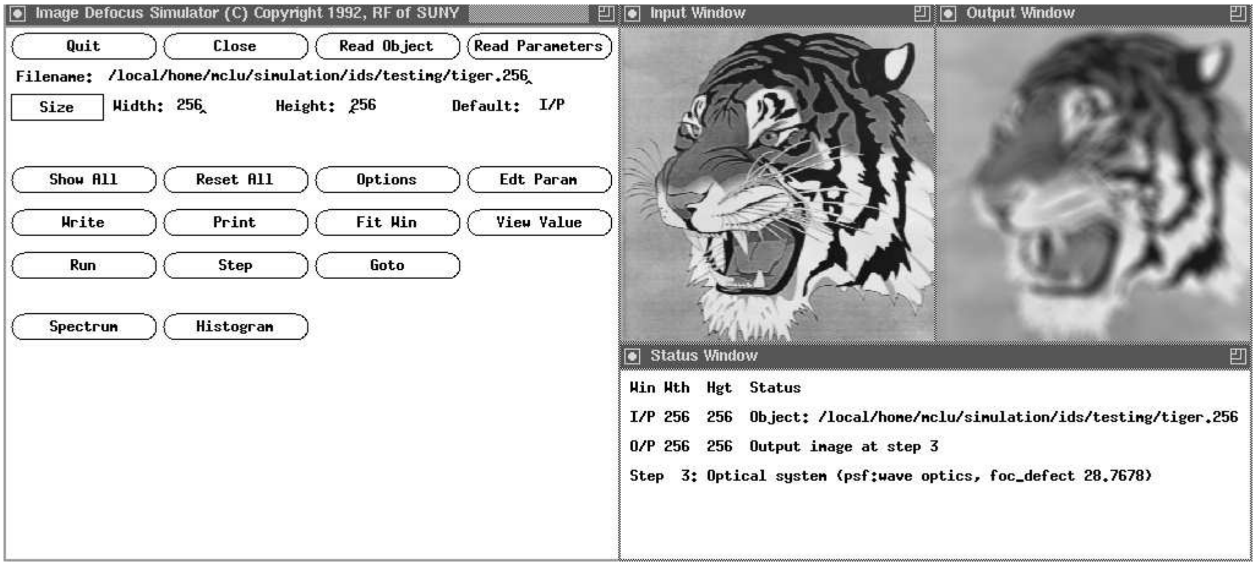
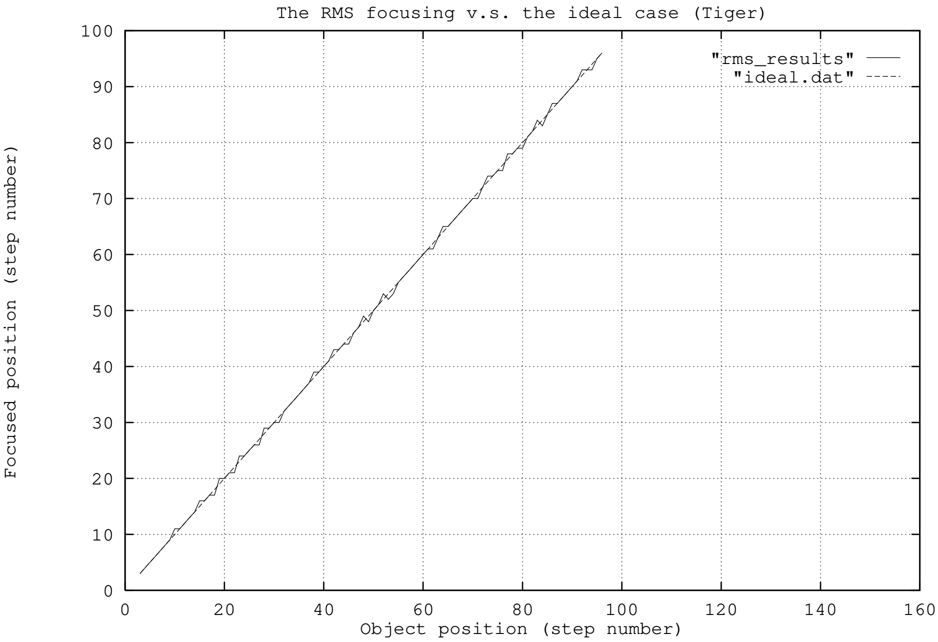



Figure 4-9: Simulation result using diffraction principle.

the corresponding three images were given as input to the program for finding distance. The distance determined by the program was used to calculate a lens position which would bring the object into sharp focus. The lens position determined by the program is plotted against the correct lens position in Figure 4-10(b) and Figure 4-11(b). The results of the program are in solid line and the correct results are in dashed line. The rms deviation of the solid line from the dashed line in both cases is within 1% of the maximum range. This validates the implementation of the program and the theory behind it. The program was also run on an actual camera system named Stonybrook Passive Autofocusing and Ranging Camera System (SPARCS) built in our laboratory. In this case also, the program performed well.

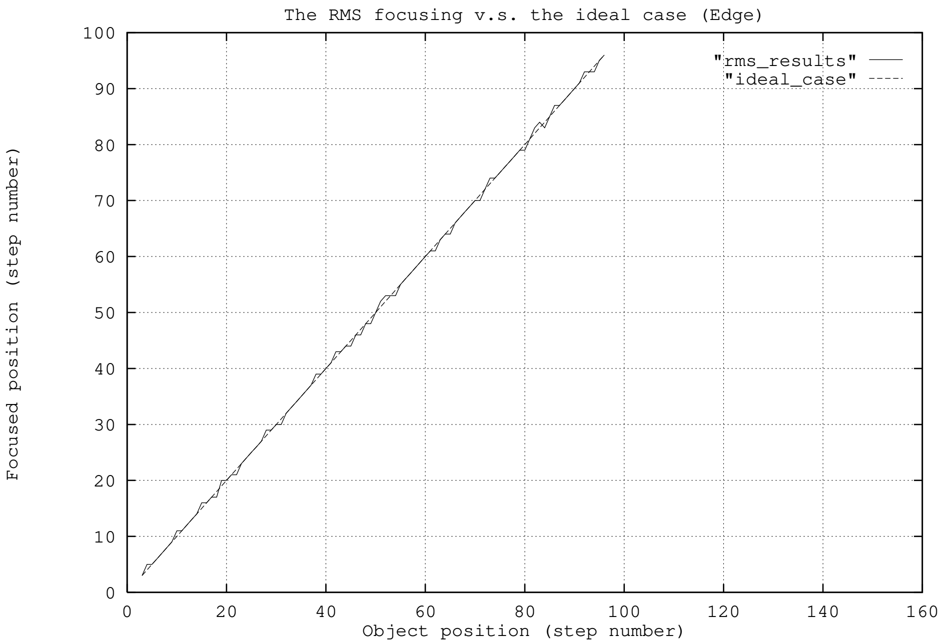
(a)




(b)

Figure 4-10: (a) Test object *Tiger*, (b) Simulation results.

(a)




(b)

Figure 4-11: (a) Test object *Edge*, (b) Simulation results.

## 4.6 Conclusion

Based on the proposed computational model presented in the previous chapter, we have developed a computer simulation package called IDS. IDS is user-friendly, modular, and extensible. It has been used to test and debug the implementations of two new methods on depth from defocus. It can also be used to generate synthesized images for research on restoration of defocused images. IDS can be easily extended for research in other areas such as simulation of image formation in a stereo CCD camera, and simulation of the image of a moving object in a CCD camera. At the expense of increased computation, IDS can also be extended for spatially-variant point spread functions. The computational time in this case can be reduced by adapting IDS to parallel computing.

The IDS simulation engine is written in ANSI C which is portable to almost any machine with a C compiler. The DTI user interface is also machine-independent and hence portable. The SGI user interface can be directly moved to SUN workstations; while the XGI graphical user interface can be easily ported to virtually any machine with standard X11/R4 distributions. Therefore, IDS can be easily installed and used by other researchers.

# Chapter 5

# The AVS Active Vision Camera System

## 5.1 Introduction

In active vision,changing the direction of view and the visual parameters facilitates and makes efficient the computational stage of machine vision. An active vision system can be considered as a system that integrates visual sensing and action. There are two common tasks to be solved in active vision systems: one is the correspondence problem in stereo imaging, the other is motion estimation to dynamically track the objects in the scene. Many researchers have proposed algorithms [2, 5, 19, 22, 23, 26, 32, 51] for these tasks. Our objective here is to provide researchers a simulation environment to simulate image sensing process in motion and stereo systems.

In this Chapter, a computer simulation system called Active Vision Simulator (AVS) is presented. AVS is an extension of the Image Defocus

Simulator (IDS) presented in the previous Chapter. It can be used to simulate image formation process in a monocular (MONO mode) or a binocular (STEREO mode) camera system. The simulation of curved objects is also included in AVS. The user interfaces for AVS are similar to those in IDS, *i.e.,* two graphical user interfaces – Sunview Graphical Interface (SGI) and X-window Graphical Interface (XGI), and a dummy terminal user interface – Dummy TTY Interface (DTI).

This Chapter is organized as follows: Section 2 describes the simulation algorithms used for curved objects, motion simulation, and stereo imaging; Section 3 describes the user interfaces of AVS; Section 4 presents the simulation results; and finally, Section 5 concludes this Chapter.

## 5.2 Simulation Algorithms

### 5.2.1 Curved Objects

Consider the photometric information $f(\theta, \phi, \lambda, t)$ and the geometric information $r(\theta, \phi)$. $r(\theta, \phi)$ contains the depth information of objects in the scene. For curved objects, $r(\theta, \phi)$ can not be approximated by a constant $u$. Under this situation, the point spread function is space-variant and is specified by $h(\theta, \phi, \theta', \phi', r(\theta, \phi), \vec{e})$ as discussed in Chapter 3. In a Cartesian coordinate system, the geometric information and the point spread function can be represented as $r(x, y)$ and $h'(x, y, r(x, y), \vec{e})$, respectively, under the assumption that all CCD elements have the same characteristics. In this case, the output of the optical system will be:

$$f_3(x, y, \lambda, t) = h'(x, y, r(x, y), \vec{e}) \star f_2'(x, y, \lambda, t) \tag{5.1}$$

where $\star$ is the convolution operator.

Assume there are $N$ different distances ($r_i, i = 1, \cdots, N$) in the scene. Using superposition, $f_2'(x, y, \lambda, t)$ can be decomposed into $N$ components as:

$$f_2'(x, y, \lambda, t) = \sum_{i=1}^{N} f_{2i}(x, y, \lambda, t) \tag{5.2}$$

where

$$f_{2i}(x, y, \lambda, t) = \begin{cases} f_2'(x, y, \lambda, t), & \text{if } r(x, y) = r_i \\ 0, & \text{elsewhere} \end{cases}$$

Thus, Equation (5.1) becomes:

$$\begin{aligned} f_3(x, y, \lambda, t) &= \sum_{i=1}^{N} h'(x, y, r(x, y), \vec{e}) \star f_{2i}(x, y, \lambda, t) \\ &= \sum_{i=1}^{N} h_i(x, y, \vec{e}) \star f_{2i}(x, y, \lambda, t) \end{aligned} \tag{5.3}$$

where $h_i(\cdot)$ is the point spread function for the planar object at distance $r_i$. Note that, if the profile of the scene in a small field-of-view is smooth, we have $N = 1$ and

$$f_3(x, y, \lambda, t) = h'(x, y, \vec{e}) \star f_2'(x, y, \lambda, t)$$

as derived in Equation (3.4). Therefore, the algorithm for the simulation of curved objects can be summarized as in Figure 5-1 where FFT

Step 1: Decompose the object into $N$ planes, $f_{2i}(x, y, \lambda, t)$, of distance
$r_i, i = 1, \cdots, N$, according to the depth map information;
Step 2: **for** $i = 1$ **to** $N$ **do**
**begin**
Compute and store the point spread function $h_i$
**end**;
Step 3: $f_3 \leftarrow 0$;
**for** $i = 1$ **to** $N$ **do**
**begin**
$f_3 \leftarrow f_3 + f_{2i} \star h_i$
**end**;

Figure 5-1: Simulation algorithm for curved objects.

algorithm can be applied in Step 3 to reduce the large amount of computations needed.

### 5.2.2 Motion Simulation

The motion parameters used in the simulation are specified by the vector $\vec{m} = [V_x \quad V_y \quad V_z \quad \Delta t]$, where $V_x$, $V_y$, and $V_z$ are the velocity components of the motion; $\Delta t$ is the duration of the motion. Here, we assume that the scene contains only rigid objects so that the object will not change its shape while it is moving.

For objects moving perpendicular to the optical axis, *i.e.* $V_z = 0$, the size of the objects in the scene will remain unchanged. However, part of the original image will move out of the field-of-view and will not appear in the image plane. This will also introduce other objects into the scene which are not in the original image. Therefore, the original input image must include the objects that may come into the camera's field of view due to motion. This problem can be avoided by assuming a dark background. When parts of the objects move out of the camera's field of view, the dark background appears in the field of view. Here, we use this approach for its simplicity and efficiency in memory management.

When an object moves toward or away from the camera, the objects in the scene will be enlarged or shrunk. Therefore, resampling must be done to compensate for this effect. In AVS, we use bi-linear interpolation to compute the value $g(m,n)$ from its four neighbors $f(i,j)$, $f(i+1,j)$, $f(i,j+1)$, and $f(i+1,j+1)$. The result is:

$$g(m,n) = a \cdot (m-i) + b \cdot (n-j) + c \cdot (m-i)(n-j) + d \tag{5.4}$$

where $i \leq m \leq i+1$, $j \leq n \leq j+1$, and

$$\begin{aligned} a &= f(i+1,j) - f(i,j) \\ b &= f(i,j+1) - f(i,j) \\ c &= f(i+1,j+1) + f(i,j) - f(i,j+1) - f(i+1,j) \\ d &= f(i,j) \end{aligned}$$

The simulation of an object moving with an arbitrary motion vector $\vec{m}$ is done by a shift operation if $V_x \neq 0$ or $V_y \neq 0$, and then a resampling operation if $V_z \neq 0$ to get the synthesized image. The algorithm is shown in Figure 5-2. Note that, during up-sampling process, the image might be smoothed, while in the down-sampling process, some image details might be lost.

### 5.2.3 Stereo System

For a binocular camera system, the two camera positions are specified by the vectors $\vec{O}_l = [x_l \quad y_l \quad z_l \quad \theta_{xl} \quad \theta_{yl} \quad \theta_{zl}]$ and $\vec{O}_r = [x_r \quad y_r \quad z_r \quad \theta_{xr} \quad \theta_{yr} \quad \theta_{zr}]$ with respect to the global origin $O$ in Figure 3-5. The components of these vectors specify the positions and the orientations of the two cameras.

Stereo image pairs can be generated using the motion algorithm presented in Figure 5-2 where the motion displacement corresponds to $(-x_l, -y_l, -z_l)$ for the left camera and $(-x_r, -y_r, -z_r)$ for the right camera. Note that, the orientation parameters are fixed to be $\theta_x = 90°$, $\theta_y = 90°$, and $\theta_z = 0°$.

```
Step 1: if V_x ≠ 0 or V_y ≠ 0 then
        begin
            Shift the object horizontally by the amount V_x Δt;
            Shift the object vertically by the amount V_y Δt
            Append dark background if part of the object is
                moved out of the view;
        end;
Step 2: if V_z ≠ 0 then
        begin
            if V_z > 0 then
                move the object toward the camera and resample;
            else /* V_z < 0 */
                move the object away from the camera and down-sample;
            Append dark background if the neighbor of the object
                appears in the view;
        end;
```

Figure 5-2: Simulation algorithm for moving objects.

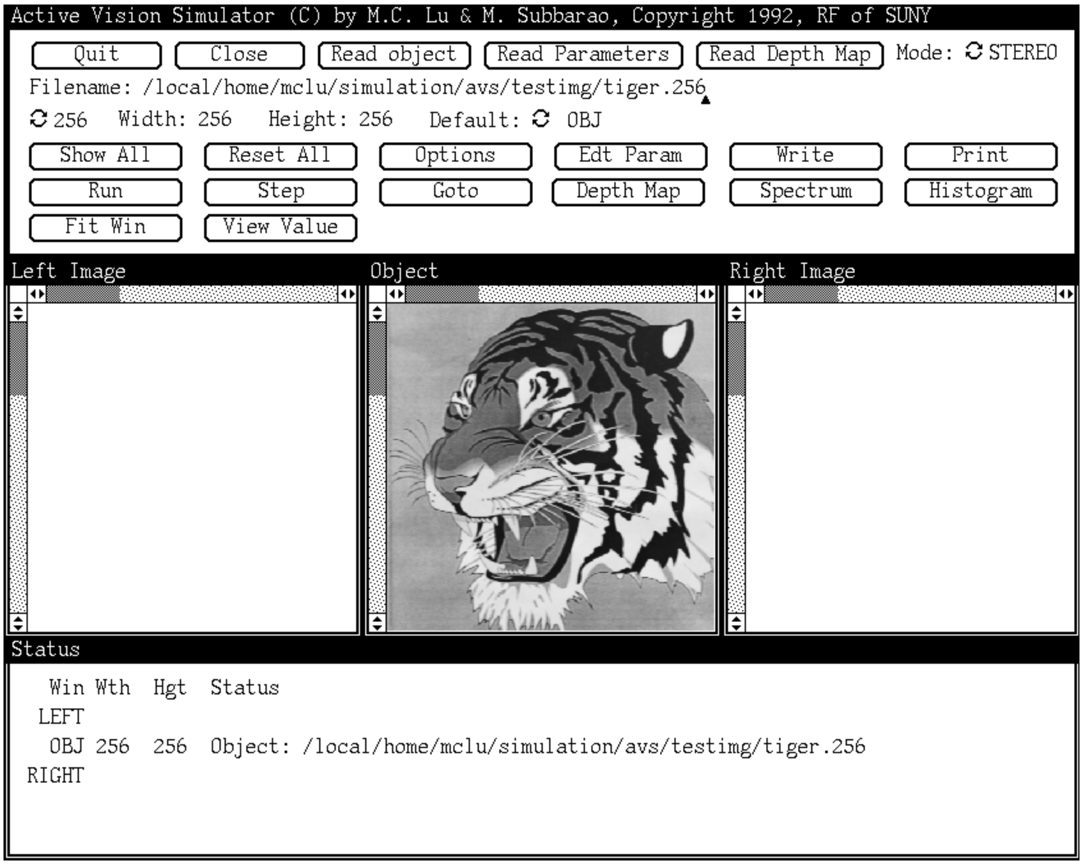



Figure 5-3: AVS graphical user interface.

## 5.3 The User Interfaces

Three user interfaces are provided in AVS – SGI, XGI, and DTI. The appearance and the basic functions of these user interfaces are similar to those in IDS. AVS has all the functions of IDS plus one more window and some additional features as shown in Figure 5-3. Besides, the single parameter window in IDS is now three parameter windows in AVS – one

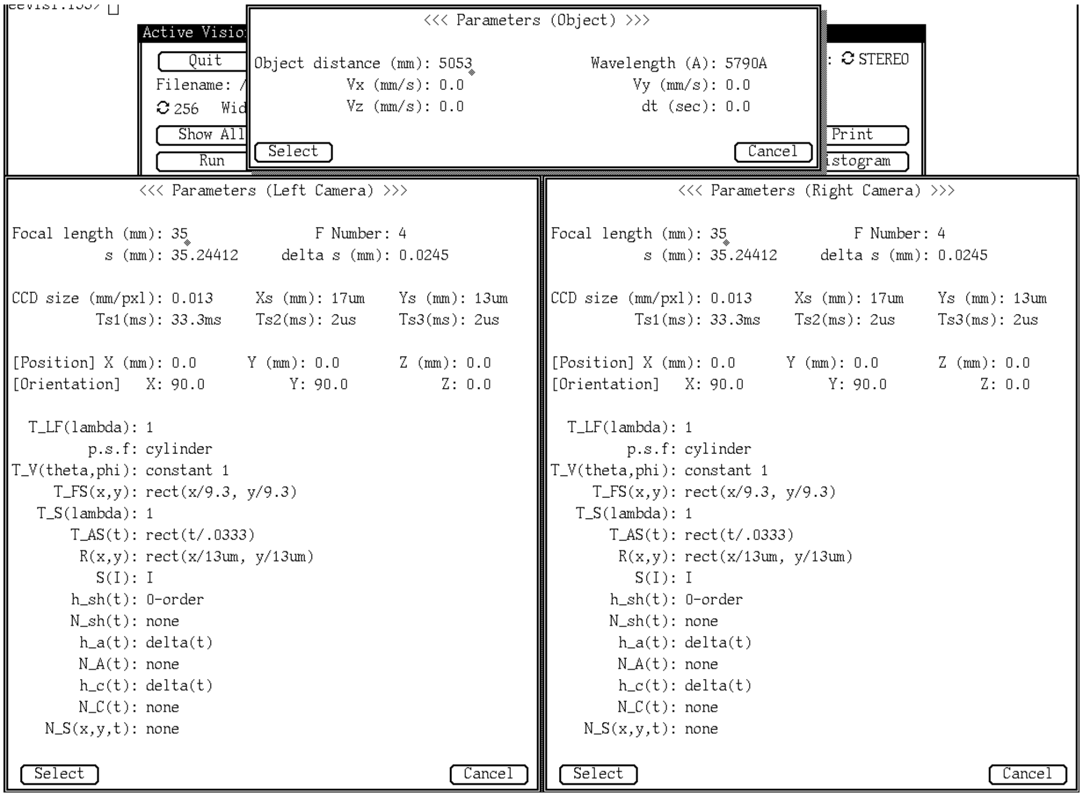



Figure 5-4: Categorized parameters in AVS.

each for the left and the right camera (camera parameters), the other for object-specific parameters as shown in Figure 5-4

For curved object simulation, the depth map is read from a file by using the "Read DepthMap" command. The depth information stored in the file is a relative value, $\Delta r(x, y)$, with respect to the object distance $u$ which is the shortest distance between the global origin $O$ and the scene (*i.e.*, $\min\{d_i, i = 1, \cdots, N\}$). The object distance $r(x, y)$ is computed as

$$d(x, y) = \Delta r(x, y) \cdot k + u$$

where $k$ is the scaling factor option in the option menu popped up by the

"Option" command. The format of the depth map file is also specified in this menu.

When the depth map is loaded, depth information $r(x, y)$ can be viewed by using the "Depth Map" command which will pop up a window with depth profile. The value of the depth at each point can then be viewed on the screen by moving the mouse pointer to the desired location. In DTI, the value is displayed according to the command line arguments used.

The parameters can be edited/viewed by the "Edt Param" command which searches the "Default" field for target window. The target can be object parameters, left camera parameters, or the right camera parameters as shown in Figure 5-4. The object parameters contain object distance and wave length information for general object information; and $V_x, V_y, V_z, dt$ for motion information. The camera parameters are basically the same as those in IDS except that (i) the object distance and wave length information are moved to the object parameters window; and (ii) the camera position and orientation information ($\vec{O}_r$, $\vec{O}_l$) are added.

Another added feature is the "Mode" choice in Figure 5-3 which can be toggled between MONO and STEREO mode to simulate monocular and binocular image formation process. In "MONO" mode, "Left Camera" window will disappear. Therefore, the image will be synthesized in the "right camera" window by default. All the other commands are borrowed from IDS and carry the same functions. Besides, the *3-D* object generation program described in Chapter 2 is also integrated into this system as a tool to generate the depth map information $r(x, y)$.

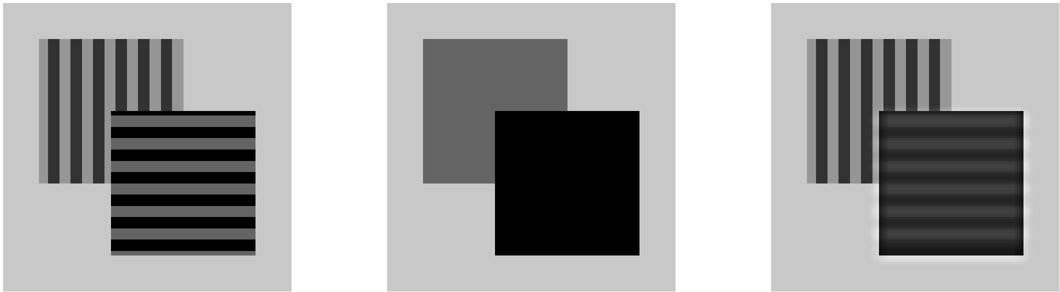
## 5.4 Simulation Results

In this section, some simulation results are presented to illustrate the capability of AVS simulator.

### 5.4.1 Curved Objects

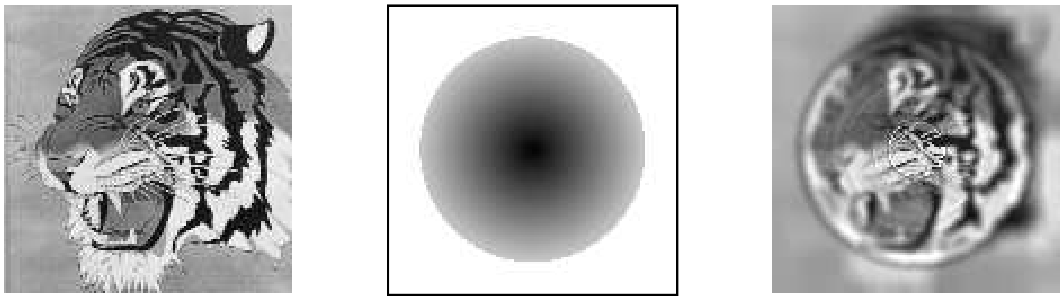
Figure 5-5 gives a simulated image of two striped boxes placed at two distances. The scene and the depth map are shown in Figure 5-5(a) and Figure 5-5(b), respectively. Note that, the darker the value the depth map is coded, the closer the object is to the camera. The horizontal-striped box is located near the camera, while the vertical-striped box is located away from the camera. The camera parameters are adjusted to focus at the vertical-striped box. The resulting image is shown in Figure 5-5(C).

Another example is the tiger face placed on a cone-shape depth map as shown in Figure 5-6(a) and Figure 5-6(b), respectively. The depth range is from 2000mm to 3600mm (inside the cone) and the camera parameters are adjusted to focus at an object distance of 2000mm. The resulting image is shown in Figure 5-6(c). Note that, the depth outside



(a) (b) (c)

Figure 5-5: Simulated images for two planar boxes placed at different distances.



(a) (b) (c)

Figure 5-6: Simulated images for object placed on a cone-shaped depth map.

(a) $\vec{m}_1$ (b) (c) $\vec{m}_2$

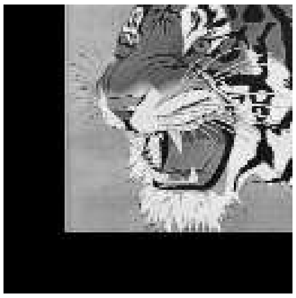
Figure 5-7: Resampled images in motion simulation.

the cone (white area) is assumed to be infinity. Therefore, a circle is visible in Figure 5-6(c).

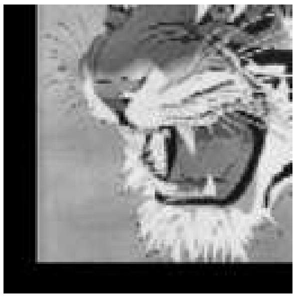
### 5.4.2 Motion

Figure 5-7 shows the simulated images of moving objects. The center image is the original one. The left and the right images are generated with motion vector $\vec{m}_1 = [0 \quad 0 \quad -2.5m/s \quad 1sec]$ and $\vec{m}_2 = [0 \quad 0 \quad 2.5m/s \quad 1sec]$, respectively. All other parameters are the default ones in Figure 5-4.

The simulation of the shift operation (motion with $V_z = 0$) and the combined operation are shown in Figure 5-8 with $\vec{m}_3 = [100 \quad 100 \quad 0 \quad 1sec]$



(a) $\vec{m}_3$



(b) $\vec{m}_4$

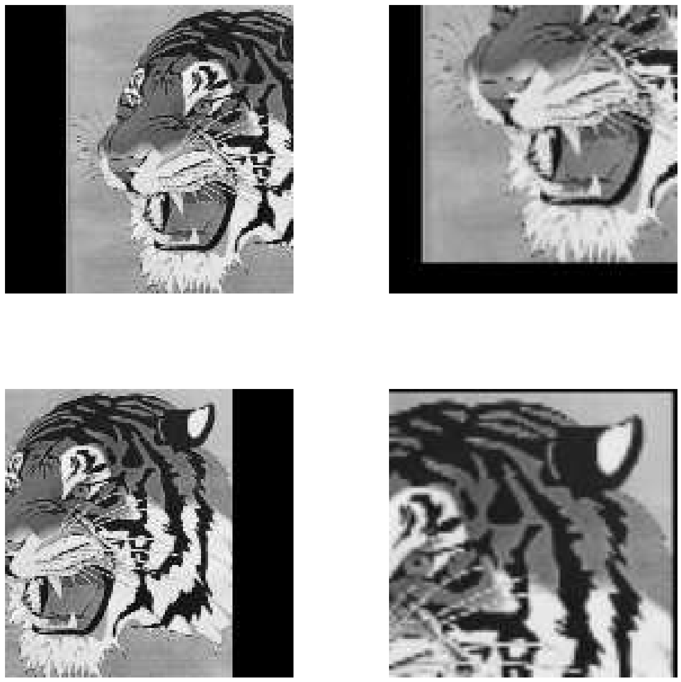
Figure 5-8: Simulated images under shift operation and the general motion vector.

and $\vec{m}_4 = [100 \quad 100 \quad 2.5m/s \quad 1sec]$. All other parameters remain the default ones.

Note that in Figure 5-7(c), Figure 5-8(a) and (b), dark background is introduced because the object is moved away from camera or part of the object move out of the field of view as mentioned in Section 5.2.

### 5.4.3 Stereo

The simulation of the stereo image pairs for the left camera position $\vec{O}_l = [-100mm \quad y_l \quad z_l \quad 90° \quad 90° \quad 0]$ and the right camera position $\vec{O}_r = [100mm \quad y_r \quad z_r \quad 90° \quad 90° \quad 0]$ is shown in Figure 5-9 where the first row is the image on the left camera, the second row is the



(a) (b)

Figure 5-9: Simulation of stereo image pairs.

image on the right camera. In Figure 5-9(a), $y_l = z_l = y_r = z_r = 0$ which corresponds to the shift operation; in Figure 5-9(b), the left lens is moved toward the object with $y_l = -100mm, z_l = 500mm$ while the right camera is moved toward the camera with $y_l = 100mm, z_l = -500mm$. The image resampling and the dark background effect are visible in these simulations.

## 5.5 Conclusion

In this Chapter, we have implemented the curved object, motion, and stereo image sensing simulation in the computer simulation package called Active Vision Simulator. AVS is a natural extension of IDS presented in the previous Chapter. It can be used to synthesize the images for research on image restoration, motion analysis, depth from defocus, and algorithms for solving the correspondence problem in stereo vision area.

The efforts spent on extending the IDS to AVS is limited – two added modules on motion and stereo, and some changes in the user-interfaces – because of the module design and embedded extensibility of our original design of IDS. Again, AVS can also be easily extended if needed and can be used by other researchers on the verification of various vision theories.

# Chapter 6

# Conclusion

## 6.1 Summary

In this dissertation, we presented the computer modeling and simulation techniques for two computer vision problems – object recognition and image sensing process. We first focused on the *3-D* object identification and pose estimation problem in object recognition process. The goal was to identify an unknown object from the given image and to estimate the orientation of the object in the scene.

We have successfully modeled and implemented a *3-D* object identification and pose estimation algorithm using a neural network architecture. Feature vectors of the image were extracted using a fixed weight neural network. These vectors were then fed to a variable weight neural network for training and identification purpose. The back-propagation algorithm was used to train the network. A simulation system called

Object identification and Pose Estimation Network simulator (OPEN) was developed which can be used to obtain a configuration for better identification rate. Once the network is trained, the identification and pose estimation of the unknown object can be obtained concurrently. The proposed model is highly parallel and can be realized using hardware.

We also presented a computational model for image sensing and formation process. This model decouples the photometric and the geometric properties of the object in the scene. It consists of seven functional blocks, sixteen computational steps, and twenty-seven user controllable parameters. Based on this model, we have developed a computer simulation system called Image Defocus Simulator (IDS). IDS provides three user-friendly interfaces. It is efficient, modular, and extensible. IDS is currently being used by our research group and other research groups in academic and industrial laboratories.

The computational model for monocular camera system was further extended to motion simulation and stereo vision system. A computer simulation system based on the extended model was also developed. This simulator, called Active Vision Simulator (AVS), inherits all the functions of the IDS plus the ability to simulate curved objects, moving objects, and stereo image pairs. AVS is also efficient, modular, and extensible.

To sum up, the theories on the object recognition, image sensing and formation, motion simulation, and stereo vision have been studied in this dissertation. We presented a neural network computational model to solve the problem of *3-D* object identification and pose estimation;

we also presented a computational model to synthesize images in the monocular CCD camera system, motion environment, and stereo vision systems. All the models presented in this dissertation are implemented and can be easily used/modified by other researchers for their computer vision applications.

## 6.2 Future Research

The research described here can be extended in several ways. One natural extension to AVS is the modeling and simulation of a stereo camera system where the relative position and orientation of the two cameras is arbitrary. At present, the AVS is limited to a stereo camera where the optical axes of the left and the right cameras are parallel. When the optical axes are not parallel, the scene geometry specified as input with respect to one camera will have to be transformed using a translation vector and a rotation matrix to obtain the geometry with respect to the other camera. Further, ray tracing will be needed to compute the sensed image near the occlusion boundaries. This extension is computationally very intensive.

A second extension is a more detailed modeling and simulation of the various components of a camera system such as the optical system, CCD sensor, sources of noise, etc. This also increases the computation.

Further extensions include modeling and simulation of color camera systems, parallel implementation on parallel/super computers, and efficient modeling and simulation of a large sequence of images to create a movie of a scene which has multiple objects each moving with different translational and rotational parameters.

# Bibliography


[1] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools,* Addison-Wesley, Massachusetts, 1986.

[2] S. T. Barnard and M. A. Fischler, "Computational Stereo," *Computing Surveys*, Vol. 14, No. 4, Dec. 1982.

[3] R. M. Bolle and D. B. Cooper, "Bayesian Recognition of Local 3-D Shape by Approximating Image Intensity Functions with Quadric Polynomials," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No. 4, pp. 418-429, July 1984.

[4] M. Born and E. Wolf, *Principles of Optics,* Sixth edition, Pergamon Press, Oxford, 1980.

[5] P. Bouthemy and P. Lalande, "Motion Detection in an Image Sequence Using Gibbs Distributions," *Proceeding of IEEE International Conference on Acoustic, Speech, and Signal Processing,* pp. 1651-1654, May 1989.

[6] Y. C. Chen, "Lens Effect on Synthetic Image Generation Based on Light Particle Theory," *Computer Graphics,* pp. 347-366, 1987.

[7] Y. C. Chen, "Synthetic Image Generation for Highly Defocused Scenes," in N. Magnenat-Thalmann and D. Thalmann (Eds.), *New Trends in Computer Graphics,* Proceeding of CG International'88, Springer-Verlag, Berlin Heidelberg, pp. 117-125, 1988.

[8] D. Cyganski and J. A. Orr, "Applications of tensor theory to object recognition and orientation determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-7, pp. 662-673, Nov. 1985.

[9] A. M. Darwish and A. K. Jain, "A Rule Based Approach for Visual Pattern Inspection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 10, No. 1, pp. 56-68, Jan. 1988.

[10] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* John Wiley & Sons, New York, 1973.

[11] A. R. Edmonds, *Angular Momentum in Quantum Mechanics,* Princeton Univ. press. 1974.

[12] E. A. Feibush, M. Levoy, and R. L. Cook, "Synthetic Texturing Using Digital Filters," *SIGGRAPH* 80, pp. 294-301, 1980.

[13] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principle and Practice,* Second edition, Addison-Wesley, Massachusetts, 1990.

[14] J. W. Goodman, *Introduction to Fourier Optics,* McGraw-Hill Book Co., San Francisco, 1968.

[15] G. Healey and R. Kondepudy, "Modeling and Calibrating CCD Cameras for Illumination Insensitive Machine Vision," *Proceedings of Optics, Illumination, and Image Sensing for Machine Vision VI,* SPIE Proceedings Conf. 1614, Nov. 1991.

[16] E. Hecht, *Optics,* Second edition, Addison-Wesley, Massachusetts, 1987.

[17] L. Hodges and D. McAllister, "Stereo and Alternating-Pair Techniques for Display of Computer-Generated Images," CG&A, **5**(9), pp. 38-45, Sep. 1985.

[18] H. H. Hopkins, "The Frequency Response of a Defocused Optical System," *Proceeding of Royal Society of London,* A 231, pp. 91-103, 1955.

[19] B. K. P. Horn, *Robot Vision,* McGraw-Hill, New York, 1986.

[20] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, pp. 141-152, 1985.

[21] A. K. Jain and R. Hoffman, "Evidence-Based Recognition of 3-D Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 10, No. 6, pp. 783-802, Nov. 1988.

[22] J. Konrad and E. Dubois, "Bayesian Estimation of Motion Vector Fields," *IEEE Transaction on Pattern Analysis and Machine Intelligence,* Vol. 14, No. 9, pp. 910-927, Sep. 1992.

[23] E. P. Krotkov, *Exploratory Visual Sensing for Determing Spatial Layout with an Agile Stereo Camera System,* Ph.D. dissertation, Department of Computer and Information Science, University of Pennsylvania, PA, April 1987.

[24] M. D. Levine, *Vision in Man and Machine,* McGraw-Hill, New York, 1985.

[25] R. P. Lippmann, "An introduction to computing with Neural nets," *IEEE ASSP Magazine,* April 1987.

[26] Y. Liu and T. S. Huang, "A Linear Algorithm for Determining Motion and Structure From Line Correspondences," *Computer Vision, Graphics, and Image Processing,* Vol. 44, No. 1, pp. 35-57, 1988.

[27] C.-H. Lo and H.-S. Don, "3-D moment forms : their construction and application to object identification and positioning," *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol. PAMI-11, pp. 1053-1064, Oct. 1989.

[28] C.-H. Lo and H.-S. Don, "Pattern Recognition Using 3-D Moments," *The Tenth International Conference on Pattern Recognition,* Atlantic City, June 16-21, 1990.

[29] M.-C. Lu, C.-H. Lo, and H.-S. Don, "3-D Object Identification and Pose Estimation," in C.H. Dagli, S. R. T. Kumara, and Y. C. Shin (Eds.), *Intelligent Engineering Systems Through Artificial Neural Networks,* ASME Press, New York, pp.473-478, 1991.

[30] M.-C. Lu, C.-H. Lo, and H.-S. Don, "A Neural Network Approach to 3-D Object Identification and Pose Estimation," *Proceeding of the IEEE International Joint Conference on Neural Networks,* Vol. 3, pp.2600-2605, Singapore, Nov. 1991.

[31] M.-C. Lu and M. Subbarao, "Image Defocus Simulator: A Software Tool," Technical Report No. 92.05.27, Computer Vision Laboratory, Department of Electrical Engineering, State University of New York, Stony Brook, May 1992.

[32] D. Murray and B. Buxton, "Scene Segmentation From Visual Motion Using Global Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, pp. 220-228, March 1987.

[33] G. Neilson and D. Olsen, Jr., "Direct Manipulation Techniques for 3D Objects Using 2-D Locator Devices," in *Proceedings of the 1986 Workshop on Interactive 3D Graphics,* ACM, New York, pp. 175-182, 1987.

[34] A. V. Oppenheim and R. W. Schafer, *Discrete-time Signal Processing,* Prentice-Hall, Englewood Cliffs, 1989.

[35] M. Potmesil and I. Chakravarty, "Synthetic Image Generation with a Lens and Aperture Camera Model," *ACM Transactions on Graphics,* Vol. 1, No. 2, pp. 85-108, April 1982.

[36] A. Reeves and R. Taylor, "Identification of three-dimensional objects using range information", *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol. 11, No. 4, pp. 403-410, Apr. 1989.

[37] A. Rosenfeld and A. C. Kak, *Digital Picture Processing,* Second edition, Academic Press, New York, 1982.

[38] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in D. Rumelhart and J. McClelland, *Parallel Distributed Processing,* Vol. 1 M.I.T Press, Cambridge, MA. 1986.

[39] D. Rumelhart and J. McClelland, *Parallel Distributed Processing,* Vol. 1 M.I.T Press, Cambridge, MA. 1986.

[40] S. D. Roth, "Ray Casting for Modeling Solids," *Computer Graphics and Image Processing,* **18**, pp. 109-144, 1982.

[41] F. A. Sadjadi and E. L. Hall, "Three dimensional moment invariants," *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol. PAMI-2, pp. 127-136, Mar. 1980.

[42] S. A. Shafer, "Automation and Calibration for Robot Vision Systems," Technical Report No. CMU-CS-88-147, Computer Science Department, Carnegie Mellon University, 1988.

[43] W. N. Sproson, *Colour Science in Television and Display Systems,* Adam Hilger Ltd., Bristol, 1983.

[44] M. Subbarao, "Efficient Depth Recovery Through Inverse Optics," In: Freeman H (Ed.) *Machine Vision for Inspection and Measurement,* Academic Press, Boston, pp 101-126, 1988.

[45] M. Subbarao, "On the Depth Information in the Point Spread Function of a Defocused Optical System," Technical Report No. 90.02.07, Computer Vision Laboratory, Department of Electrical Engineering, State University of New York, Stony Brook, 1990.

[46] M. Subbarao and M.-C. Lu, "Computer Modeling and Simulation of Camera Defocus," Technical Report No. 92.01.16, Computer Vision Laboratory, Department of Electrical Engineering, State University of New York, Stony Brook, 1992. (Also appears in *Proceedings of Optics, Illumination, and Image Sensing for Machine Vision VII,* SPIE Proceedings Conf. 1822, Boston, Nov. 1992)

[47] M. Subbarao and A. Nikzad, "Model for Image Sensing and Digitization in Computer Vision," *Proceedings of SPIE conference,* OE/90, Boston, Nov. 1990.

[48] M. Subbarao and G. Surya, "Application of Spatial-Domain Convolution/Deconvolution Transform for Determining Distance from Image Defocus," Technical Report No. 92.01.18, Computer Vision Laboratory, Department of Electrical Engineering, State University of New York, Stony Brook, 1992. (Also appears in *Proceedings*

*of Optics, Illumination, and Image Sensing for Machine Vision VII,* SPIE Proceedings Conf. 1822, Boston, Nov. 1992)

[49] M. Subbarao and T.-C. Wei, "Depth From Defocus and Rapid Autofocusing: A Practical Approach," Technical Report No. 92.01.17, Computer Vision Laboratory, Department of Electrical Engineering, State University of New York, Stony Brook. (An abbreviated version appears in *Proceedings of the IEEE Computer Vision and Pattern Recognition,* pp. 773-776, June 1992)

[50] C. F. R. Weiman, "Continuous Anti-Aliased Rotation and Room of Raster Images," *SIGGRAPH* 80, pp. 286-293, 1980.

[51] J. Weng, T. Huang, and N. Ahuja, "Motion and Structure from Line Correspondences: Closed-Form Solution, Uniqueness, and Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 14, No. 3, pp. 318-336, March 1992.

# Appendix A

# Sample Input Description File for *3-D* Object Generation

In this Appendix, a sample input description file for *3-D* object generation using ray casting algorithm is included. The output range image is the “depth stop” machine part as shown in the first row of Figure 2-8, corresponding to the composite tree of Figure 2-5. The first entry in this file is the number of nodes in the tree. This *3-D* object generation program is available as a tool in AVS simulation system.

```
17
depth_stop
0
composite
difference
cone
0. 0. 0.
2. 2. 2.
0. 0. 120.
ds
00
composite
```

```
difference
cone
0. 0. 0.
1. 1. 1.
0. 0. 0.
ds
01
primitive
union
cylinder
0. 90. 0.
4.25 4.25 40.
7. 31. -4.
ds
000
composite
union
cone
0. 0. 0.
1. 1. 1.
0. 0. 0.
ds
001
composite
union
cone
0. 0. 0.
1. 1. 1.
0. 0. 0.
ds
0000
composite
union
cone
0. 0. 0.
1. 1. 1.
0. 0. 0.
ds
```

```
0001
primitive
union
block
0. 0. 0.
35. 6. 24.
-30. -12. -12.
ds
0010
composite
union
cone
0. 0. 0.
1. 1. 1.
0. 0. 0.
ds
0011
primitive
union
cylinder
0. 0. 0.
9. 9. 28.
-39. -9. -14.
ds
00000
composite
union
cone
0. 0. 0.
1. 1. 1.
0. 0. 0.
ds
00001
composite
union
cone
0. 0. 0.
1. 1. 1.
```

```
0. 0. 0.
ds
00100
primitive
union
cylinder
0. 0. 0.
20. 20. 28.
25. -9. -14.
ds
00101
primitive
union
block
0. 0. 0.
4. 32. 28.
23. 9. -14.
ds
000000
primitive
union
cylinder
0. 0. 0.
30. 30. 24.
25. -9. -12.
ds
000001
primitive
union
block
0. 0. 0.
36. 28. 24.
7. 11. -12.
ds
000010
primitive
union
block
```

```
0. 0. 0.
64. 32. 6.
-39. -25. -3.
ds
000011
primitive
union
cylinder
0. 0. 0.
16. 16. 24.
-39. -9. -12.
```

# Appendix B

# The Point Spread Function From Diffraction model

In this appendix, an expression is derived for the point spread function of an optical system based on the Fraunhofer diffraction principle. Most of the derivation here follows from Born and Wolf[4], Chen[6], Hopkins[18], and Subbarao[45]. The derivation here integrates and supplements the relevant material from there for the case of a CCD camera.

Figure B-1 shows the diagram used for image formation where $\vec{q}$

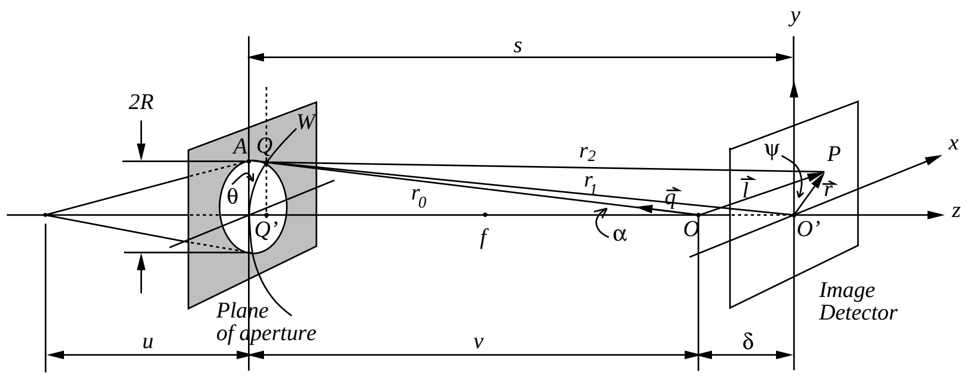



Figure B-1: Image formation using diffraction model.

is the unit vector from $O$ to $Q$. The derivation makes the following assumptions:

(B.1) Point $P$ lies in the neighborhood of point $O'$,

(B.2) $f \gg R \gg \lambda$,

(B.3) $\frac{R^2}{\lambda f} \gg 1$,

(B.4) Huygens-Fresnel principle is used, and

(B.5) Normal incidence of point light source.

Note that, the last assumption is used to simplify the derivations. It can be removed without loss of generality.

In Figure B-1, the field at point $P$ can be expressed as:

$$U(P) = -\frac{i}{\lambda}\frac{A}{r_0}e^{-ikr_0}\iint_W \frac{e^{ikr_2}}{r_2}\,dS \tag{B.1}$$

where $W$ is the wavefront originating at $O$ and $\frac{A}{r_0}$ is the amplitude at $Q$ of the incident wave. Since only small angles are involved, we have, to a good approximation:

$$r_2 - r_0 = -\vec{q}\cdot\vec{l} \tag{B.2}$$

Also, $dS = v^2\,d\Omega$ ($d\Omega$ is the element of solid angle that $dS$ subtends at $O$), the field at $P$ can be approximated by:

$$\begin{aligned} U(P) &\approx -\frac{i}{\lambda}\frac{A}{r_0}e^{-ikr_0}\iint_\Omega \frac{e^{ikr_2}}{r_2}v^2\,d\Omega && \text{(B.3)} \\ &\approx -\frac{iA}{\lambda}\iint_\Omega e^{-ik\vec{q}\cdot\vec{l}}\,d\Omega && \text{(B.4)} \end{aligned}$$

This is the *Debye integral* which expresses the field at $P$ as a superposition of plane waves of different directions of propagation. From the assumptions $f \gg R \gg \lambda$ and $\frac{R^2}{\lambda f} \gg 1$, the Debye integral can be evaluated using the following approach.

Let the coordinates of $P$ and $Q$ be $(x, y, z)$ and $(\xi, \eta, \zeta)$, respectively. In polar notation,

$$\begin{cases} x = r \sin\psi \\ y = r\cos\psi \end{cases} \qquad \begin{cases} \xi = R\rho\sin\theta \\ \eta = R\rho\cos\theta \end{cases} \tag{B.5}$$

where $R\rho$ is the length of $\overline{QQ'}$; $r$ is the length of $\overline{PO'}$; and $0 \leq \rho \leq 1$. Since $Q$ lies on the spherical wave-front $W$, we have

$$\zeta = -\sqrt{v^2 - R^2\rho^2} = -v\left[1 - \frac{1}{2}\frac{R^2\rho^2}{v^2} + \cdots\right] \approx -v\left[1 - \frac{1}{2}\frac{R^2\rho^2}{v^2}\right] \tag{B.6}$$

Therefore,

$$k\vec{q}\cdot\vec{l} = \frac{2\pi}{\lambda}\left(\frac{R}{v}r\right)\rho\cos(\theta-\psi) - \frac{2\pi}{\lambda}z + \frac{1}{2}\left(\frac{2\pi}{\lambda}z\right)\left(\frac{R}{v}\right)^2\rho^2 \tag{B.7}$$

Introducing the dimensionless variables

$$u_0 = \frac{2\pi}{\lambda}\left(\frac{R}{v}\right)^2 z, \qquad v_0 = \frac{2\pi}{\lambda}\left(\frac{R}{v}\right) r \tag{B.8}$$

we have

$$k\vec{q}\cdot\vec{l} = v_0\rho\cos(\theta-\psi) - \left(\frac{v}{R}\right)^2 u_0 + \frac{1}{2}u_0\rho^2 \tag{B.9}$$

From Equation (B.4) and $d\Omega = \frac{R^2\rho\, d\rho\, d\theta}{v^2}$, we have:

$$\begin{aligned} U(P) &= -\frac{iR^2A}{\lambda v^2}e^{i\left(\frac{v}{R}\right)^2 u_0}\int_0^1\int_0^{2\pi} e^{-i\left[v_0\rho\cos(\theta-\psi)+\frac{1}{2}u_0\rho^2\right]}\rho\, d\rho\, d\theta && \text{(B.10)} \\ &= -\frac{2\pi iR^2A}{\lambda v^2}e^{i\left(\frac{v}{R}\right)^2 u_0}\int_0^1 J_0(v_0\rho)\, e^{-\frac{1}{2}iu_0\rho^2}\rho\, d\rho && \text{(B.11)} \\ &\triangleq -\frac{i\pi R^2A}{\lambda v^2}e^{i\left(\frac{v}{R}\right)^2 u_0}\left[C(u_0, v_0) - iS(u_0, v_0)\right] && \text{(B.12)} \end{aligned}$$

where $J_0$ is the *Bessel function of the first kind of order 0*, and

$$C(u_0, v_0) = 2\int_0^1 J_0(v_0\rho)\cos\left(\frac{1}{2}u_0\rho^2\right)\rho\, d\rho \tag{B.13}$$

$$S(u_0, v_0) = 2\int_0^1 J_0(v_0\rho)\sin\left(\frac{1}{2}u_0\rho^2\right)\rho\, d\rho \tag{B.14}$$

$C(u_0, v_0)$ and $S(u_0, v_0)$ can be evaluated in terms of the *Lommel functions*

$$U_n(u_0, v_0) = \sum_{m=0}^{\infty}(-1)^m\left(\frac{u_0}{v_0}\right)^{n+2m} J_{n+2m}(v_0) \tag{B.15}$$

$$V_n(u_0, v_0) = \sum_{m=0}^{\infty}(-1)^m\left(\frac{v_0}{u_0}\right)^{n+2m} J_{n+2m}(v_0) \tag{B.16}$$

and the identities

$$\frac{d}{dx}\left[x^{n+1}J_{n+1}(x)\right] = x^{n+1}J_n(x) \tag{B.17}$$

$$\frac{d}{dx}\left[x^{-n}J_n(x)\right] = -x^{-n}J_{n+1}(x) \tag{B.18}$$

The results are:

$$C(u_0, v_0) = \begin{cases} \dfrac{2\cos\frac{u_0}{2}}{u_0}U_1(u_0, v_0) + \dfrac{2\sin\frac{u_0}{2}}{u_0}U_2(u_0, v_0), & |\dfrac{u_0}{v_0}| \le 1 \\ \dfrac{2}{u_0}\sin\dfrac{v_0^2}{2u_0} + \dfrac{2\sin\frac{u_0}{2}}{u_0}V_0(u_0, v_0) \\ \qquad -\dfrac{2\cos\frac{u_0}{2}}{u_0}V_1(u_0, v_0), & |\dfrac{u_0}{v_0}| > 1 \end{cases} \tag{B.19}$$

$$S(u_0, v_0) = \begin{cases} \dfrac{2\sin\frac{u_0}{2}}{u_0}U_1(u_0, v_0) - \dfrac{2\cos\frac{u_0}{2}}{u_0}U_2(u_0, v_0), & |\dfrac{u_0}{v_0}| \le 1 \\ \dfrac{2}{u_0}\cos\dfrac{v_0^2}{2u_0} - \dfrac{2\cos\frac{u_0}{2}}{u_0}V_0(u_0, v_0) \\ \qquad -\dfrac{2\sin\frac{u_0}{2}}{u_0}V_1(u_0, v_0), & |\dfrac{u_0}{v_0}| > 1 \end{cases} \tag{B.20}$$

The intensity in the neighborhood of the focus can be computed as:

$$
\begin{aligned}
I(u_0, v_0) &= U(P)U(P)^* \\
&= \begin{cases} I_0 \left(\frac{2}{u_0}\right)^2 \left[U_1^2(u_0, v_0) + U_2^2(u_0, v_0)\right], & |\frac{u_0}{v_0}| \leq 1 \\ I_0 \left(\frac{2}{u_0}\right)^2 \{1 + V_0^2(u_0, v_0) + V_1^2(u_0, v_0) & \\ \qquad - 2V_0(u_0, v_0) \cos\left[\frac{1}{2}\left(u_0 + \frac{v_0^2}{u_0}\right)\right] & \\ \qquad - 2V_1(u_0, v_0) \sin\left[\frac{1}{2}\left(u_0 + \frac{v_0^2}{u_0}\right)\right]\}, & |\frac{u_0}{v_0}| > 1 \end{cases}
\end{aligned}
\tag{B.21}
$$

where $I_0 = (\frac{\pi R^2 |A|}{\lambda v^2})^2$ is the intensity at the geometrical focus point $u_0 = v_0 = 0$. This is the point spread function using Fraunhofer diffraction principle.

The relationships between the point spread function in equation (B.21) and the camera parameters can be derived as follows:

Using the geometrical relationship in Figure B-1, we have

$$\overline{O'Q}^2 = \delta^2 + v^2 + 2\delta v \cos\alpha \tag{B.22}$$

For small $\alpha$, $\cos\alpha \approx 1 - \frac{R^2}{2v^2}$. Therefore,

$$\overline{O'Q} = (v + \delta)\sqrt{1 - \frac{\delta R^2}{v(v+\delta)^2}} \tag{B.23}$$

If $\left|\frac{\delta R^2}{v(v+\delta)^2}\right| \ll 1$, $\overline{O'Q}$ can be approximated by

$$\overline{O'Q} \approx (v + \delta)\left[1 - \frac{\delta R^2}{2v(v+\delta)^2}\right] \tag{B.24}$$

From *Rayleigh's tolerance* on defocusing, the amount of focus defect $\Delta$ can be defined as $\Delta = \frac{W_{max}}{\lambda/4}$. Since

$$W_{max} = \overline{O'A} - \overline{O'Q} = s - \overline{O'Q} \tag{B.25}$$

the amount of focus defect can be computed as

$$\Delta = \frac{4}{\lambda}\left(\frac{\delta R^2}{2v(v+\delta)}\right) \tag{B.26}$$

In terms of lens data, we have the following parameters to compute the point spread function $h(r)$ of the camera system:

$$\begin{aligned}
\Delta &= \frac{f^2}{2\lambda F^2}\left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s}\right) && \text{(B.27)}\\
u_0 &= \pi\Delta && \text{(B.28)}\\
v_0 &= \frac{\pi f}{\lambda v F} r && \text{(B.29)}\\
h(r) &= I(u_0, v_0) && \text{(B.30)}
\end{aligned}$$

where $r = \sqrt{x^2 + y^2}$.