A MASSIVELY PARALLEL APPROACH TO MODELING 3D OBJECTS IN MACHINE VISION

By

RONALD ELLISON DANIEL JR.

Bachelor of Science in Electrical Engineering
Oklahoma State University
Stillwater, Oklahoma
1985

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1987

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 1993

OKLAHOMA STATE UNIVERSITY

A MASSIVELY PARALLEL APPROACH

TO MODELING 3D OBJECTS

IN MACHINE VISION

Thesis Approved:

Thesis Advisor

Dean of the Graduate College

ACKNOWLEDGEMENTS

Superquadric Description is an interesting technique for modeling 3D objects in machine vision tasks. It was first proposed by Dr. Alex Pentland, whom I would like to thank for his encouragement and ready supply of references. An improved approach to computing the description was developed by Dr. Franc Solina, whom I would like to thank for the copy of his dissertation that started this line of research.

My first thanks go to Dr. Keith Teague, my principal advisor, for his constant attention, proposal-writing skills, and friendship.Dr. James Baker, Head of the School of Electrical and Computer Engineering, deserves thanks for the constant employment that made my course of study possible. Additional thanks in this area go to Bill Harland of Texas Instruments for supervising the contract which supported two years of this research. Prof. Frank Fallside and Dr. Richard Prager of Cambridge University Engineering Dept. also have my thanks for giving me considerable time away from my work so that I could finish this study. I would also like to thank the Office of Naval Research, the University Center for Energy Research, and the Defense Advanced Research Projects Agency for their support of earlier stages of the research.

Special thanks go to Mike Carter for all the code we wrote and all he taught me during the process. Dr. Charles Bacon, Chairman of my Advisory Committee, deserves special mention for his support, faith, and example. Finally, this dissertation is dedicated to my wife, Laura, and my daughter, Fiona. It has been a long, hard slog to reach this stage, but we made it through together.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF ALGORITHMS

CHAPTER I

INTRODUCTION

Mankind has long used machines to help carry out tasks which we find boring or dangerous. As our machines have become more capable, the range of tasks they can handle has increased. Many of the remaining tasks we would like to automate will require the machines to sense their environment and to respond to it in what we consider an intelligent fashion. One of the most useful senses for humans is sight. With it we can recognize objects from a distance, as well as determine their location and velocity. Even unfamiliar objects can be described and their likely properties, behavior, and purpose inferred. The apparent ease with which we determine the identity, position, and velocity of objects belies the extreme difficulty of the problem. Despite three decades of research, progress in machine vision has been disappointing [44].

Machine vision processing is typically divided into several stages. The first stage, sometimes referred to as the signals domain, operates on images and produces other images as output. Contrast enhancement, edge finding, optic flow, etc. are typical operations in this domain. Later processing is carried out in the symbols domain. In this domain, objects are recognized and their behavioral properties inferred. A transformation must be made to go from the signals domain to the symbols domain. This signals to symbols transformation is the object modeling process.

The rest of this chapter provides a brief survey of object modeling techniques in machine vision. Chapter II describes a particular object modeling technique known as Superquadric Description (SQD), its promise, and the problems with its current implementation. Chapter III describes neural networks, an area that has been the focus of a

great deal of recent research interest. The goal of the research described in this dissertation was to see if an approach inspired by a particular neural network, the Koch network [35], could overcome some of the problems with previous implementations of SQD. Chapter IV describes the first network developed to test this possibility. For simplicity, the initial network operated on 1-dimensional data rather than 2 dimensional images. Chapter V describes the extension to 2D data, while Chapter VI presents my conclusions.

Object Modeling Techniques

As mentioned earlier, Machine Vision (MV) processing is typically divided into two domains: signals and symbols. This simple two-level characterization leaves us with the difficult problem of the transformation from the signals domain to the symbols domain. This transformation is the object modeling process, which is not to be confused with object recognition. We presume that at least a partial model of the object must be built before it is possible to recognize it. The type of model that will be assumed throughout the rest of this paper is similar to 3D mechanical Computer Aided Design (CAD) models. While this model provides the strongest basis for reasoning about the spatial properties of objects, not all vision researchers assume that this sort of model is needed, or even desirable. For example, [67] describes work where objects are recognized from linear combinations of 2D views without ever constructing 3D information. Their approach seems to give very good results, but it is inherently limited to recognition of known objects. While more difficult to obtain, a full 3D object model would be necessary to support inferences into the function and behavior of novel objects.

Many ways have been proposed to model objects in MV. See [6, 16] for surveys of these techniques. Simple methods are rarely useful except for very limited problem domains. An example of such a domain is a simple industrial inspection system where the lighting and background can be strictly controlled, the objects to be recognized are few and have dissimilar appearances, and the objects do not touch or overlap. In such a restricted

environment, a vision system might be able to use an object model that is merely the position, orientation, and area/perimeter ratio of "blobs" in a thresholded image. The gulf between such a restricted environment and the rich environment of the real world is immense, making such an impoverished representation totally unsuitable for general use.

The complex and uncontrollable environment of the real world requires much more powerful object modeling techniques. A few of these more powerful object modeling techniques are surveyed in the rest of this chapter. This chapter does not attempt to survey the large body of MV literature. The interested or naive reader is referred to [3] and [16] for an introduction to the field. More detailed treatments of a smaller number of topics are given in [30] and [45]. A useful survey of 3D object modeling is given in [6].

Object modeling techniques can be characterized by the representation they use for the object, and the method of determining the representation from the image. Representations for the shape of 3D solids can be divided into three classes: surfaces, volumes, and sweeps [3]. Surface representations model the visible portions of curved objects by using parametric patches, such as those used in computer graphics. For polyhedral objects, a planar polygon representation can be used. A great deal of work has been done with polyhedra, starting with the seminal work of Roberts [54], and the relaxation labeling algorithm of Waltz [68]. However, since objects frequently have curved surfaces, a polyhedral representation is not broadly applicable. A natural extension is to use curved surface patches. As noted in [6], this is a much more difficult problem, since the occluding contours of curved surfaces vary continuously as the viewpoint changes. Aspect graphs [36] are an approach which has been developed to reduce this problem. An aspect graph represents those features of an object which appear and disappear from view as the object is viewed from different angles. These different sets of visible features are called aspects. While these aspects can be obtained from a CAD model, it is only possible to obtain them analytically from a relatively limited class of shapes, such as solids of revolution [38]. Another problem with surface-based representations is that they do not

provide any information to support "guesses" as to the shape of the hidden side of the object. While this is safe, humans do make such guesses, using principles of perceptual organization such as symmetry.

The second class of object representation is volumes. Geometric solids such as superquadrics [4] and spherical harmonics [3] fall into this class. Since superquadrics are the representation used in this study, we postpone a discussion of them until the next chapter. Voxels (Volume Elements) are another of these representations. Voxels represent volumes as 3D arrays of small cells, each of which is marked as filled or empty [3]. This representation is popular for medical imaging. In these applications the voxel might contain an index of the transparency to X-rays, rather than a simple filled/empty indication.

The disadvantage of this approach is that the voxel must be very small to represent curved surfaces closely. This makes the storage requirement very large. Octrees can be used to reduce this storage requirement. If 8 voxels form a 2 x 2 x 2 array, and all of them have the same state (filled or empty), they are replaced by a single, larger, voxel that is filled or empty. Nonhomogeneous arrays are marked as such, and the smaller voxels are retained as children of the node. This process is repeated hierarchically until the object is represented by a spatial occupancy tree. The size of the volume covered by a leaf in the tree depends on its level in the tree. This can achieve significant compression, but at the cost of more complex algorithms to determine what shape is represented by a particular tree.

The third class of representations is known as sweeps. Generalized cylinders [8] are by far the most popular representation in this class. These represent a volume by sweeping a closed two-dimensional curve along a (possibly open) three-dimensional curve. The shape of the two-dimensional curve can vary as a function of its position along the axis. Quite elaborate shapes may be described by this representation, but recovery can be very difficult because there are few constraints. Because of this problem, generalized cylinders with particular constraints are frequently used [41].

Whichever technique is used to model primitive objects, there will always be objects in a scene too complex to model with a single primitive. The natural solution is to use multiple primitives to describe the shape. There are two main approaches to this problem. The first, and simplest, is composition. In this scheme a complex object is represented as the union of the primitive chosen as the basis for modeling shapes. This allows a much larger class of objects to be modeled, but it has limitations. Relatively simple objects with holes are difficult to describe without large numbers of primitives. A more general technique for representing complex objects is Constructive Solid Geometry (CSG) [10, 53, 64]. Primitives are combined using the set operations of union, intersection, and difference. The model of an object is a CSG tree whose leaves are primitives and whose internal nodes are the set operators. This is shown below in figure 1. Nodes of the CSG tree may also be transformations which scale, translate, or rotate the subtree of that node. Given an adequate set of primitives, almost any object can be modeled using CSG, but there are still difficulties. CSG representations are not unique, i.e., a particular object can be modeled using several different CSG trees. Another problem is concerned with recovery. CSG allows the difference and intersection operators. Thus, we must recover the shapes of primitives that, in some sense, are not really in the image.

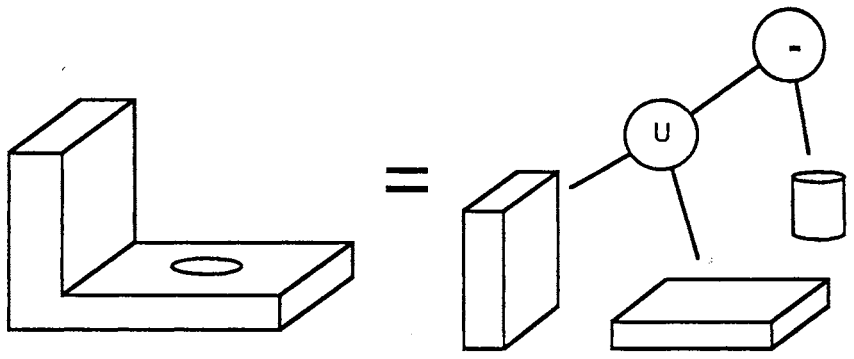

Figure 1: Simple Object and its CSG Representation

Whatever object representation is chosen, a procedure must be developed for recovering the parameters describing the object from an image. This recovery procedure is quite difficult, as it forces a compromise between the generality of the models and the ease of extracting the necessary information from the image. Early methods used high-pass filters to extract edges from the images. These were then used to try to model objects using a wire-frame representation [54, 68]. The deficiencies of wire frame representations soon became evident, as did the difficulty in extracting high-quality edge information from real images. However, work still continues in this area. Advances in edge operators have made these more robust, although still far from perfect. Wireframe representations are no longer used in any but the simplest systems. Instead, the edges are grouped using principles of perceptual organization such as adjacency, co-linearity, and parallelism. Constraints are applied to these groupings, such as the viewpoint consistency constraint [41], or constraints on likely shapes [7, 31]. These researchers are attempting to extract generalized cylinders which have various constraints on their shapes, rather than wireframe representations of polyhedra.

Edges are not the only information that can be extracted from images. There is a large family of algorithms, collectively known as Shape-From-X techniques, which estimate the shape (surface normals) or depth (distance to) imaged objects. Binocular disparity is the best known of these techniques, but there are a host of others that use clues such as shading [32, 39], texture [33], specular reflections [25], focal gradients [48], or motion [66]. These techniques do not provide the quality of information that is obtainable from edges, so research is underway on combining the output of multiple shape-from-x techniques [46]. While this work is underway, researchers are also using laser rangefinder images. These provide much more accurate depth information, but still confront their users with the problems associated with image processing.

Given depth or shape information, it should be possible to estimate the size, shape, and position of objects in the image. Edge-based approaches can operate on depth images,

and are frequently used as a pre-processing step in other approaches. If an object model with a simple analytical representation is used, there are several ways to recover the parameters controlling its shape, size, and position. One is to iteratively adjust the parameter estimates so as to minimize the difference between the measured values and those predicted by the current parameter estimates. This is the approach used in this paper. A second approach is to use a linear (or non-linear) regression, where the parameters can potentially be estimated by the use of direct matrix operations rather than an iterative procedure. A third technique is based on solving the differential equations for Lagrangian dynamics, where the image data exerts simulated forces on the estimated objects to pull them into position and shape [63].

CHAPTER II

SUPERQUADRICS AND SUPERQUADRIC DESCRIPTION

Superquadrics

The superquadric (SQ) is an interesting geometric primitive that has been proposed for modeling objects in machine vision, as well as in CAD/CAM (Computer Aided Design / Computer Aided Manufacturing) [4, 5, 47, 49, 13]. Unit SQs are a 2-parameter family of geometric solids which include spheres, cubes, cylinders, and octahedrons as special cases. The two parameters, e_1 and e_2, control the roundness vs. squareness of the longitudinal and latitudinal cross-sections. Figure 2 shows unit SQ solids for e_1 and e_2 = 0.2, 1.0, and 2.0.

There are implicit and explicit forms for the equations that define SQs. The explicit equation (1) gives the x, y, and z coordinates of points on the surface of the SQ as a function of the latitude and longitude, η and ω. The notation $C_\eta^{e1} S_\omega^{e2}$ means $\cos^{e1}(\eta)$ $\sin^{e2}(\omega)$. Equation (2) gives the x, y, and z components of the surface normal of the SQ as a function of η and ω. Note that a dual relationship exists between (1) and (2).

The implicit form (3) tells if an x, y, z point is outside, inside, or on the surface of a SQ defined by particular parameter values of $\underline{a}$, $\underline{x}$, and $\underline{e}$. If F = 1, the point is on the surface. If it is less than 1, the point is inside, if greater than 1 the point is outside. For this reason, the implicit equation is also known as the inside/outside function. Equations (1 - 3) explicitly show the correct treatment of signs that is implicit in [4, 47, 58, 14]. The sgn(x) function returns the sign of its argument.

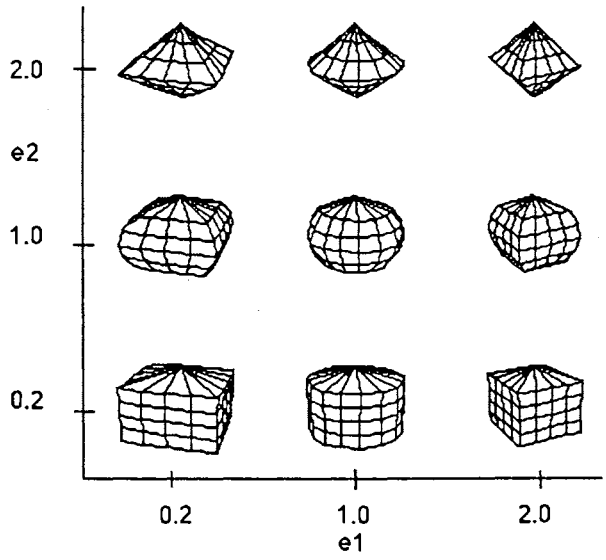

Figure 2: Unit Superquadrics vs. e_1 and e_2

Unit SQs centered at the origin are not particularly useful, so both the explicit and implicit equations show the additional parameters needed to place the center of the SQ at an arbitrary position x_0, y_0, z_0. The size of the SQ is given by a_1, a_2, and a_3, which are scale factors in the x, y, and z directions. SQs may be given arbitrary orientations through the use of the standard rotation matrices used in computer graphics [20], however, we will not be considering rotations in this study.

$$\underline{X}(\eta,\omega) = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} a_1\, \mathrm{sgn}(C_\eta^{e1}\, C_\omega^{e2})\, |C_\eta|^{e1}\, |C_\omega|^{e2} \\ a_2\, \mathrm{sgn}(C_\eta^{e1}\, S_\omega^{e2})\, |C_\eta|^{e1}\, |S_\omega|^{e2} \\ a_3 \quad \mathrm{sgn}(S_\eta^{e1}) \quad |S_\eta|^{e1} \end{pmatrix} \tag{1}$$

$$\underline{N}(\eta,\omega) = \begin{pmatrix} 1/a_1\, C_\eta^{2-e1}\, C_\omega^{2-e2} \\ 1/a_2\, C_\eta^{2-e1}\, S_\omega^{2-e2} \\ 1/a_3\, S_\eta^{2-e1} \end{pmatrix} \tag{2}$$

$$F(\underline{x};\, \underline{a},\, \underline{x}_0,\, \underline{e}) = \left(\left(\frac{|x-x_0|}{a_1} \right)^{\frac{2}{e_2}} + \left(\frac{|y-y_0|}{a_2} \right)^{\frac{2}{e_2}} \right)^{\frac{e_2}{e_1}} + \left(\frac{|z-z_0|}{a3} \right)^{\frac{2}{e_1}} \tag{3}$$

Figure 2 and equations (1 - 3) describe what are actually known as SQ ellipsoids. SQ toroids and SQ hyperboloids of one and two sheets also exist [4], but are not considered in this study. Whenever we refer to a SQ, we are actually discussing SQ ellipsoids.

An important advantage of using SQs to model objects is that a single equation can be used to express a wide variety of shapes simply by varying a few parameters. This should make recovery easier since we don't have to know if the object we are attempting to model is a sphere, cube, cylinder, etc. before we try to model it. This "chicken and egg" problem is one of the major difficulties in image segmentation, so the potential for avoiding it makes SQs especially attractive.

The expressive range of SQs can be considerably extended by introducing deformations such as bending, tapering, or twisting [5, 58]. These deformations could also vary with time [62, 42]. Figure 3 shows some deformed SQs, however, this study will not attempt to recover deformed SQs.

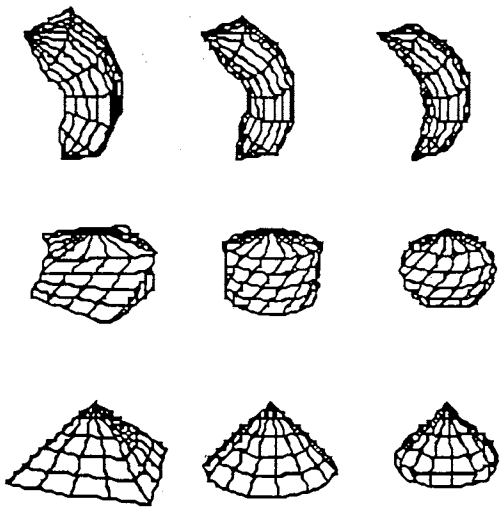

Figure 3: Deformed Superquadrics

An important disadvantage of SQs is the non-uniqueness of descriptions. By rotating and scaling, two different sets of SQ parameters can describe the same object. This can be seen in the first and last columns of figure 2.

Superquadric Description (SQD) is a technique to describe physical objects as CSG combinations of SQ primitives [13]. One of the biggest advantages of SQD is that its models, while very expressive, are compact. Only a few parameters are needed to represent quite complex shapes. This eases the job of matching recovered models with known models for the purpose of recognition. A second advantage to SQD is that the parameters control the shape in a very obvious manner. This is in contrast to spherical harmonics, a volumetric primitive mentioned in the previous chapter. While this is not very important for recovery of models, it is an important advantage for the humans who must construct the models to be recognized. A third advantage of SQD is that it naturally decomposes complex objects into parts. Furthermore, this decomposition seems very similar to those that humans naturally make [47].

While SQD has several important advantages, it also has several disadvantages. Even deformed SQs are only a subset of the models that can be expressed by generalized cylinders. This disadvantage is mitigated by two factors. First, SQs form a very useful subset of generalized cylinders. Second, as was mentioned in the previous chapter, the full power of generalized cylinders is rarely exploited. Another disadvantage of SQD is that CSG shares the non-uniqueness problem of SQs. While the non-uniqueness problem of SQs can be mitigated by constraining the acceptable values of the shape exponents, overcoming the non-uniqueness of CSG is still a topic of research.

Superquadric Parameter Estimation

Assuming we wish to model the objects in an image using combinations of SQ primitives, how do we recover the parameters controlling the size, shape, position, and orientation of these SQs from an image? The original method was suggested by Pentland in

[47]. This utilized the dual relation between the explicit equations (1) and (2). Surface normals for objects in images were estimated using shape-from-X procedures [30, 32, 39]. The dual relation between (1) and (2) was then used to formulate a linear regression to solve for the size, orientation, and shape parameters of the underlying SQ shapes. However, this technique proved too difficult to extend to deformed objects in general position. Pentland later suggested another technique which was analytically more tractable, but involved a computationally expensive search of the SQ parameter space [50].

A more promising approach was suggested in [9]. This utilized the implicit equation (3). Recall that if an xyz point lies on the surface of the SQ described by the parameters $\underline{a}$, $\underline{x}$, $\underline{e}$, then F = 1. If the point is outside the SQ, F > 1, while if it is inside, F < 1. Boult and Gross used the square of the inside/outside function in a minimization procedure where all the parameters were iteratively adjusted to achieve the best fit to the 3-D data points. The 3D data points can be obtained from an intensity image by integrating the output of Shape-From-X procedures, but will usually be obtained from laser rangefinder data or other active sensors. This technique proved quite promising, although they reported problems recovering cylindrical objects.

Solina was responsible for two important changes to the minimization-based approach to estimating the SQ parameters. First, and most importantly, he modified the inside/outside function to achieve much better recovery of cylindrical objects [58, 59]. His version of the inside/outside function is given below in equation (4). Note that he has added the outermost exponent e_1.

$$F_S(\underline{x}, \underline{a}, \underline{x_0}, \underline{e}) = \left(\left(\left(\frac{|x-x_0|}{a_1} \right)^{\frac{2}{e_2}} + \left(\frac{|y-y_0|}{a_2} \right)^{\frac{2}{e_2}} \right)^{\frac{e_2}{e_1}} + \left(\frac{|z-z_0|}{a_3} \right)^{\frac{2}{e_1}} \right)^{e_1} \qquad (4)$$

Solina explains that the

> ... additional exponent e_1 does not change the shape of the superquadric surface itself but is necessary if the function is used for shape recovery with a least-squares minimization method. The additional exponent ensures that, independent of the current value of e_1, points at the same distance from the superquadric surface have

the same value of $F(\underline{x})$. Consider, for example, a cylindrically shaped object where $e_1 = 0.1$ and $e_2 = 1$. Then the third term in (4) is

$$\left(\frac{z}{a_3}\right)^{\frac{2}{e_1}} = \left(\frac{z}{a_3}\right)^{20} \tag{5}$$

Because of the large exponent, very small deviations of z from a_3 will be greatly amplified. The outermost exponent e_1 in (4) cancels out e_1 in (3) and ensures that deviations always have a quadratic weight. Minimizing the inside/outside function without this correction does not give consistent solutions [58, pp. 19-20].

Later work by Solina's colleagues at the University of Pennsylvania [24] provided a mathematical justification for Solina's intuitive understanding of the need for the outermost exponent.

The actual objective function used by Solina incorporates rotations and deformations. It also incorporates a term that insures that the recovered model tightly fits the data points. Consider viewing a cylinder from an angle where one end and part of the shaft is visible. Data points for the other end will not be available, so the size parameter that gives its length could assume any value greater than or equal to the actual length. To avoid this problem, Solina added the term $\sqrt{a_1\, a_2\, a_3}$ to the objective function. This penalizes the recovery of volumes larger than the data actually supports. This also introduces the trivial solution where $\underline{a} = 0$, so the $\underline{a}$ must be constrained to be greater than 0. The cost function he used was

$$\min \sum_{i=1}^{N} \left[\sqrt{a_1\, a_2\, a_3}\,(1 - F_s\,(\underline{x}_{w_i}, \underline{a}, \underline{x}_0, \underline{e}, \underline{\theta}\,))\right]^2 \tag{6}$$

where $\underline{x}_{w_i}$ are the N 3D points in world coordinates, and $\underline{a}$, $\underline{x}_0$, $\underline{e}$, $\underline{\theta}$ are the parameter vectors for the scaling factors, position, shape, and orientation, respectively, of the SQ. The modification of the inside/outside function for rotations has not been shown, since it is not considered further in this study.

The problem of recovering the parameters of a SQ in general position is cast as a minimization problem where we must estimate the values of the 8 parameters (3 for position, 3 for size, and 2 for shape) that best fit the N 3-D data points. Although techniques exist to estimate the 3-D coordinates of points on the surface of objects in intensity images [32, 39, 30], laser rangefinder data is being used by almost all researchers in this area due to its higher quality.

Solina's second major contribution to estimating the SQ parameters was to allow his minimization to perform segmentation at the same time it was estimating the SQ parameters. In other words, the data points that make make up a particular object, and the parameters describing that object, are determined simultaneously. For scenes with multiple objects, Solina's minimization proceeds by trying to fit all the points to a single set of parameters. Those data points which fit poorly are temporarily discarded. If the data points fit reasonably well once the parameter estimates have been changed, they are restored to the set of points that are considered to be accounted for by one SQ. Once the minimization succeeds for a single object, it is started again on the rejected points. This approach is detailed below in Algorithm 1.

ALGORITHM 1

SOLINA'S MULTIPLE OBJECT RECOVERY METHOD

```
1   INPUT data points x1 ... xn, initial parameter estimates λ0
2   set M0 = 0,  Nacc = N
3   FOR j = 1 to N
4       M0 = M0 + (R(xj, λ0))^2
5   FOR k = 0, 1, ... REPEAT
6       compute λk+1
7       set threshold T = f(Mk)
8       DO
9           set P = 0, Mk+1 = 0
10          FOR j = 1 to N
11              IF (R(xj, λk))^2 > T THEN
12                  Mk+1 = Mk+1 + (R(xj, λk))^2
13                  P = P + 1
14          T = 2 T
15      UNTIL P > 0.75 Nacc
16      set Mk+1 = Mk+1 / P
17      IF Mk+1 < Mk THEN
18          accept Λk+1
19          set Nacc = P
20      ELSE MK+1 = Mk
21  END UNTIL (Mk small enough or changes are statistically meaningless)
```

This element of data selection is very important because it offers a means of escaping from the "chicken and egg" dilemma that is posed by segmentation. Other researchers typically determine the data points that are believed to come from a single object, then fit a geometric model to them, taking no account of the possibility of outliers or multiple objects [19]. Once the preprocessing step has made a decision, there is no way to reverse it, even if the modeling stage is capable of indicating a problem.

The potential of SQD is quite exciting, but there are several problems with Solina's implementation. One is speed. Solina used a serial minimization technique (multigrid Levenberg-Marquardt). He reports execution times of about 20 sec. on a VAX 11/785 for relatively small data sets ($N \approx 250$). Since typical image sizes are 256^2 or 512^2, this technique is somewhat limited. Solina suggested that the minimization technique could

easily be implemented in parallel. As part of the research described in this dissertation, a parallel version of Solina's algorithm was implemented [14]. This work, which is believed to be unique, experimentally verified Solina's speculation about the potential parallelism of the method. It confirmed that the evaluation of the objective function and its partial derivatives for each data point can be carried out in parallel. It also noted that, for typical numbers of data points relative to the number of parameters, the repeated evaluation of the objective function and its partial derivatives dominates the cost of the computation. Evaluating them in parallel will exhibit almost perfectly linear speedup, even if the linear system solution required elsewhere in the Levenberg-Marquardt algorithm is carried out serially.

A more important limitation of Solina's method is that his approach for handling multiple objects in a scene suffers from two disadvantages. One is that it makes execution time proportional to the number of objects in a scene. More importantly, it severely limits the number of objects that can be modeled in a scene. This is quite a handicap. Early in this course of research I implemented Solina's algorithm and tested its discriminatory capability. For the cases tested, it modeled one object well, but could only occasionally disambiguate two objects and model them correctly. I was never able to get it to correctly model 3 objects, but did not test it on as many cases as Solina, whose thesis displays some scenes with three objects being modeled.

An important limitation of Solina's algorithm is that it does not exploit the very powerful property of coherence in the image data. If a data point is on the surface of a particular SQ, it is quite likely that its neighboring points will also be on the same SQ. However, Solina's technique treats all the data points as if they were independent. If a way could be found to exploit the powerful property of image coherence, it should be possible to extend SQD to scenes with very large numbers of objects. This was a major goal of this study, which investigated using a neural network technique to exploit image coherence.

During the time the research described in this paper was conducted, other researchers continued looking into the problem of recovering SQ parameters from images. Two camps reported their results in the literature. Interestingly, neither camp extended the function minimization approach of Solina. Instead they both based their procedures on Lagrangian dynamics, inspired by the work presented in [63]. The first group was led by Dimitri Terzopoulos, primary author of [63], the second group was led by Alex Pentland. We will examine the approach taken by Terzopoulos' group first, then briefly note how Pentland's method differs.

The first technique, *deformable superquadrics*, was developed by Demetri Metaxas and Dimitri Terzopoulos [61, 62, 42]. It represents the shape of the object, $\mathbf{p}(t)$, as the sum of a reference shape, $\mathbf{s}(t)$, and local deformations from that shape, $\mathbf{d}(t)$. For deformable superquadrics, $\mathbf{s}$ is a SQ and $\mathbf{d}$ is represented using finite element techniques. The parameters controlling the size and shape of the SQ, along with any parameters controlling global deformations such as bending, tapering, or twisting, are collected into a parameter vector $\mathbf{q}_s$. The local deformations are represented using finite element basis functions. The SQ is tessellated and a displacement vector, $\mathbf{q}_i$, is associated with each node i at the corners of the elements. Collecting all the displacements into a vector $\mathbf{q}_d = (..., \mathbf{q}_i, ...)^T$, the local displacement can then be expressed as $\mathbf{d} = \mathbf{S}\mathbf{q}_d$, where $\mathbf{S}$ is the shape matrix whose entries are the finite element basis functions.

The shape of the deformable SQ, $\mathbf{p}$, is in model-centered coordinates. For machine vision problems, the objects will be specified in a different, world, coordinate system. The set of x,y,z points on the surface of the deformable superquadric are denoted by $\mathbf{p}$ in model-centered coordinates, and by $\mathbf{x}$ in world coordinates. The two systems are related by the equation

$$\mathbf{x} = \mathbf{c} + \mathbf{R}\mathbf{p}, \tag{7}$$

where $\mathbf{c}(t)$ is the position of the center of the SQ over time, and $\mathbf{R}(t)$ is the rotation matrix.

The SQ parameters, the local deformation parameters, and the coordinate system transformation parameters are collected into a vector of degrees of freedom $\mathbf{q} = (\mathbf{q}_c^T, \mathbf{q}_\theta^T, \mathbf{q}_s^T, \mathbf{q}_d^T)^T$, where $\mathbf{q}_c$ is $\mathbf{c}$ and $\mathbf{q}_\theta$ is the vector of rotational coordinates of the model. The goal of fitting the image data to the deformable SQ model is to recover this vector of parameter estimates. Metaxas and Terzopoulos carry out this recovery in a physically based way by introducing mass, damping, and a deformation strain energy through the mechanism of Lagrangian dynamics. The equations of motion take the form:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{g}_q + \mathbf{f}_q, \tag{8}$$

where $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ are, respectively, the mass, damping, and stiffness matrices, $\mathbf{g}_q$ are inertial forces from the coupling between the local and global degrees of freedom, and $\mathbf{f}_q(t)$ are the forces arising from the degrees of freedom of the model. For machine vision, the force vector $\mathbf{f}_q$ is derived from the image data. For this application, $\mathbf{M}$ can be discarded since we are modeling image regions, not masses, and $\mathbf{D}$ can be assumed diagonal and constant over time. This decouples the equations, so $\mathbf{g}_q$ also vanishes. Using a simple first-order Euler method to integrate (8), the update step becomes:

$$\mathbf{q}^{(t+\Delta t)} = \mathbf{q}^{(t)} + \Delta t \mathbf{D}^{-1}\left(\mathbf{f}_q^{(t)} - \mathbf{K}\mathbf{q}^{(t)}\right) \tag{9}$$

The stiffness matrix $\mathbf{K}$ determines the elastic properties of the model, which arise from a spline deformation energy. The derivation of $\mathbf{K}$ is not germane to this presentation, but can be found in [61, 62]. We merely note in passing that during the solution of (9) is is not necessary to assemble $\mathbf{K}$ in its entirety. Instead, it is possible to compute $\sum_j k_{ij}\mathbf{q}_{d_j}$ for each node i in an element-by-element fashion.

The forces $\mathbf{f}_q$ which drive the dynamic system of (8) are obtained from the visual images. Techniques for generating these forces from different types of imagery have been described in [63]. As an example, for 3D data points such as those from a laser rangefinder, we can define a long-range force

$$\mathbf{f}(u_r) = \beta\|\mathbf{r} - \mathbf{x}(u_r)\| \tag{10}$$

based on the separation between a datapoint **r** in space and the point of influence u_r of the force on the model's surface, and where β controls the strength of the attraction. The point of influence needs to be the model point nearest to the datapoint. There is no closed form solution to determine the point u_o on the surface of a deformable SQ that is closest to an arbitrary point **r** in space. While Metaxas and Terzopoulos describe several procedures for determining which node on the model is closest to **r**, their reported experimental results are from algorithms where the matching is done by exhaustive search. This has a high order of complexity, $O(mn)$ where m is the number of datapoints and n is the number of nodes, but it is simple and robust.

Pentland's method [51] was also based on the research reported in [63]. Rather than representing a shape as the sum of a reference SQ and local deformations, Pentland's work models non-rigid SQs that are subjected to periodic forcing functions which induce modal deformations. While this may ease recovery of symmetrically deformed objects, the modal deformations are global, which will limit their utility.

CHAPTER III

NEURAL NETWORKS

The current implementation of SQD is not the only machine vision technique with significant limitations. Progress in machine perception has been disappointing, given the amount of effort put into the problem over the last three decades. Recently, there has been an explosion of research interest in the area known as neural networks. These networks have demonstrated surprising capabilities in pattern recognition tasks. Inspired by the organization of the central nervous system, neural networks consist of many units with dense interconnections between neighboring units. While each unit performs a simple computation, the whole ensemble performs a complex computation which emerges from the interactions of all the units. The complex computation performed depends upon the topology and strength (weight) of the interconnections, as well as the function computed at each unit. The connection weights can either be designed in advance or learned from representative data samples.

The field of neural networks actually began over 20 years ago under the name of perceptrons. Interest died in the field after a critical review of the capabilities of perceptrons was published [43]. The reawakening was due, in large measure, to a new learning algorithm that overcame the problems so effectively exposed by Minsky and Papert. This algorithm, the backward error propagation algorithm [55], allows networks of perceptrons to be extended from single layers to multiple layers. Single layers of perceptrons can only classify linearly separable problems. Extending them to multiple layers allows the linear decision boundaries to be combined and used as higher-order decision boundaries.

A neural network has two components - its architecture (or topology) and its learning algorithm. These are usually intimately related, so the ensemble is frequently referred to as a particular type of network without confusion. For instance, a multiple layer perceptron architecture trained by the backward error propagation algorithm is known as a backpropagation network. Other learning algorithms have been developed for multiple layer perceptrons [17, 34], since the backward error propagation algorithm is slow to learn. However, these learning algorithms will sometimes fail to learn a classification that can be learned by the backward error propagation algorithm, so the simple backpropagation network remains the most commonly used neural network at this time.

Multiple layer perceptrons are not the only network architecture capable of learning. Some of the other major ones at this time are ART (Adaptive Resonance Theory) [12, 23], Kohonen's Self-Organizing Maps [37], and some varieties of Hopfield nets [26, 27, 28]. Not all networks learn, several have all their weights set in advance. The major networks with weights designed in advance are other varieties of Hopfield nets [29, 60].

While neural networks have only recently regained widespread popularity, the literature has already become too large for a survey of the entire field. Instead, we will look at the general classes of neural networks and at a representative network from each of these classes. The interested reader should consult [40] or [56] for an more detailed introduction to the field. After this broad introduction, we will pay particular attention to one particular network, the Koch network [35], that was the inspiration for the research in this study.

Neural Network Characterization

As mentioned above, networks can be characterized by whether their weights are learned or fixed. Within the class of learning networks there are two subclasses, supervised and unsupervised. Networks can also be characterized by the type of problem they are intended to solve. The most common problems are classification, associative memory, and

optimization. These problems are very similar. To show this similarity, we must first define what is meant by a classification problem.

A pattern vector, $\underline{x}$, is a vector of measurements, x_i, which contain all the measured information available about the underlying pattern.

$$\underline{x} = (x_1, x_2, \ldots, x_n)^T \tag{11}$$

A pattern class is a category determined by some given common attributes of a set of pattern vectors [65]. Classification is the problem of assigning a new pattern vector to an existing pattern class. Determining the pattern classes given a representative set of pattern vectors is another problem, known as clustering.

Associative memory is the problem of retrieving the stored memory that most closely matches a partial memory, which is presented as a search key. This can be viewed as a classification problem where the classes are a complete memory, and the pattern vectors presented for classification are the partial memories.

In neural networks, both classification and associative memory tasks are usually performed by minimizing an error measure between the input pattern vector and the output pattern class. This minimum seeking property can be used directly to solve minimization problems. The rest of this chapter discusses the three general classes of neural networks mentioned above. Unless mentioned otherwise, we will assume that the network will be used to solve a classification problem.

Supervised Learning Networks

The most popular class of neural networks are those whose weights are trained by a supervised learning procedure. A supervised network is trained by repeatedly presenting input patterns and the desired output. The network learns the mapping between inputs and outputs. This is called supervised learning because the correct output for each input pattern in the training set is known. The backpropagation network [55] is the dominant network in this class, so we will look at its operation as an example of this class of network.

A backpropagation network is constructed of layers of units. A network will have input, output, and some number of hidden layers. Networks with more hidden layers can model more complex regions, but will have a longer training time than a network with fewer hidden layers and the same number of hidden units. Networks with more than 2 hidden layers are rare, because networks with 2 hidden layers are capable of performing any desired input-output mapping. Unfortunately, the number of hidden units required may tend towards infinity. Also, there is no guarantee that the input-output mapping can be learned.

Each unit in a layer has weighted connections to all the units in the previous layer. A unit has an internal state, known as its activation, a. The activation is the weighted sum of the outputs, o_j, from the previous layer:

$$a_i = \sum_j o_j w_{ij} \tag{12}$$

The output of a unit is a function of its activation, usually the nonlinear sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

Input and output units may use a linear activation function, but it is important that at least the units in the hidden layer(s) use a non-linear function [55]. If all the layers use a linear activation function, then superposition applies and all the layers can be reduced to one linear layer. This means that the network will only be able to classify linearly separable patterns.

The discussion above describes how an output pattern is generated from an input pattern. This procedure, known as the forward pass, is essentially the same for a very wide range of neural networks. It did not describe how the weights are modified, which is how the network learns the desired mapping between input and output patterns. To train a backpropagation network, an input pattern from the training set is presented to the network,

and an output pattern generated. The desired output is then presented to the output units. Each unit computes an error signal, δ_j:

$$\delta_j = (t_j - o_j)\, f_j'(a_j) \tag{14}$$

where j is the index of the unit in the output layer, o_j and t_j are the output and target values, f_j' is the derivative of that unit's activation function, and a_j is the activation of that unit (12). The weights between the last hidden layer and the output layer are updated according to the rule:

$$\Delta w_{ij} = \eta\, \delta_j\, o_i \tag{15}$$

where Δw_{ij} is the change in the weight between unit i in the previous layer and unit j in the output layer, and η is a positive scalar constant less than 1.0 known as the learning rate. The larger the value for η, the faster the network learns, but excessive values for η will make the learning procedure fail to converge.

The weights between hidden layers, or between the input layer and the first hidden layer, are modified by a similar procedure. However, the determination of the error signal is different, since we do not have a target value for the outputs of the hidden units. Instead, equation (16) is used for the error signal:

$$\delta_j = f_j'(a_j) \sum_k \delta_k\, w_{kj} \tag{16}$$

where the index k ranges over all the units in the subsequent layer. The update rule remains the same as (15).

The learning procedure described by equations (14 - 16) implements a simple gradient descent algorithm that minimizes the squared error between the output and target patterns in the training set. This simple minimization algorithm leads to long training times. Therefore, a considerable amount of research has been devoted to incorporating more sophisticated minimization techniques into the layered structure of the network. The technique that is receiving the most interest at this time is conjugate-gradients [34],

although other techniques certainly have their adherents [17]. Another problem with the simple backpropagation network is that there is no way to determine in advance how many units should appear in each hidden layer. Network architectures that add new units during learning are being studied in order to overcome this problem [18].

Unsupervised Learning Networks

In contrast to supervised learning, unsupervised networks are not presented with the desired output for each input pattern. Instead, the network is expected to perform a self-organization in order to classify patterns into a number of categories. The Adaptive Resonance Theory networks (ART1, ART2, ART3) [12, 23] and Kohonen's Self-Organizing Maps [37] are the dominant networks in this class. We will examine the operation of the ART1 network as an example of this class of neural network. The structure of this network is shown below in figure 4.

The ART1 network is described by a set of differential equations. It has a moderately large number of parameters, as well as fast and slow learning modes. However, by fixing some of the parameters and using the fast learning mode, the behavior of the network is considerably simplified. The explanation of the network given below follows that presented in [40]. The interested reader is referred to [12] and [23] for the full details of the network.

The network is composed of two layers, or fields. These are indicated in figure 4 by the boxes labeled F_1 and F_2. The number of units, N, in F_1 is set by the size of the input pattern vectors. The number of units, M, in F_2 is the number of pattern classes. For the ART1 network, the input and output vectors are binary. This restriction on the input vector is removed in ART2.

The units x_i in F_1 are fully interconnected to the units y_j in F_2 through two interconnection matrices, B and T, which are not shown in the figure. B holds the weights

used in the bottom-up pass, while T holds those used in the top-down pass. At time t = 0, these are initialized to:

$$T_{ij}(0) = 1 \tag{17}$$

$$B_{ij}(0) = \frac{1}{1 + N} \tag{18}$$

Their values will be updated as the network begins to learn its classifications. The T matrix encodes the exemplars for the pattern classes. This is known as the Long Term Memory (LTM) of the network. The Short Term Memory (STM) is the pattern of activation in the two layers.

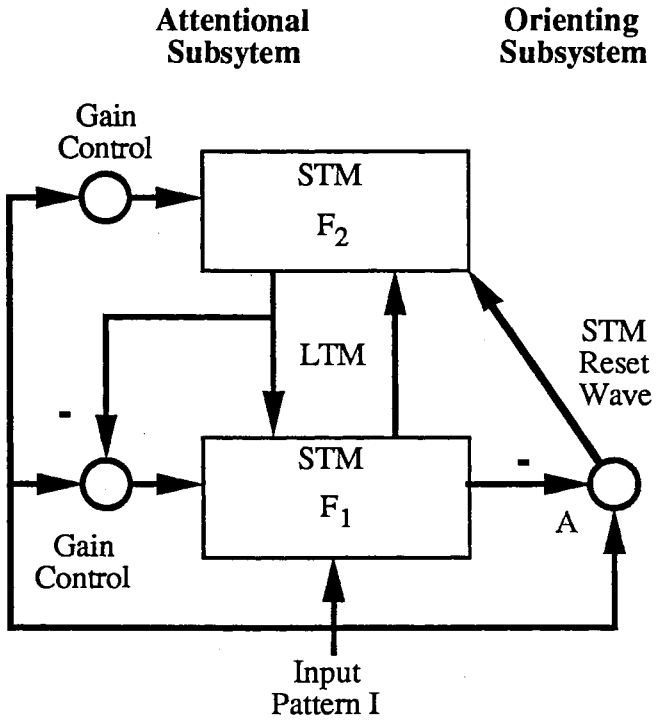

Figure 4: ART1 Network (from [12]).

When an input pattern, I, is presented to an ART1 network, the units in F_1 assume an initial pattern of activations, X, which is equal to I. Recall that the input pattern is

binary, so the activations x_i are too. The activations of the output nodes are set to the weighted sum of the activations in F_1:

$$y_j = \sum_{i=1}^{N} B_{ij}(t)\, x_i \;,\; 1 \le j \le M \tag{19}$$

The maximum element in F_2 is then chosen through lateral inhibition. Ties are broken in favor of the unit with the lowest index j. This leaves a single output unit, y_j, active.

The maximum element identifies the class to which the pattern is assigned. The next step in the operation of the network is to test if the input pattern matches the class exemplar well enough to be considered as a member of the class, or if a new classification should take place. This decision is governed by the vigilance parameter, ρ. Recall that the T matrix encodes the class exemplars. We compute the ratio, q, of the number of active elements in the exemplar to the number of active elements in the input pattern:

$$\|X\| = \sum_{i=1}^{N} x_i \tag{20}$$

$$\|T\, X\| = \sum_{i=1}^{N} T_{ij}\, x_i \tag{21}$$

$$q = \frac{\|T\, X\|}{\|X\|} \tag{22}$$

In the equations above, j is the index of the active unit in F_2. The ratio, q, is compared to the vigilance parameter, ρ. If $q > \rho$, the input is considered close enough to the exemplar. The class exemplar is then modified to account for the new member of the class, according to equation (23). The bottom up weights are also modified to enhance the classification of the input pattern, I, to the same class, j, according to (24).

$$T_{ij}\,(t+1) = T_{ij}\,(t)\, x_i \tag{23}$$

$$B_{ij}(t+1) = \frac{T_{ij}(t)\, x_i}{\frac{1}{2} + \sum_{i=1}^{N} T_{ij}(t)\, x_i} \tag{24}$$

If, on the other hand, $q \leq \rho$, the input pattern is considered to be too far from the exemplar. The active unit in F_2 is disabled and the input pattern is presented again. This is the STM Reset Wave in figure 4. The process repeats until the pattern matches an existing exemplar, or a new exemplar is created. The new exemplar is identical to the input pattern.

The ART1 network is limited to binary patterns, and is rather sensitive to noise. These problems have been reduced, though not eliminated, in later versions of the network.

Hopfield Network

The Hopfield network [26, 28] was one of the first and simplest neural networks other than the perceptron. Hopfield networks can be classified along two major axes. The first axis distinguishes between networks that have units, or 'neurons', that compute a threshold function [26] and those that use a sigmoid nonlinearity [28]. The second axis distinguishes between units whose weights are learned and those whose weights are designed in advance.

The Hopfield network is a single layer network where each unit is (potentially) connected to every other unit. In the initial Hopfield model, connection strengths were symmetric, and direct feedback from a unit to itself was not allowed. Both of these conditions have been relaxed in subsequent work [11]. The basic topology of a Hopfield network is shown below in figure 5. This is a 4-unit network. The units are the circles at the bottom. The outputs of the net are the four lines labeled $O_1..O_4$. The outputs are fed back to the inputs of all the other units through the connection matrix. The connections are the small black boxes. A connection from unit i to unit j is denoted by T_{ij}. The connections are conductances, and negative values are allowed. For electronic implementation this would be accomplished by providing inverting and non-inverting outputs from the units

and connecting the appropriate one. The external inputs, I, to the network also appear at the top of the figure.

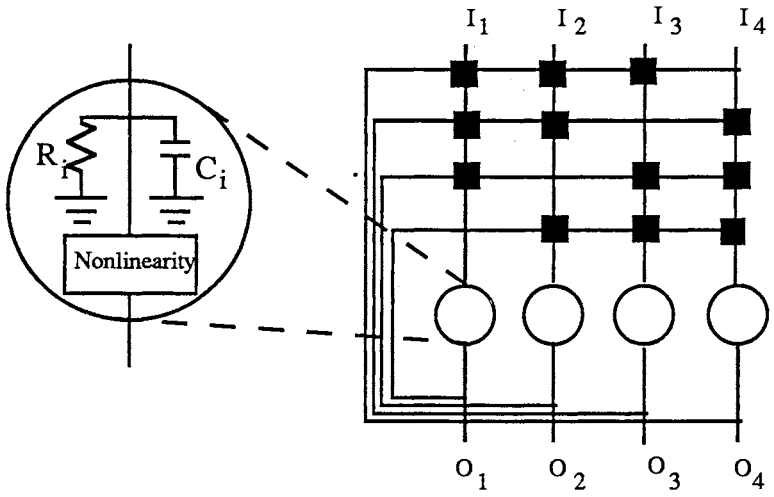

Figure 5: 4-Unit Hopfield Network

Another addition to the model is the introduction of dynamics to model the capacitive delays of real neurons. Each unit in the network has its input grounded by a parallel RC network. We can now model the instantaneous output voltage of unit i, v_i, in terms of its instantaneous input voltage, u_i, and its RC loading. u_i is the sum of the products of the neighboring units' output voltages, v_j, and the strength of the interconnections, T_{ij}. Due to the RC time constant, u_i will lag behind the instantaneous values of the neighboring v_j's. The equations describing the behavior of unit i are:

$$v_i = g(u_i) \tag{25}$$

$$g(u_i) = \frac{1}{1 + e^{-2\lambda u_i}} \tag{26}$$

$$C_i \frac{du_i}{dt} = \sum_j T_{ij} V_j - \frac{u_i}{R_i} + I_i \quad , \tag{27}$$

Hopfield [28] showed that for certain conditions on the interconnection network, the update rule for the units, equation (27), will cause the entire ensemble to seek the nearest minimum in the energy landscape. Because of this property, it is possible to use Hopfield networks to solve minimization problems. A procedure for designing networks of this type is developed in [60].

Koch Network

Several people have followed the procedure outlined in [60] to develop Hopfield networks for problems of interest to them. Koch, Marroquin, and Yuille [35] looked at using it for data smoothing. Least squares data smoothing can be cast as a minimization problem with the function to be minimized (in one dimension):

$$E(\underline{f}) = \sum_i (f_{i+1} - f_i)^2 + C_d \sum_i (f_i - d_i)^2 \tag{28}$$

where d_i is the input data, f_i is the smoothed output, and C_d adjusts the conflicting requirements of smoothness vs. fidelity to the data. Hopfield noted that this function can be minimized by an analog network of N nodes, each connected to its two neighboring nodes. Each unit is grounded by a parallel RC network. The input data is provided by a current into each node, the output is the voltage at each node. Such a network can be simulated by having N elements, each of which is updated by the rule

$$\frac{df_i}{dt} = -\frac{\delta E}{\delta f_i} = -\frac{1}{C_i}\left[\sum_j f_j/R_{ij} - \frac{f_i}{R_i} + I_i \right] , \tag{29}$$

where R_{ij} is the resistance between nodes i and j, R_i and C_i are resistance and capacitance to ground, I_i is the input current to each node, and f_i is the output function (voltage) at each node.

The problem with this model, and least-squares smoothing in general, is that it blurs discontinuities. Since discontinuities in images are usually very informative, this is quite a problem. Inspired by the seminal paper of the Geman brothers [22], Koch and his

colleagues extended this model to preserve large discontinuities by introducing 'breakpoint' terms, across which no smoothing is performed. Their objective function (in one dimension) is:

$$E(\underline{f}) = \sum_i (1-h_i)(f_{i+1} - f_i)^2 + C_d \sum_i (f_i - d_i)^2 + C_c \sum_i h_i \qquad (30)$$

where h_i is a 0..1 variable indicating the presence of a break between data points i and i+1. C_c is the cost of inserting a breakpoint. The h_i can be considered as learned weights whose update rule is:

$$\frac{dh_i}{dt} = -\frac{\delta E}{\delta h_i} = (f_{i+1} - f_i)^2 - C_c \qquad (31)$$

Koch also provides a formulation of this network to smooth 2D datasets [35]. The interpolation term has both horizontal and vertical breakpoints. He also extends the simple cost for inserting a breakpoint to an expression involving neighboring breakpoints in order to encourage straight lines and penalize adjacent, parallel lines [35].

Results from one and two-dimensional Koch networks are shown below in figures 6 and 7. Figure 6 shows the least-squares solution for two different values of smoothing. The input data is a pulse corrupted with 20% uniformly-distributed noise. The D1:S5 and D5:S1 entries in the legend indicate the relative weights of the data (D) and smoothing (S) terms of the equation. Figure 7 is for a 2D Koch network implemented at the beginning of this study. The top two illustrations show the input and output for a two-dimensional pulse, while the bottom two are for a truncated ramp. As can be seen in the figures below, the data smoothing network performs well. It smooths small errors while preserving large discontinuities. In addition to smoothing the output data, edge maps can be obtained by examining the breakpoints. These are not shown, but accurately identify edges in the image.

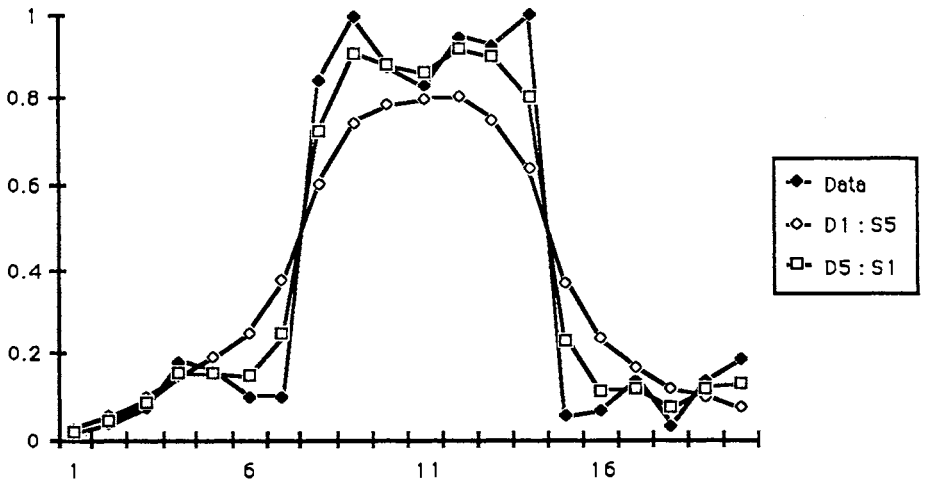

Figure 6: 1D Least-Squares Network Results

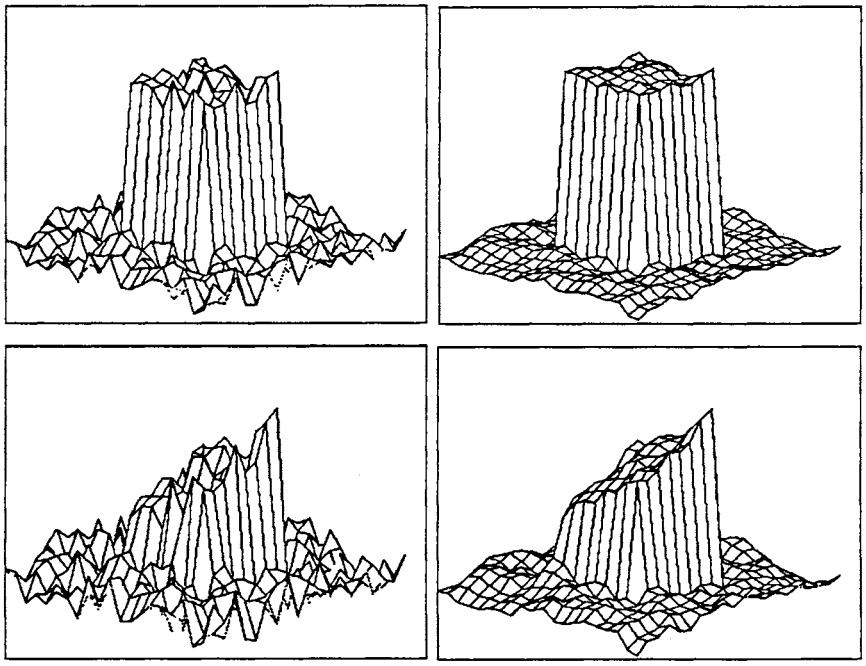

Figure 7: Results from 2D Koch Network

While the Koch network performs well, it is certainly no miracle algorithm. It should be noted that Figure 7 depicts the best results I was able to obtain from the 2D Koch network. Midway through this course of research I developed parallel implementations of the 2D Koch network and the median filter in order to compare their performance on an image smoothing task. That comparison, which is believed unique, is detailed elsewhere [15]. In brief, its findings were that the median filter was faster, usually gave better noise reduction while preserving image edges, and was easier to use since it did not suffer from the need to determine the convergence criteria and other free parameters in the Koch network.

CHAPTER IV

INITIAL SUPERQUADRIC NETWORK

Three problems with Solina's SQ recovery algorithm were listed earlier in this thesis: the slowness of the serial minimization algorithm, difficulty in recovering parameters from scenes with multiple SQs, and recovery time serialized in the number of estimated SQs. We saw that the speed of the method could be addressed by a parallel implementation. The other two problems arose from his method for handling scenes with multiple objects. Despite these problems, the data selection aspect of the multiple object recovery method offers potential benefits for segmentation. These benefits seem great enough that it would be worthwhile to try and find a better method for implementing the data selection. We also indicated that these problems might be ameliorated if we were to take advantage of image coherence.

Koch's network, which smooths data at the same time as it discovers image edges, seems to provide a reasonable framework for trying to exploit image coherence. The goal of this project was to see if a modified Koch network might be able to overcome the problems we noted in Solina's recovery procedure. The next section informally describes the organization of such a modified Koch network, and discusses how it might be able to overcome the problems noted above. For initial investigations the network was simplified by modeling 2D superellipses rather than the full 3D superellipsoids that are ultimately of interest. The subsequent section details the objective function used for the simpler network and discusses the design decisions that led to the network's organization. The following two sections discuss the initial simulations of this network and a hybrid minimization technique that was developed to overcome the problems found during the initial

simulations. The final section in this chapter presents the results of several experiments that were run to investigate the capabilities of the hybrid approach.

General Organization of the SQ Network

Recall that the Koch network was used for smoothing data that could be regarded as a surface. At each data site, the Koch network has a single unit to encode the height of the surface at that site. Between the sites there are breakpoint units to indicate discontinuities in the surface. How can this scheme be changed to model SQs? First, we will want to retain the breakpoint units between each site, as they could be used to locate the borders of the SQs. Second, the single unit at each site which encoded the estimated surface value will be replaced by a vector containing the estimates of the size (a_1, a_2, a_3), position (x_0, y_0, z_0), and shape (e_1, e_2) parameters for the SQ shape underlying the particular data point. Third, the model term of the objective function will need to be a function of the data values in a small neighborhood centered around the site of the parameter vector. This is because an infinite number of parameter vectors could fit a single data point, in contrast to the Koch network's simple $(f_i - d_i)^2$ model. The breakpoint units can play two roles in the parameter estimation. Like the Koch network, adjacent sites without an intervening breakpoint should have similar parameter values while adjacent sites with one should not. Additionally, the breakpoints will be used to determine if data points in a local neighborhood should be excluded from the minimization. This vector extension of Koch's network is one of the unique results of this thesis.

This network organization is illustrated in figure 8. The large square at the bottom represents the neighborhood from which the data points are drawn. In this figure, the neighborhood is 7x7. The central column represents the parameter vector associated with the data point at the center of the neighborhood. The short, thick, black lines between the data points represent the horizontal and vertical breakpoints.

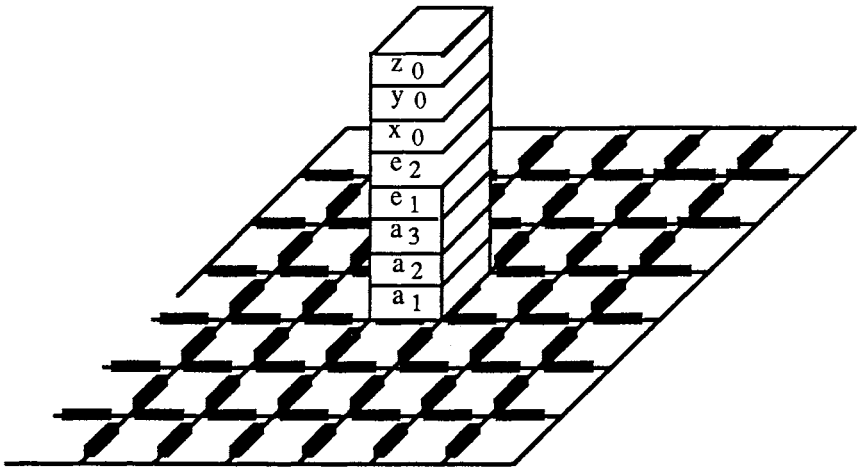

Figure 8: Local View of the Organization of the SQ Network

How might this organization overcome the three disadvantages noted for Solina's approach? By fitting the parameters to local neighborhoods, and by using the breakpoint terms to exclude points from the neighborhood, the network exploits image coherence in a very direct and natural fashion. As shown in figure 8, a parameter vector is associated with each pixel and only needs data from a small area of the image that surrounds the associated pixel. Given a computer with enough processors, all of the minimizations could be carried out in parallel. This would allow us to model scenes with large numbers of shapes, which was a problem with Solina's method.

The organization described above is not the only way that things could be arranged. A problem with the arrangement described above is that associating a parameter vector with each pixel yields an enormous number of parameters to estimate. Another problem is that the computational complexity of this approach is greater than that of Solina's method. Each parameter vector is estimated from a local neighborhood of data. If there are M active data points, the SQ network would have to estimate M parameter vectors. If each of these would be determined from an N element local neighborhood, we must evaluate the fit and partial derivatives for the SQ term MN times for each step. Solina's technique would only evaluate

the fit and partials M times, since it tries to fit all the available data to a single parameter vector.

These problems could be reduced by having one parameter vector for relatively small image regions, but what would happen when a SQ boundary bisected such a region? Handling this case would require the regions associated with each parameter vector to overlap, and it was decided that the job of determining what each breakpoint meant to each parameter vector region was just too complex, especially for an initial investigation into the feasibility of the technique.

SQ Network Objective Function

Once the general structure of the SQ network was decided upon, an objective function had to be found to make these ideas explicit. The objective function needed to incorporate the SQ fitting term. It also needed to ensure that neighboring parameter vectors without intervening breakpoints would converge to similar values, and finally it needed to promote the formation of good object boundaries without excessive breakpoint terms. To simplify initial investigations of the objective function, I decided to look at a 1-D network (SQNET1) that would estimate the parameters for 2D superellipses, rather than the 2D organization for estimating 3D superellipsoids that was shown above in figure 8. The objective function of SQNET1 is:

$$
\begin{aligned}
E = C_I & \sum_{i=\rho}^{N-\rho-1} (\underline{\lambda}_{i+1} - \underline{\lambda}_i)^2 \, (1-h_i) \\
& + \sum_{i=\rho}^{N-\rho} \sum_{j=i-\rho}^{i+\rho} (1-F(\underline{\lambda}_i, \underline{x}_j))^2 \prod_{k=\lfloor i,j \rfloor}^{\lceil i-1,j-1 \rceil} (1-h_k) \\
& + C_c \sum_i h_i + C_p \sum_i h_i \, h_{i+1}
\end{aligned}
\tag{32}
$$

where $\underline{\lambda}_i$ is the vector of parameter estimates at pixel i, h_i is the breakpoint between sites i and i+1, $\underline{x}_j$ are the x, y coordinates of the part of the object imaged on pixel j, and $F(\underline{\lambda}_i, \underline{x}_j)$

is a slightly modified version (see eq. 34) of the SQ inside/outside function which measures the fit of the data from pixel j to the parameters at pixel i. C_I weights the interpolation term relative to the unit weight of the model term. The C_C term penalizes each discontinuity inserted, while the C_P term penalizes adjacent discontinuities. Equation (32) extends Koch's objective function (30) to the vector domain. It also incorporates the SQ model and utilizes data coherence. This objective function is one of the unique results of this research.

How does it solve the problem of modeling SQs? The first term in (32) penalizes adjacent parameter vectors that have different values in the absence of an intervening discontinuity. The second term implements the SQ fitting, while the third term introduces a constant cost for every breakpoint inserted. The fourth term penalizes the formation of adjacent breakpoints. Since we have more parameters to estimate than in the data smoothing example, we must look at more data. The index i selects the parameter vector under consideration. The index j selects the data points from sites $i-\rho$ to $i+\rho$ that will be used in the minimization of parameter vector i. Finally, the index k selects the discontinuities between sites i and j. The product term removes data points from consideration in the parameter estimation if a discontinuity lies between the parameter vector at pixel i and the data point at pixel j. The $\lfloor\ \rfloor$ and $\lceil\ \rceil$ operators in the summation indices of the product term indicate the minimum and maximum operations, respectively. Edge conditions are handled by not running the minimization closer to the boundary than the size of 1/2 the neighborhood.

The layout of SQNET1 is shown below in figure 9. The figure is composed of three main blocks. The top one is the input data, N x-y pairs. The middle block is the breakpoints. The bottom block is the parameter vector associated with each data point. Note that the blocks are not the same size. Since the breakpoints indicate a discontinuity between adjacent data points, one fewer breakpoint is needed than the number of data points. The variable ρ is the neighborhood size parameter that is used in the indices of summation of

the cost function. Since we do not want the parameter vectors accessing nonexistent data, the edge conditions are handled by not having any parameter vectors associated with the first and last ρ data points.

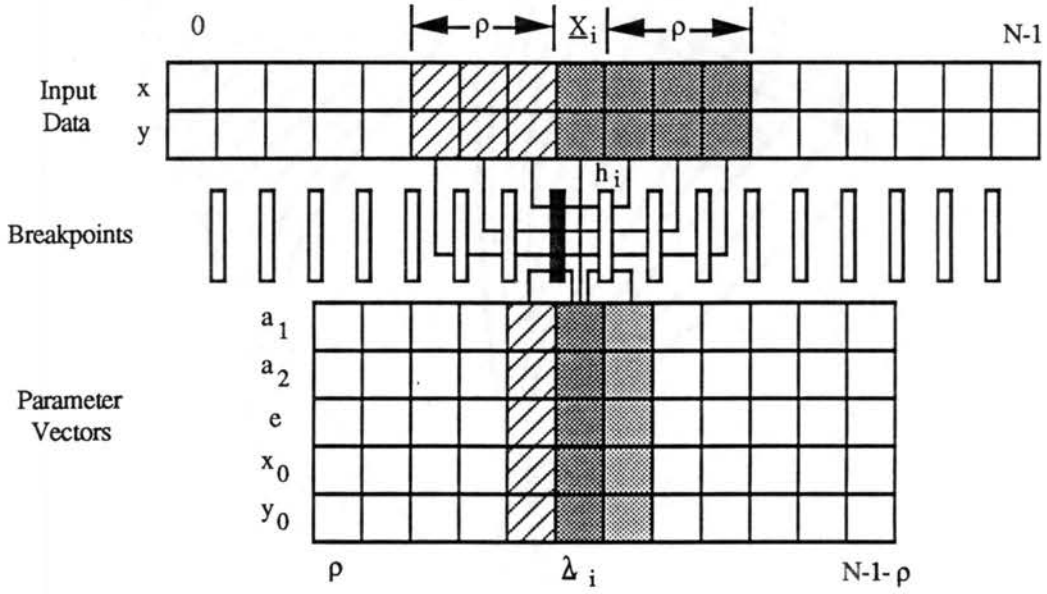

Figure 9: SQNET1 Organization

Now consider the highlighted parameter vector, $\underline{\lambda}_i$. The parameters in this vector are estimated from the data points in a 2ρ+1 local neighborhood. The data neighborhood is also highlighted, and lines are drawn from $\underline{\lambda}_i$ to the data points. Note that these lines go through the breakpoints. If a breakpoint develops somewhere in the neighborhood, all points further away from the parameter vector than that breakpoint should not be considered. This is shown in the figure by the black breakpoint, and by the crosshatched highlighting on the data points that are not used to estimate the parameters for $\underline{\lambda}_i$.

$\underline{\lambda}_i$ also depends on $\underline{\lambda}_{i+1}$ and $\underline{\lambda}_{i-1}$, because of the interpolation term in the cost function which tries to force adjacent parameter vectors to have similar values in the absence of an intervening breakpoint. The dependence of $\underline{\lambda}_i$ on $\underline{\lambda}_{i+1}$ and $\underline{\lambda}_{i-1}$ is shown by highlighting, and by the lines connecting the parameter vectors through the breakpoints. The vector that has the intervening discontinuity is give the same shading as the discarded data points, while the one that is used has shading similar to $\underline{\lambda}_i$, although a lighter shade.

The elements of the parameter vector are a_1 and a_2, the size of the SQ in the x and y directions, e_1, the SQ shape parameter, and x_0 and y_0, the position of the center of the SQ in the x and y directions. Although the breakpoint terms, h_i, are also estimated parameters, they are not considered as part of the parameter vector $\underline{\lambda}_i$. This is because their interactions are more complex than the simple smoothness requirement $(\underline{\lambda}_{i+1} - \underline{\lambda}_i)^2$.

Every data point in the image has a parameter vector associated with it. The parameters describe the object on which the data point lies. The interpolation term of (32) should force neighboring parameter vectors, which describe the same object, to similar values. Once the minimization is complete, a postprocessing step would label regions of similar parameter estimates as a single object.

Solina's modified inside/outside function, (4), was first presented in [58]. The only modification to the standard SQ inside/outside function is the outermost exponent e_1. This modification is necessary to allow accurate recovery of cylindrical objects, as was explained in Chapter II. Equation (33) is a very slightly rewritten form of (4), where $\underline{\lambda}$ is the parameter vector, while $\underline{x}$ is the xyz triplet giving the coordinates of a point on an object.

$$F(\underline{\lambda}, \underline{x}) = \left(\left(\left(\frac{x-x_0}{a_1} \right)^{\frac{2}{e_2}} + \left(\frac{y-y_0}{a_2} \right)^{\frac{2}{e_2}} \right)^{\frac{e_2}{e_1}} + \left(\frac{z-z_0}{a3} \right)^{\frac{2}{e_1}} \right)^{e_1} \qquad (33)$$

Equation (33) would form the basis for a 2D network to estimate the parameters for 3D SQs. The function used for the 1D network (34) is simpler, since it has no z term and only a single shape parameter, e_1.

$$F(\underline{\lambda}, \underline{x}) = \left(\left(\frac{x-x_0}{a_1} \right)^{\frac{2}{e_1}} + \left(\frac{y-y_0}{a_2} \right)^{\frac{2}{e_1}} \right)^{e_1} \tag{34}$$

SQNET1 Simulations and Results

To test the feasibility of a network-based minimization of the SQ function, I began looking at the 1D network just described. This network simulation, using the state-variable approach for the h_i, was written in C. MACSYMA was used to compute the partial derivatives of the objective function with respect to the elements in the parameter vector. The update rule applied at each unit was the negative of this partial derivative, *i.e.*, a simple crawl down the gradient.

The objective function minimized rapidly. However, a significant residual error remained, and the parameter estimates never varied significantly from their initial values. Many values for the adjustable constants (C_b, C_d, λ, etc.) were tried with no appreciable improvement in performance. The explanation found for this behavior was that the discontinuity terms rapidly assumed their correct values. These terms pervade the objective function, explaining the rapid decrease in E. The interpolation and the $(1-F(\underline{\lambda}_i, \underline{x}_j))^2$ terms had very limited effect relative to that of the discontinuities.

To discover the reason for this problem, the discontinuities were fixed at their correct values and the minimization restarted. Again, the objective function decreased, although not as rapidly. However, the residual error and the lack of significant change in the parameters remained. This time, the interpolation term was enforcing similarity between neighboring sets of parameters, but the $(1-F(\underline{\lambda}_i, \underline{x}_j))^2$ term was still not having a significant effect on the objective function.

Further investigation revealed the source of the problem. Figure 10 shows the value of E vs. various values of y_0 (the y position parameter) and a_2 (the y scale parameter). This long shallow valley is the classic case where gradient descent fails. Since the Koch network implements gradient descent (using a fixed step size rather than the more common line search) it is not surprising that it fails to make significant progress in this case. This valley arises because of the negative correlation between the y_0 and a_2 parameters that occurs because only the top half of the ellipse can be sampled. This is illustrated below in figure 11. The figure contains three ellipses, each with the same x-size and x-position. The y-size and y-position differ for each. The surface of the mid-sized ellipse is periodically sampled to give y values vs. x. These tuples are the input data for the network. Notice that the other two ellipses fit the data equally well, and almost as well as the true data.

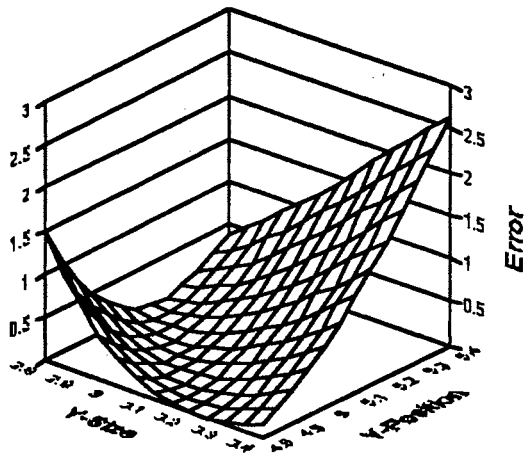

Figure 10: E vs. a_2 (y-size) and y_0 (y-position)

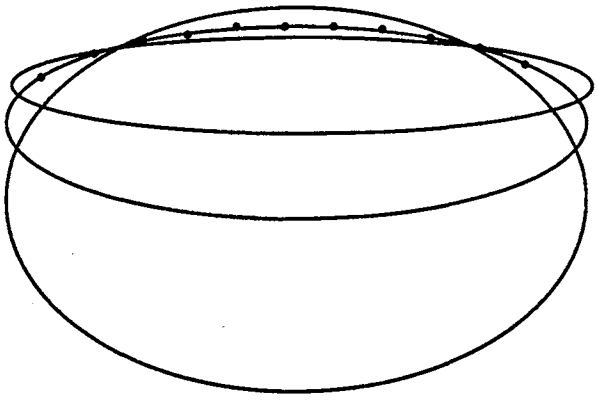

Figure 11: Cause of Correlation between y_0 and a_2

Hybrid Algorithm

Recall that the goal of this project was to see if a minimization based on the Koch network could overcome some of the problems with Solina's SQ parameter estimation technique. However, we have just seen that the Koch network's simple gradient descent is not capable of minimizing the most important part of the objective function. Nevertheless, the network organization and the objective function (32) were intuitively appealing to me. They seemed to express a natural way of exploiting image coherence to find regional estimates of SQ parameters while simultaneously discovering their borders. If successful, this could significantly ease the "chicken and egg" nature of the segmentation problem that has plagued machine vision for so long. Therefore, to continue investigating the capabilities of the network, other minimization techniques were tried. A line search version of gradient descent, a conjugate-gradient method, and the Levenberg-Marquardt (LM) algorithm were tried [57, 52]. The first two were not successful, but the LM method did succeed in minimizing the $(1\text{-}F(\underline{\lambda}_i, \underline{x}_j))^2$ term.

Having found an algorithm that could minimize the most difficult portion, it was now time to rebuild to the complete version of the objective function. However, practical

considerations prevent us from simply using the LM algorithm to minimize the entire objective function. The LM algorithm achieves its speed and performance by approximating the Hessian matrix, which is NxN where N is the number of parameters to be estimated. Recall that normal images will have on the order of 250,000 pixels, and that each of those pixels would have approximately 10 parameters to be estimated. This means that the Hessian matrix would require 100 Gigabytes of working storage!

Because of this problem, and because of a desire to remain as close as possible to the spirit of the Koch network, a hybrid minimization algorithm was developed to minimize (32). First, the minimization of the interpolation term was restored. The gradient descent network was retained for this, while the LM method was used to minimize the $(1-F(\underline{\lambda}_i, \underline{x}_j))^2$ term. The discontinuity terms were initially set to their correct values and not allowed to change. This minimization was successful.

The breakpoint terms were then restored to the minimization. The gradient descent algorithm was also used for their update. This algorithm was not successful. The behavior noted was that the overall cost would decrease for a time, then begin to increase. To see why this behavior was observed, recall that the breakpoint terms are a non-linear function of an underlying state variable. The particular non-linearity used was the sigmoid (13), which is only asymptotic to 0 and 1. Therefore, the breakpoint terms do not provide perfect isolation. Also recall that if the data are perfectly modeled by the SQ term, the cost of that term goes to zero. Now consider a parameter vector and its associated data neighborhood. Assume there is a breakpoint set that removes some of the data points from consideration. Given the two facts mentioned above, it is easy to see that when the parameters at the site are adjusted to correctly account for the data in the neighborhood, the data outside the breakpoint begins to dominate the cost function. At this time the parameters begin to diverge from their best values in order to account for the data they should not be considering.

The only solution that I can see for this problem is to make sure that the discontinuity terms adopt binary values. The sigmoid nonlinearity was replaced by a threshold function. The update rule for the underlying state variable remains the negative partial derivative of the cost w.r.t. the discontinuity. With this change, the minimizations succeeded. The hybrid minimization algorithm used for SQNET1 is presented below as Algorithm 2.

ALGORITHM 2

HYBRID MINIMIZATION ALGORITHM FOR SQNET1

```
declare λ and χ as N element arrays of floats
declare x and s as 2ρ+1 element arrays of floats
declare p as a 5 element array of floats
declare v as N element array of Booleans
declare h as an N-1 element array of Booleans
declare XX as an Nx2 element matrix of floats
declare PP as an Nx5 element matrix of floats
declare h_freq, lm_iters, iter_limit as integers
declare chi_tol as a float

Get data and initial estimates of parameters and discontinuities. Also get control values for
minimization procedure.
input XX, PP_t=0, h_t=0
input h_freq, lm_iters, iter_limit, chi_tol, mag_tol

iter = 0
v = 0                              Set all sites to "unconverged"
do {
    iter = iter + 1
    χ = 0                          Clear χ² at all sites
    all_converged = 1              Be optimistic

    For each non-border data point
    for i = ρ, N-ρ  {

        Skip sentinels and converged sites
        if XX_i is a sentinel or v_i = 1
            continue to next i

        Extract site i's parameters and local data neighborhood into
        working variables. Compute the s vector which temporarily
        marks local neighborhood points as sentinel points if they
```

are on the other side of a discontinuity.
$\underline{x} = XX_{i-\rho} \ldots XX_{i+\rho}$
$\underline{p} = PP_i$
$\underline{s}_j = \prod_{k=\lfloor j,i \rfloor}^{\lceil i,j \rceil} (1-h_k)$

Do a few iterations of the model-based minimization
for n = 0, lm_iters
lm_minimization_step ($\underline{x}$, $\underline{s}$, $\underline{p}$, $\underline{\lambda}_i$, χ_i)

Insert the new parameters for site i into the network
$PP_i = \underline{p}$
}

Do the interpolation, adding to the χ^2 vector
for i = ρ, N-ρ {
if XX_i is a sentinel or PP_i has converged
continue to next i
$PP_{i_{t+1}} = ss * ((1-h_{i-1})*PP_{i-1_t} - 2 * PP_{i_t} + (1-\underline{h}_i)*PP_{i+1_t})$
$\chi_i = C_I * ((1-h_{i-1})*PP_{i-1_t} - 2 * PP_{i_t} + (1-\underline{h}_i)*PP_{i+1_t})$
}

Update the discontinuities if it is time to do so, and add to the χ^2 vector
if iter MOD h_freq = 0 {
update_discontinuities(PP, $\underline{h}$)
for i = ρ, N-ρ−1
$\chi_i = C_C * \underline{h}_i + C_P * (\underline{h}_i * \underline{h}_{i+1})$
}

Test for convergence
for i = ρ, N-ρ {
if $\underline{v}_i = 1$
continue to next i

if χ_i < chi_tol
$\underline{v}_i = 1$
else
all_converged = $\underline{v}_i = 0$

} until (all_converged OR iter > ilimit)

Preprocessing and Postprocessing

Since the recovery procedure is an iterative algorithm, its behavior is dependent upon the initial parameter estimates. Therefore, before we can discuss the performance of the network we must first discuss the procedure for obtaining the initial parameter estimates. This section first discusses the form of the input data, then discusses the preprocessing performed to obtain the initial parameter estimates. The region-growing procedure used as a post-processing step is discussed next, followed by the performance measure that is be used to evaluate the network's performance.

The input data to the network is a vector of xy pairs. The x values are uniformly spaced on an interval, typically 20 data points from 0.0 to 10.0. The y values can either be the y-coordinate of a point on the top half of a SQ ellipse (see figure 11) or a sentinel value indicating a background pixel. This assumes that a figure-ground segmentation has already been performed.

A finite-state machine was used to implement the 1D preprocessing. The input is scanned in left-to-right order, looking for transitions between sentinel and non-sentinel values, or large discontinuities in the non-sentinel values. These transitions are assumed to mark the edges of the superquadric shapes in the input data. The initial discontinuity estimates are set to 1 at the transitions, and 0 elsewhere. Between the transitions, the minimum and maximum values of x and y are recorded. This essentially sets up a bounding box for the top half of the underlying superquadric, as shown by the bold boxes in figure 12. The y-position is set to the minimum y value, the y-size is set to $(y_{max} - y_{min})$. The x-position is set to $(x_{max} + x_{min}) / 2$, and the x-size is set to $(x_{max} - x_{min}) / 2$. These are shown by the crosses in figure 12. The shape parameter, e_1, is always set to 1.0. This yields an initial estimate of the shape as an ellipsoid that fits within the larger boxes shown in the figure. The accuracy of the estimate depends upon how closely the SQ is sampled to its extrema in the x direction, and on the true value of its e_1 shape parameter.

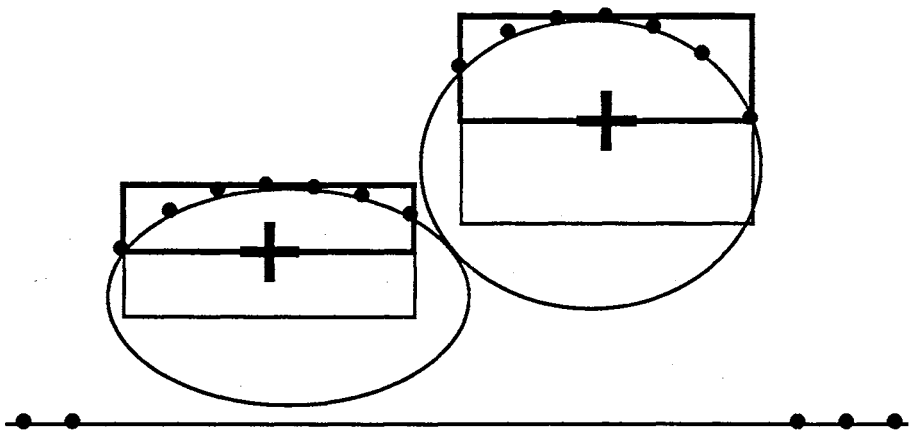

Figure 12: Initial Parameter Estimation

Some postprocessing is performed on the output of SQNET1. Because the network outputs discontinuities estimating the borders of the SQs, the region growing algorithm used for post-processing the parameter estimated was very simple. The vector of discontinuities was scanned from left-to-right, looking for runs of non-discontinuities. The parameter estimates within these runs were collected and the median of each component computed. The median value was found to be a more accurate estimator than the mean, so that is what is reported as the vector of parameter estimates for a particular region. The output of the procedure is a parameter vector for each region bounded by discontinuities.

Finally, we need to define a performance measure. Some of the experiments we will be running with SQNET1 will be varying the adjustable constants C_I, C_C, etc. We cannot use the χ^2 values computed inside the minimization technique to compare the effects of changing these constants, since it would be possible for identical parameter estimates to have different χ^2 values depending on the values for the weights. The performance measure used in the following experiments is denoted SSE_N. It is defined as:

$$SSE_N = \frac{1}{N}\sum_{i=1}^{R}\sum_{j=1}^{N_i}(1 - F(\underline{\lambda}_i, \underline{x}_j))^2 + |\, N_{est} - N_{in}\,| \qquad (35)$$

where N is the number of sites inside the labeled regions, i indexes the regions, R is the number of regions, j indexes the data points within each region, from 1 to N_i, F is the SQ fitting term from (34), N_{est} is the number of SQs estimated from the data, and N_{in} are the number of SQs that actually generated the input data. The first term is normalized to prevent penalizing larger regions. This form was chosen over the more obvious $(y - y(\underline{\lambda}, \underline{x}))^2$ because near the edge of recovered shapes there are frequently points with an imaginary discriminant.

Results

This section discusses the performance of SQNET1. We need to know many of the properties of the network in order to adequately evaluate its capabilities. For example, it is position invariant? Is the accuracy of the parameter estimates sensitive to the shape of the SQ being modeled? How densely do we need to sample a SQ in order to be reasonably certain of being able to model it? Is the method robust when the input data is noisy? Several experiments were designed and run in order to answer these, and other, questions.

Before we could run the experiments, we had to determine values for the constants that weight the various terms in the objective function and control the operation of the minimization procedure. There is no method of determining, *a-priori*, the best set of values for these constants. Therefore, a search procedure was written that would vary these over a range and record SSE_N for each set. The best performing values were identified, tighter bounds were set on the parameters, and the procedure repeated. Best performing in this case means most reliable over a range of noise values. The most reliable set of values found are given below in Table 1.

TABLE 1

BEST PERFORMING CONSTANTS
FOR SQNET1

Parameter	Value
C_I	2.0
C_C	10.0
C_P	1.0
ρ	3
ss	0.1
chi_tol	1.0
lm_iters	3
ilimit	3
h_freq	1

Unless stated otherwise, all the experiments were run with these figures. There are too many parameters to plot a meaningful figure showing the results of this process, but we can use surface plots to show the behavior as two parameters are varied and the rest held constant. Examples of the behavior are shown in figures 13 and 14. The first shows that if the interpolation term is not given enough weight, the cost of inserting a discontinuity becomes too expensive and a single SQ is fit to the data. The consequent poor fit is the high plateau at the back of figure 13. On the other hand, if C_I is too big, extra discontinuities will be inserted to ensure closer and closer fits to noise in the data. This is the relatively small rise at the front of the figure. Figure 14 shows that having too small a neighborhood, or running the minimization too long can lead to estimating too many SQs in an attempt to fit the noise. Figure 15 plots the input data and the recovered superquadrics for a run using the most reliable parameters listed above. As you can see, the fit is quite close. This is confirmed by looking at the input and estimated parameter vectors, which are given below in Table 2. All these figures illustrate another problem - the reader must be aware of the axes scales. The ellipses in figure 15 are actually circles. In surface plots, automatic scaling can make mountains out of molehills.

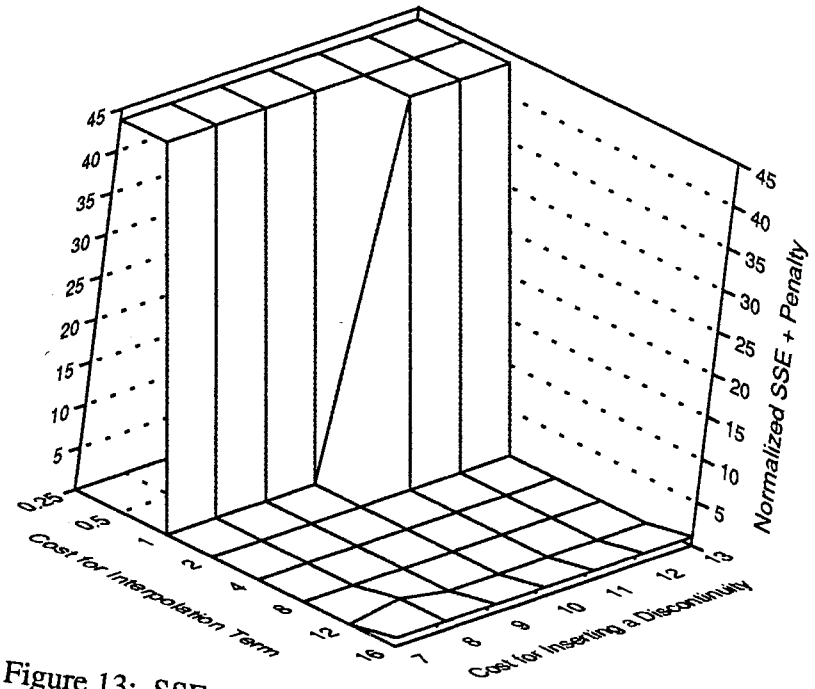

Figure 13: SSE_N vs. Interpolation & Discontinuity Costs

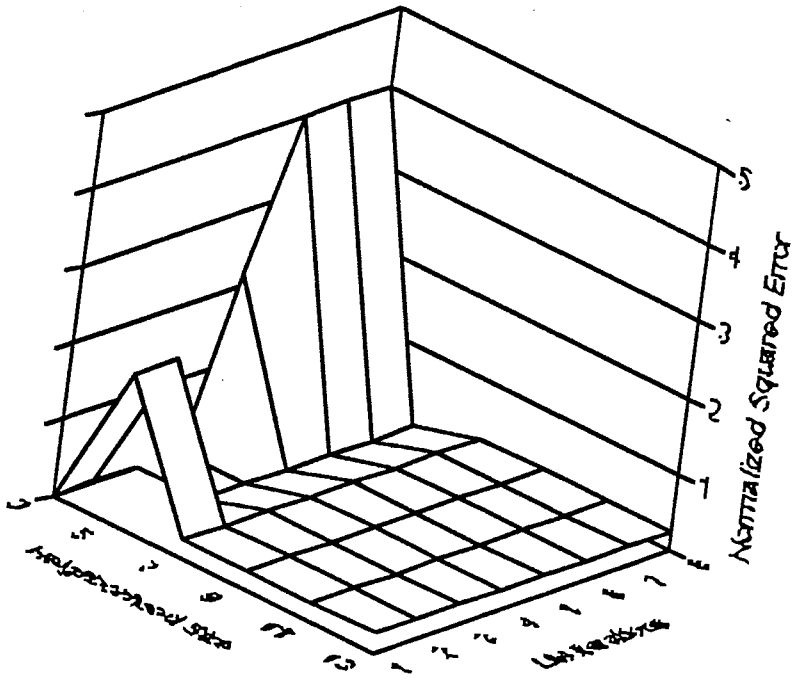

Figure 14: SSE_N vs. ρ & LM_iters

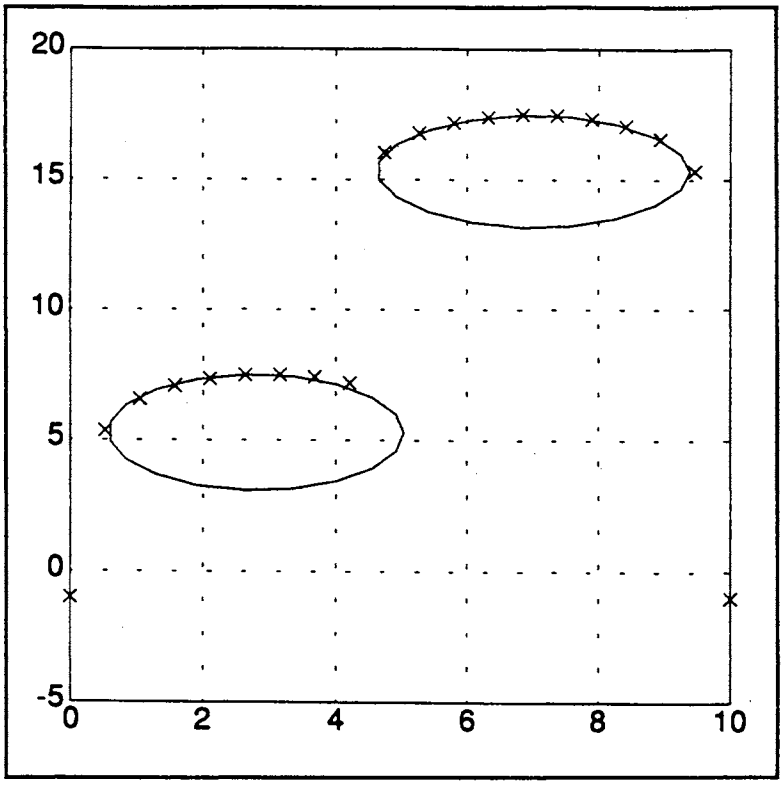

Figure 15: Input Data and Estimated Superquadrics

TABLE 2

INPUT PARAMETERS, ESTIMATED PARAMETERS, AND ESTIMATION ERROR FOR SQNET1

	a_1	a_2	x_0	y_0	e_1
Input SQ1	2.5	2.5	3.0	5.0	1.0
Estimated SQ1	2.227	2.204	2.806	5.288	1.008
Error SQ1	-0.273	-0.296	-0.194	0.288	0.008
Input SQ2	2.5	2.5	7.0	15.0	1.0
Estimated SQ2	2.388	2.167	7.006	15.330	1.032
Error SQ2	-0.112	-0.333	0.006	0.330	0.032

Experiment 1: Position Invariance

The first property of the network we would like to verify is its position invariance. Since the position of the SQ is explicitly modeled in the objective function (32) we would not expect the quality of the estimated parameters to depend upon the object's position. To verify this, a simple experiment was run. The dataset for this experiment was a single SQ whose x_0 and y_0 parameters were both varied 11 times over the range 4.0 to 5.0. As expected, the technique was insensitive to changes in y_0. However, it did show some slight variations in SSE_N when x_0 was varied. Figure 16 shows this variation for a fixed y_0. The changes in SSE_N are relatively minor, even the worst score has very reasonable parameter estimates, as can be seen in Figure 17. The differences in SSE_N are attributed to small numerical differences due to the object being sampled at slightly different places.

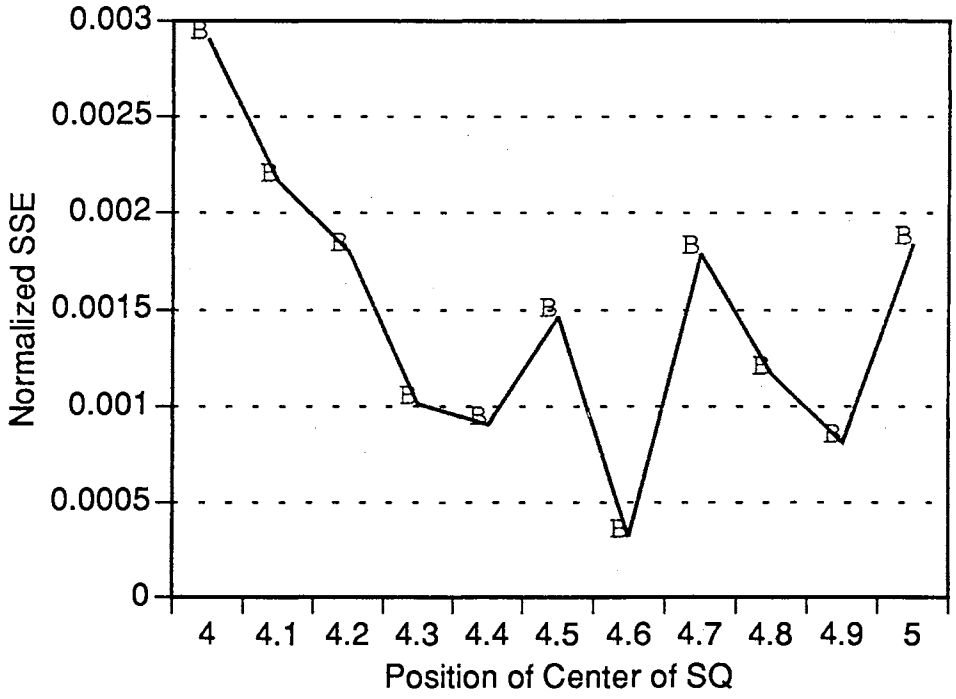

Figure 16: SSE_N vs. Position

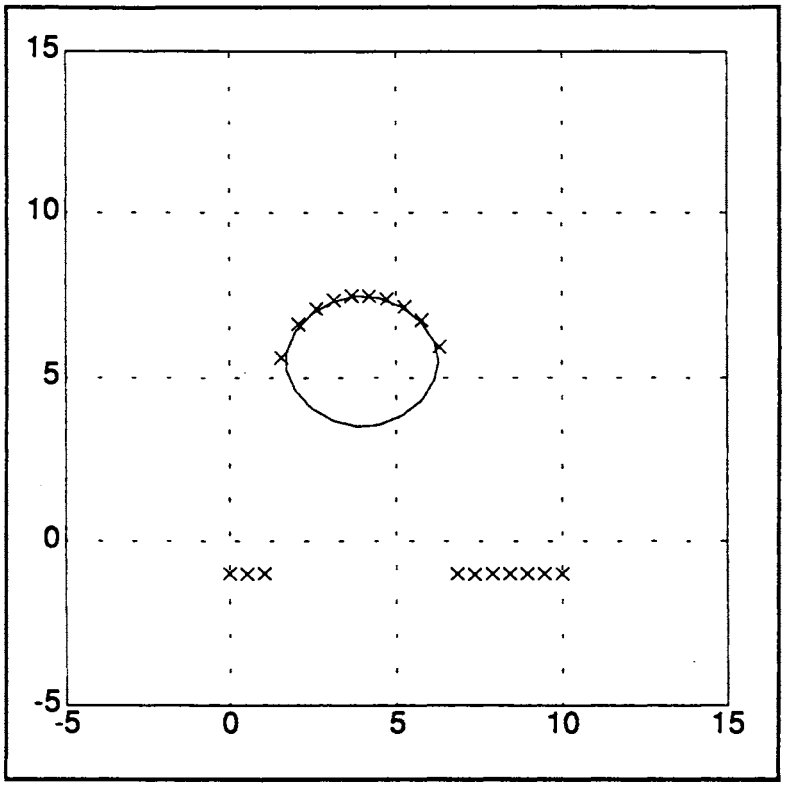

Figure 17:Data and Estimated SQs for $x_0 = 4.0$

Experiment 2: Size Invariance

Another important property that we wish to verify is size invariance. In other words, is the quality of the parameter estimates invariant to the size of the SQs we are modeling? This is equivalent to asking how many samples of the SQ are needed to accurately model it. The data set for this experiment has a single SQ, and the number of data points which sample the SQ are varied from 7 to 23. The size of the local neighborhood was also varied to see if this has an effect on the number of samples needed for accurate estimation. Figure 18 shows the effect these have on SSE_N. We see that the size of the neighborhood does not have a strong effect. The behavior with respect to the number of samples is more complex, although generally the more samples the better. The

graph below is somewhat deceptive because of the scaling. Even the worst-performing data point has made very reasonable estimates of the parameters. Given the small number of samples available, this seems like a very good result.

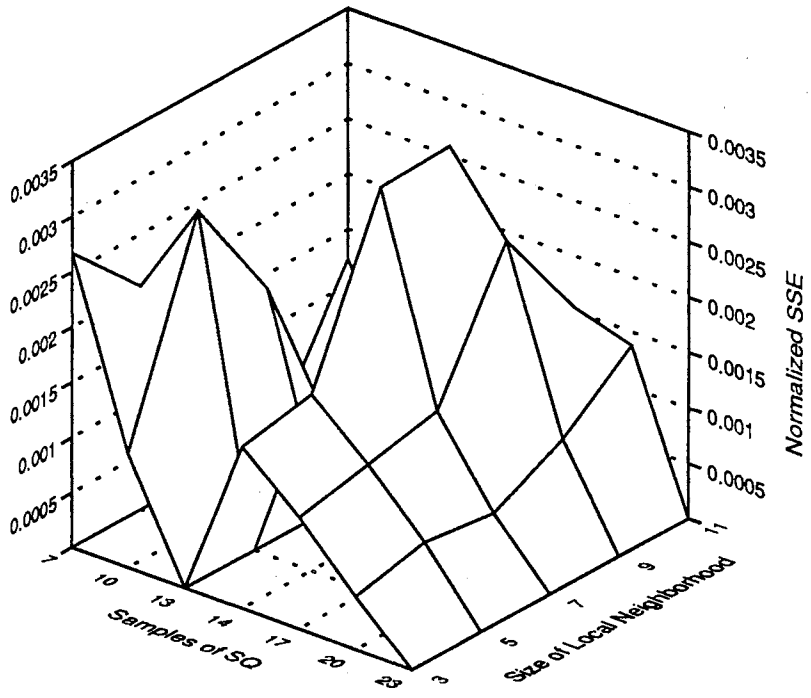

Figure 18: SSE_N vs. Sampling Density

Experiment 3: Shape Invariance - Aspect Ratio

A third important invariance property is shape invariance. In other words, is the quality of the parameter estimates is affected by the shape of the SQ we are trying to model? There are two ways we can change the shape of a SQ ellipse. The first, which is the subject of this experiment, is to change its aspect ratio by use of the a_1 and a_2 parameters. The second, which is the subject of the next experiment, is to change the SQ shape parameter e_1.

The data set for this experiment is a single SQ whose aspect ratio is changed by varying $a_2/a1$ from 1/4 to 4/1. This changes the shape of the SQ from a short, broad ellipse, through a sphere, to a tall, thin ellipse. In each run the SQ was sampled 10 times. Figure 19 shows that the technique is relatively insensitive to changes in aspect ratio, at least over the range tested.

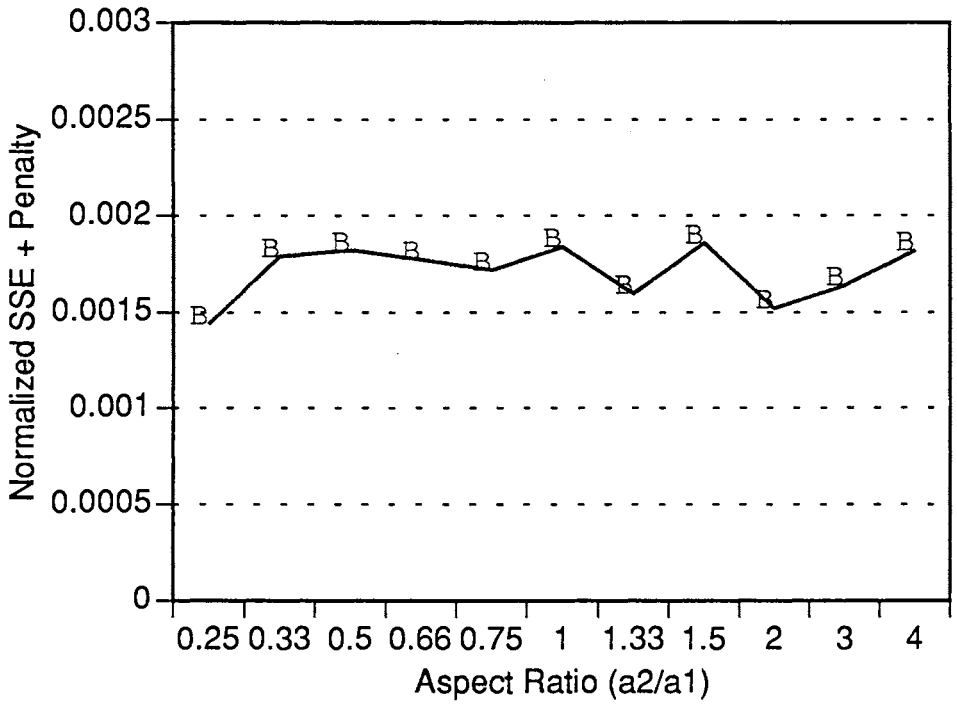

Figure 19: SSE_N vs. Aspect Ratio

Experiment 4: Shape Invariance - SQ Shape Parameter

The previous experiment mentioned that changing the aspect ratio and the shape parameter were the two ways of changing the shape of the SQ. This experiment was designed to see if SQNET1's performance is invariant to changes in the shape parameter.

Up to now, all the experiments have been run with SQs whose shape parameter was equal to 1.0, i.e., circles or ellipses.

The dataset for this experiment consisted of a single SQ whose e_1 parameter is varied over the range 0.2 to 2.0. This changes its shape from roughly square, through a sphere, to a diamond. Figure 20 shows the effect this has on SSE_N. Generally, the results are good until the shape becomes very close to a square. This result is expected, given the way we are sampling the SQ. Figure 21 shows the data and estimated SQ for the case where $e_1 = 0.2$. Note that we have essentially no information from the sides of the square, therefore we have no information about the size of the SQ or the location of its center. Given that lack of information, the estimated parameters explain the data about as well as can be expected.

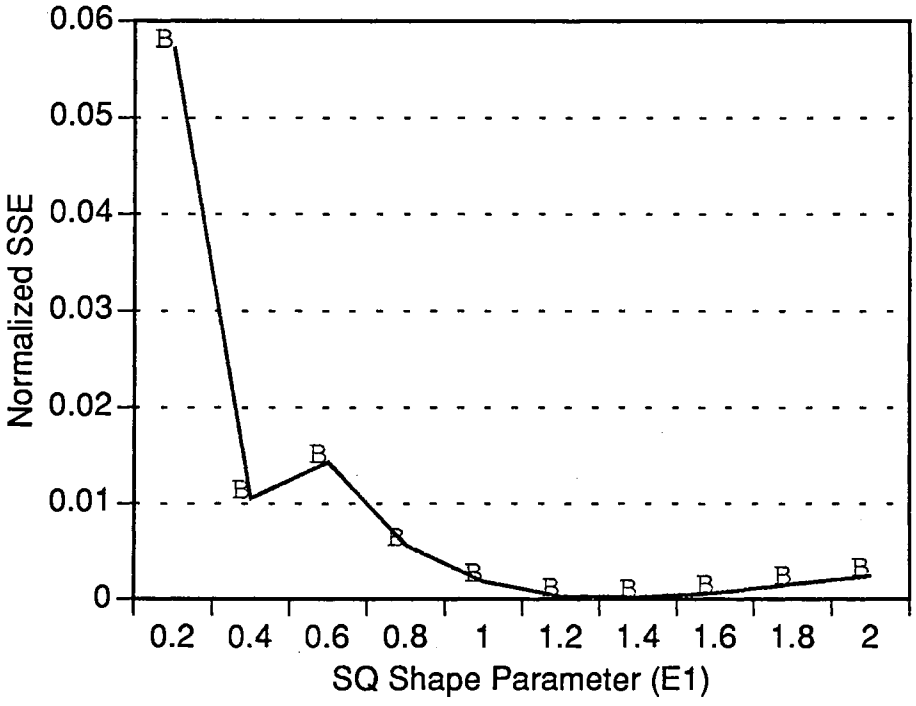

Figure 20: SSE_N vs. SQ Shape

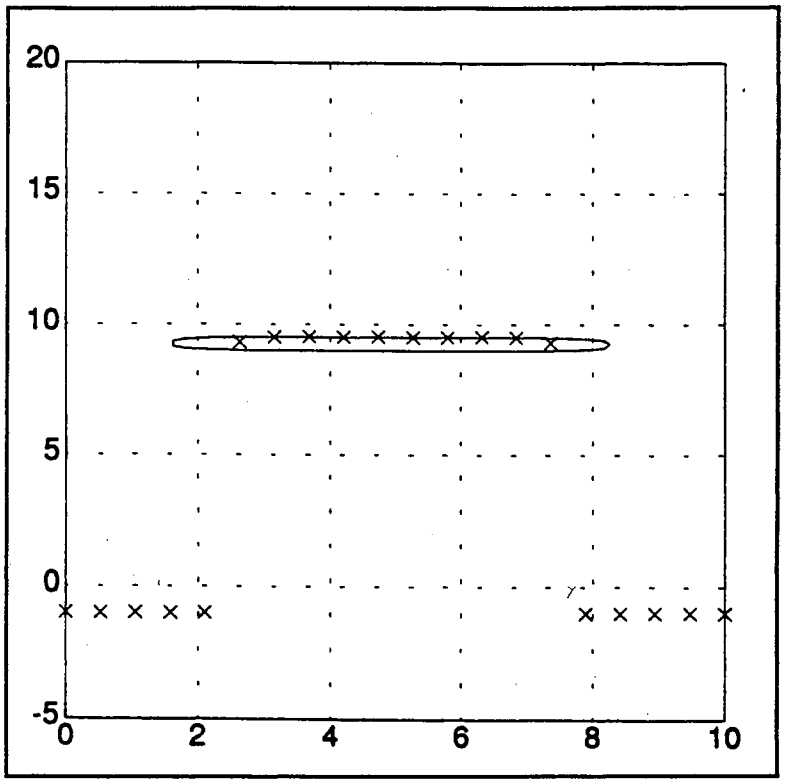

Figure 21: Data and Estimated SQ for $e_1 = 0.2$

Experiment 5: Sensitivity to Number of SQs

One of the motivations for the development of this technique was the inability of Solina's technique to handle data sets with more than a few superquadrics. To see if SQNET1 overcomes this problem, data sets with varying numbers of SQs were generated. Figure 22 shows the effect on SSE_N as the number of SQs in the data set is varied from 1 to 15. The SQs have identical size and shape parameters ($a_1 = a_2 = 2.5$, $e_1 = 1.0$). The position of each SQ is set so that the SQs usually, but not always, have a slight overlap in the x-direction. The y_0 parameter is increased by 5 for each SQ. The number of data points is increased in each set, so that each SQ is sampled approximately 10 times. Each time the correct number of SQs was estimated, and the variations in SSE_N are at a level attributable

to sampling differences. The worst performance was for 6 SQs, that case is displayed in figure 23.

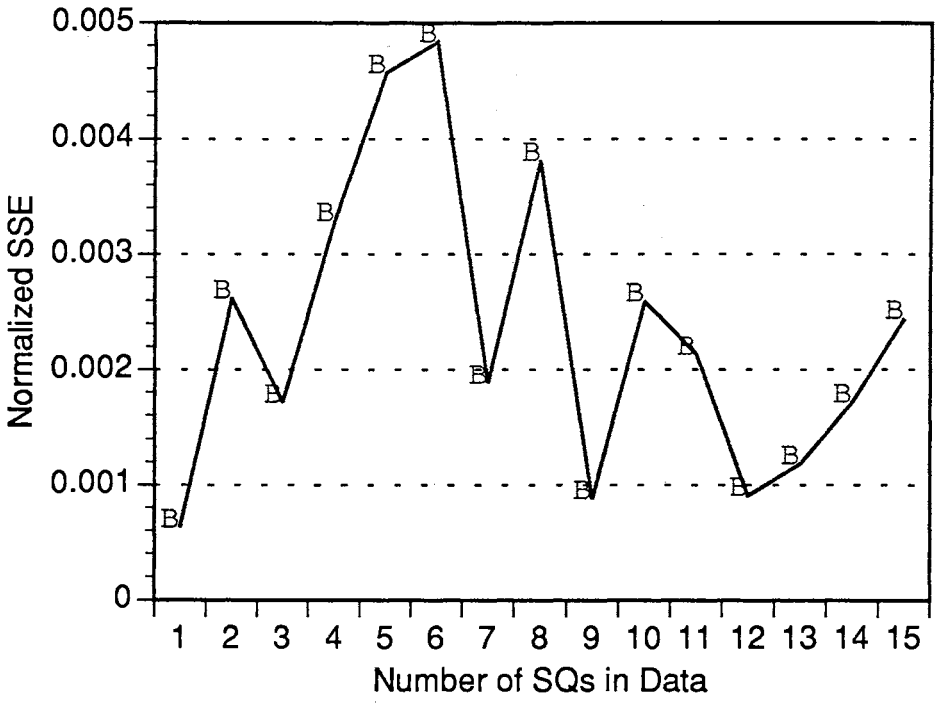

Figure 22: SSE_N vs. Number of Superquadrics in Data

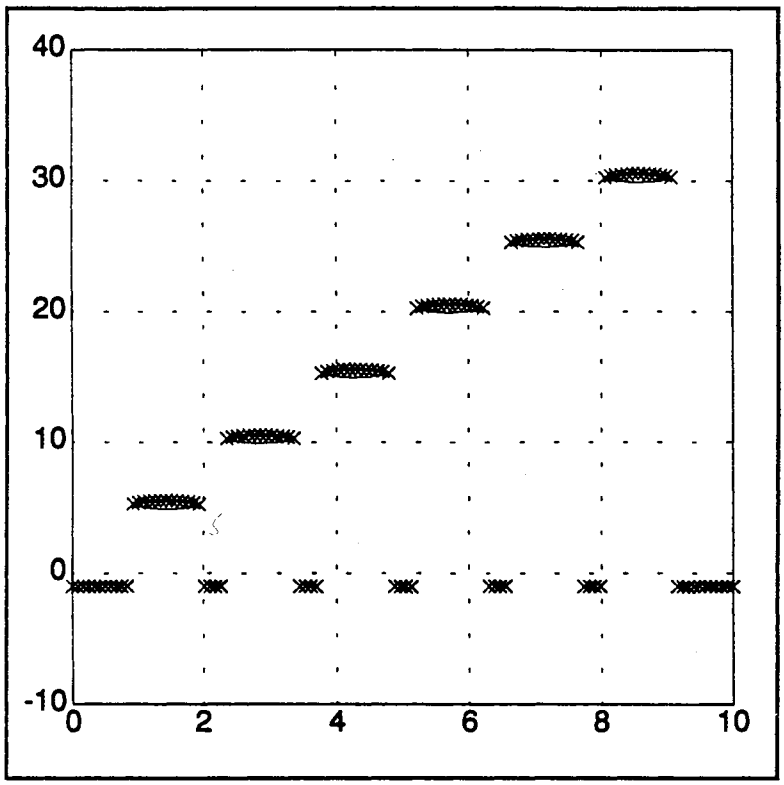

Figure 23: Data and Six Estimated SQs

Experiment 6: Noise Immunity

It is important to know how robust any machine vision algorithm is to noise in its input data. For SQNET1, increasing the neighborhood size as the data becomes noisier may offer some immunity. The purpose of this experiment was to test this hypothesis. The data set for this experiment was the same as that in figure 15, except that it was corrupted with different levels of uniform noise. Several runs were made at each noise level in order to measure the effect of different neighborhood sizes. Figure 24 shows the effect these had on SSE_N. We can see that performance falls off for moderately noisy data, but that a larger neighborhood size can help to overcome this. For very clean data the larger neighborhood

size hurts performance slightly. This effect is believed to be due to a poor choice of convergence criteria.

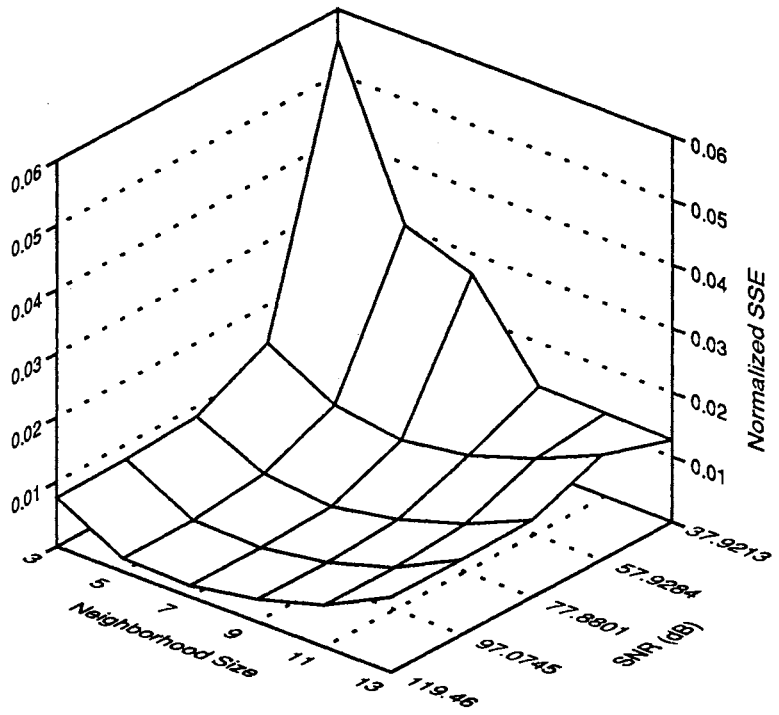

Figure 24: SSE_N vs. SNR and ρ

Experiment 7: Ability to Discriminate Overlapping SQs

The primary reason for choosing the Koch network as a basis for SQNET1 was its ability to form breakpoints to separate image regions. It was believed that these would allow the network to discriminate SQs which touched in the image, but were still best modeled as separate objects. Naturally, there is a tradeoff between this discriminatory capability and the noise tolerance. An experiment was designed to determine the discriminatory capability of the network given a reasonable degree of noise immunity. The dataset is similar to that of figure 15, but for each run the difference in y-position of the two

SQs was reduced. Figure 25 shows the results for differences of 10.0 to 0.0. The network performs well until the separation becomes less than 4. At a separation of 3 the preprocessing step is still able to distinguish the SQs, but SQNET1 discards the discontinuity between the two SQs. The preprocessing step cannot distinguish the next two datasets, nor can SQNET1.

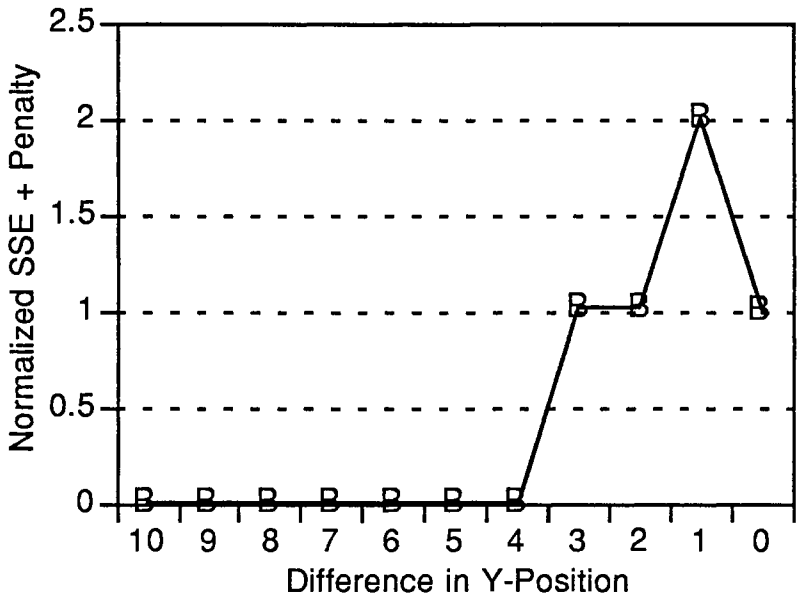

Figure 25: SSE_N vs. Separation of SQs

CHAPTER V

TWO-DIMENSIONAL SUPERQUADRIC NETWORK

The results from SQNET1 were encouraging enough to investigate extending it to a 2D network for the estimation of 3D SQs (SQNET2). Once again, Koch's lead was followed and a vector version of his objective function was created that included the SQ model and the larger local neighborhood. This new objective function (36) is one of the unique contributions of this course of research, and was one of its primary goals. However, extending (32) to two dimensions is not as simple as the extension for Koch's data smoothing network. As with his network, the breakpoint terms must be extended to encourage straight lines and penalize adjacent parallel edges. Intersecting lines should also be accounted for. This extension is discussed in [35] and was adopted unchanged for SQNET2. A more difficult problem was the need to reformulate the product term which removes data points from the parameter estimation if there is an intervening discontinuity. The difficulty of doing this for all pixels in the small neighborhood around the parameter vector of interest led me to a simpler formulation. Instead of using all the pixels, only those in the cross-shaped section centered on the parameter vector are used (see figure 26). This allows a much simpler determination of the effect of intervening discontinuities when deciding if a data point should be included or excluded from the minimization.

$$\chi^2 = C_I \sum_{i=\rho}^{N_r-\rho-1} \sum_{j=\rho}^{N_c-\rho-1} (\underline{\lambda}_{i+1,j} - \underline{\lambda}_{i,j})^2 (1-h_{i,j}) + (\underline{\lambda}_{i,j+1} - \underline{\lambda}_{i,j})^2 (1-v_{i,j})$$

$$+ \sum_{i=\rho}^{N_r-\rho-1} \sum_{j=\rho}^{N_c-\rho-1} \left(\sum_{s=i-\rho}^{i+\rho} (1-F(\underline{\lambda}_{i,j},\underline{x}_{s,j}))^2 \prod_{k=\lfloor i,s \rfloor}^{\lceil i-1,s-1 \rceil} (1-v_{k,j}) \right.$$

$$\left. + \sum_{t=j-\rho,t\neq j}^{j+\rho,t\neq j} (1-F(\underline{\lambda}_{i,j},\underline{x}_{i,t}))^2 \prod_{k=\lfloor j,t \rfloor}^{\lceil j-1,t-1 \rceil} (1-h_{i,k}) \right)$$

$$+ C_C \sum_{i,j} (h_{i,j} + v_{i,j}) + C_P \sum_{i,j} (h_{i,j}\, h_{i,j+1} + v_{i,j}\, v_{i+1,j})$$

$$+ C_L \sum_{i,j} h_{i,j} \left[(1-h_{i+1,j}-v_{i,j}-v_{i,j+1})^2 + (1-h_{i-1,j}-v_{i-1,j}-v_{i-1,j+1})^2 \right]$$

$$+ v_{i,j} \left[(1-v_{i,j+1}-h_{i,j}-h_{i+1,j})^2 + (1-v_{i,j-1}-h_{i,j-1}-h_{i+1,j-1})^2 \right] \qquad (36)$$

This mildly hideous expression is the 2D analog of (32). N_r and N_c are the number of rows and columns in the image, respectively. The vector of parameter estimates at pixel (i,j) is $\underline{\lambda}_{i,j}$. The breakpoint between pixels (i,j) and (i+1,j) is $h_{i,j}$, while $v_{i,j}$ is the breakpoint between sites (i,j) and (i,j+1). Because of the results of the previous chapter, $h_{i,j}$ and $v_{i,j}$ are thresholded versions of underlying state variables. If pixel (i,j) projects onto an object, the xyz coordinates of that patch of the object are in $\underline{x}_{i,j}$. This is set to a sentinel value if the pixel projects onto the background. $F(\underline{\lambda}_{i,j}, \underline{x}_{s,t})$ is the SQ inside/outside function which measures the fit of the data from pixel (s,t) to the parameters at pixel (i,j). The product terms exclude data points from contributing to the SQ model-based fit if there is a discontinuity between them and the parameter vector of interest. The C_I constant weights the importance of the interpolation term relative to that of the model-based term. The C_C term imposes a constant cost for each discontinuity, while the C_P term penalizes parallel discontinuities at adjacent sites. Finally, the C_L term promotes the formation of continuous lines and discourages intersections and discontinuous line segments.

General Organization of SQNET2

The organization of SQNET2 is illustrated in figure 26. Two views, top and orthographic, are given. Both show a single parameter vector, its associated data neighborhood, and the breakpoints that go with the data neighborhood. The rest of the network is the obvious extension, with a parameter vector for every pixel except for the ρ pixel border at each edge. A horizontal breakpoint is placed between every pixel on the same row, while vertical breakpoints are placed between all pixels in the same column. The data neighborhood is $2\rho+1$ by $2\rho+1$, but as mentioned earlier, not all the pixels within the neighborhood are used. Only the pixels in the same row or column as the parameter vector are used in the data neighborhood. This is illustrated by the highlighting of those data points.

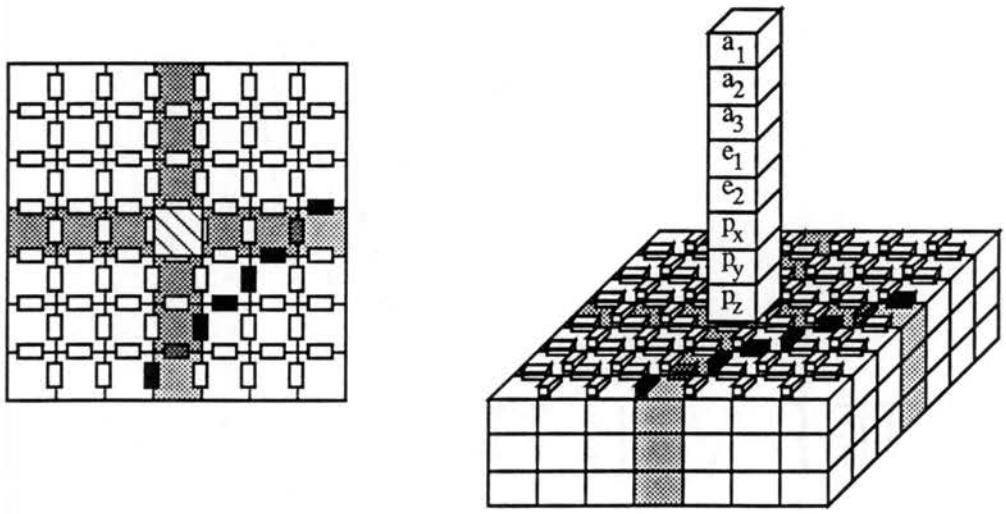

Figure 26: Organization of 2D SQNET (a) Top View (b) Orthographic View

Important discontinuities in the 2D image correspond to the borders of objects. The line of highlighted breakpoints illustrates this. Two of the breakpoints are highlighted with a slightly lighter pattern. These breakpoints, which lie on the cross-shaped region, are the ones actually used in the parameter estimation. The data points that are excluded from the estimation are highlighted with a lighter pattern than the data points that are used in the estimation.

The 2D network also uses an interpolation term, although the neighboring parameter vectors are not shown in figure 26. The parameter vectors used are the 4 nearest neighboring vectors. No interpolation is performed if the intervening breakpoint is set. As with SQNET1, a hybrid minimization algorithm is used where the SQ model term is minimized by a LM technique, while simple gradient descent was used to do the parameter interpolation and discontinuity updates.

This network organization has the disadvantage of using large numbers of parameters, a 512^2 image would have to estimate over 2 million parameters! However, the hybrid technique means that each pixel has a separate minimization, so we are running 512^2 separate minimizations, each estimating 8 parameters, not one minimization with more than 2 million parameters. Also, background pixels do not participate in the estimation. The approach has the advantage of simplicity. Using fewer parameter vectors would force us to deal with the issue of how to handle object boundaries that intersected the region of the images described by a single parameter vector. As with the 1D network, it was judged that this would be an unnecessary complication for the initial investigation into the technique.

The objective function for SQNET2 proved more difficult to minimize than that of SQNET1. The main problem was that much greater attention had to be paid to the numerical aspects of the SQ model term, specifically to the $(x-x_0)$, $(y-y_0)$, and $(z-z_0)$ factors. Because we can view the SQs from any orientation, we will frequently see the intersection of the SQ with the planes where those factors vanish. Those intersections would frequently blow up due to trying to take the log of zero. To combat this, the SQ

portion of the minimization was special cased to deal with the possibility of these factors going to zero. Anytime one of those factors became less than a certain tolerance (0.1), a special cased version of the model and its partials was used that assumed the factor was 0. This avoided the numerical problems.

Preprocessing and Postprocessing

SQNET2, like SQNET1, uses an iterative minimization algorithm. Therefore, its speed and accuracy are dependant upon the initial parameter estimates. This section describes the input data, the preprocessing performed to obtain the initial parameter estimates, and the postprocessing performed to convert regions of parameter estimates into the vectors of parameter estimates that are the desired output of the technique.

All of the images used in this study were synthetic. A freely-available graphics package, SIPP [69], was modified to produce range images such as those from laser rangefinders. Pixels that project onto an object are assigned a depth value equal to the distance from the eyepoint to that patch on the imaged object. Pixels that do not project onto objects are assigned a sentinel depth value. The SQs in the images are actually fairly coarse polyhedral approximations to SQs, with 16 divisions around the equator and 9 lines of latitude.

The first step in preprocessing the data for SQNET2 uses an inverse perspective projection to convert the range information in each pixel into an xyz triple, based on the pixel's row and column indices, the depth value in the pixel, and an eye position. The resulting three-band image is referred to as the triples image. Sentinel pixels are assigned a sentinel triple.

The triples image is scanned by a region labelling procedure. An edge-detection filter is run over the triples image in order to find region boundaries. These boundaries form closed curves. The 4-connected regions within those curves are marked as regions. The output is a single band image where all pixels within these regions are assigned a

region label. The labels are small integers in the range from one to the number of regions. Sentinel pixels are assigned a region label of zero.

The algorithm for the initial parameter estimation uses the regions and triples images as input, and produces two output images. The first output image has eight bands, the second has two bands. The eight bands correspond to the eight parameters that need to be estimated at each non-sentinel pixel (three for position, three for size, and two for the SQ shape parameters). The two-band image contains the horizontal and vertical discontinuity estimates. The triples and regions image are scanned twice in raster order. During the first pass, the elements in the discontinuity image are set as region borders are encountered. Sentinel pixels have all their surrounding discontinuity elements set. Within regions, a lookup table of the minimum and maximum x, y, and z values is built. These are used to form a 3D version of the bounding box described in the preprocessing for SQNET1 (see figure 12). At the end of the first pass, the bounding box information in the lookup table is used to compute the initial parameter estimates. The size parameters, a_1, a_2, and a_3 are set to 1/2 the size of the box in the respective directions. The position parameters, x_0, y_0, and z_0 are set to the center of the box. The shape parameters, e_1 and e_2 are always set to 1.0, which means that objects are assumed to be ellipsoids. These parameter estimates are kept in a lookup table indexed by the region label. During the second pass, each non-sentinel pixel looks up its parameter vector in the lookup table, using its region label as the index. Pixels outside the regions are set to sentinel values.

The output of SQNET2 are refined estimates of the SQ parameters and discontinuities for each pixel. These must be post-processed to yield a list of parameter vectors, with one parameter vector for each SQ found in the image. The first step in postprocessing is to run the region detector over the parameter estimates image in order to find any new SQs discovered in the image. The median of the parameters within each of these regions is then found, and is output as the estimate for that region.

Performance was evaluated using a technique exactly analogous to that for SQNET1. The scoring program read the final regions image, the final parameter estimate vectors, and the triples image. The estimates for a region were propagated to all pixels in the region. The triples data for that pixel was then used in the expression $(1 - F(\underline{\lambda}_{i,j}, \underline{x}_{i,j}))^2$, and the squared errors summed. This SSE was normalized by the number of active pixels. A penalty was added if the number of SQs discovered differed from the number of SQs used to generate the input image.

Results

After determining a likely set of constants for the various costs and minimization controls, several experiments were run to test the ability of SQNET2 to estimate the SQ parameters. As for SQNET1, there is no *a-priori* method for determining the best values for the weights in the objective function and the various iteration limits, convergence tolerances, and update frequencies in the minimization procedure. The simple search procedure was repeated in order to find a set of constants that would successfully estimate the parameters for a range of noise levels. The best performing constants found are given below in Table 3. These values were used in all subsequent experiments unless mentioned otherwise.

TABLE 3

BEST PERFORMING CONSTANTS
FOR SQNET2

Parameter	Value
C_I	1.0
C_C	7.0
C_P	1.0
C_L	1.0
ρ	3
ss	0.1
chi_tol	0.1
lm_iters	3
iters	2
h_freq	1

As with SQNET1, a range of experiments were devised to test various invariance properties of SQNET2. However, because of the results from SQNET1, not all the experiments were repeated. Specifically, because SQNET1 was not able to discriminate overlapping SQs that the preprocessing could not discriminate, no attempt was made to have SQNET2 discriminate overlapping SQs.

Experiment 1: Position Invariance

The first experiment was to verify the position invariance of SQNET2. The data set for this experiment consisted of a unit sphere moved in ten steps along a line through the origin, perpendicular to the viewing axis. Figure 27 shows the results of this experiment. As for SQNET1, there are some differences in SSE_N, but they are at a level that can adequately be accounted for by the interaction between sampling points and the coarse polygonal approximation to a SQ in for the input data.

Figure 27: SSE_N vs. Position

Experiment 2: Size Invariance

In this experiment the size invariance of SQNET2 was investigated. Varying the size of the SQ for a fixed sampling density is equivalent to varying the sampling density for a fixed size SQ, so this experiment also provides information on how densely we must sample the SQ in order to ensure adequate recovery. The data set for this experiment consisted of a singe SQ ellipse placed at the origin. Its size was varied from covering 45 to 4096 pixels.The local neighborhood size was also varied. Figure 28 shows that the quality of the fit is good until the SQ becomes quite large. At this point the SQ fills the 4096 pixel image used in this test, therefore portions of the SQ are outside of the picture. The loss of these areas of high curvature is believed to be the main contributor to the poor fit. The neighborhood size has only a minor effect on the fit. We see that the recovery works quite well for small numbers of data points, at the smallest size the SQ is contained in a 7x7 neighborhood.

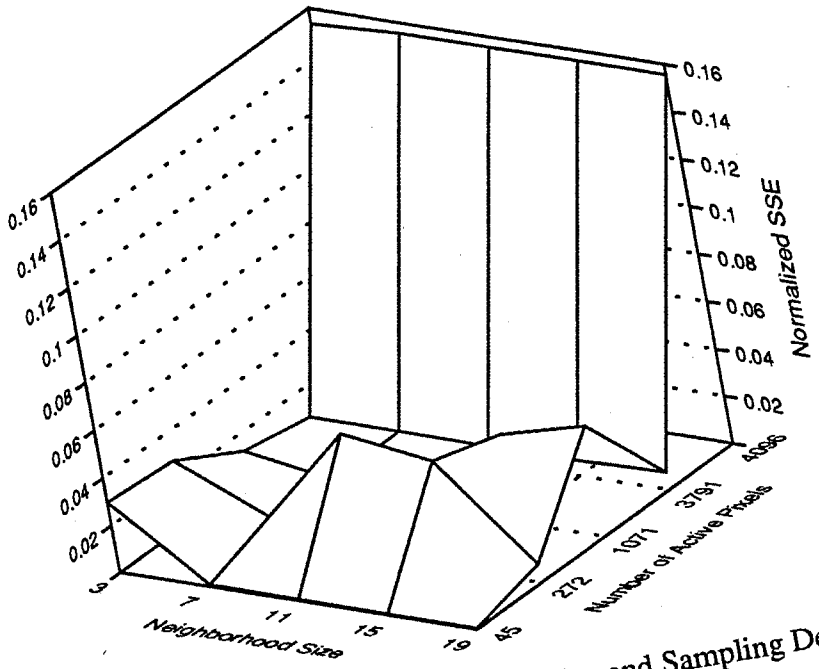

Figure 28: SSE_N vs. Neighborhood Size and Sampling Density

Experiment 3: Shape Invariance - Aspect Ratio

This experiment was designed to test the invariance of SQNET2 to changes in the aspect ratio of the SQs it is trying to model. Two sets of experiments were run. The first, shown in Figure 29 A, varied the a_1 and a_2 parameters between 0.2 and 4.0 while holding a_3 at 1.0. The second, shown in figure 29 B,varied a_1 and a_3 between 0.2 and 4.0 while holding a_2 at 1.0. In order to keep the number of pixels covered by the differently shaped SQs roughly constant, the size parameters were actually normalized so that $<a_1, a_2, a_3>$ was unit length.

We see that the aspect ratio does have an effect. Generally, the recovery is less accurate as the SQ gets further away from a sphere. This is especially true for the very high aspect ratios of approximately 20:1. However, the explanation for this effect is more involved than a simple lack of invariance to changes in aspect ratio. The three worst-performing cases in figure 29 are for highly elongated ellipsoids. Those three ellipsoids are aligned with the axes of the world coordinate system, and are viewed from a position along the direction vector <1, 1, 1>. This is illustrated below in figure 30. Recall that the parameters are fit to data from a cross-shaped region. This is the worst choice give the orientation of the SQs in the image, since only a few pixels at the center of the cross will lie on the object.

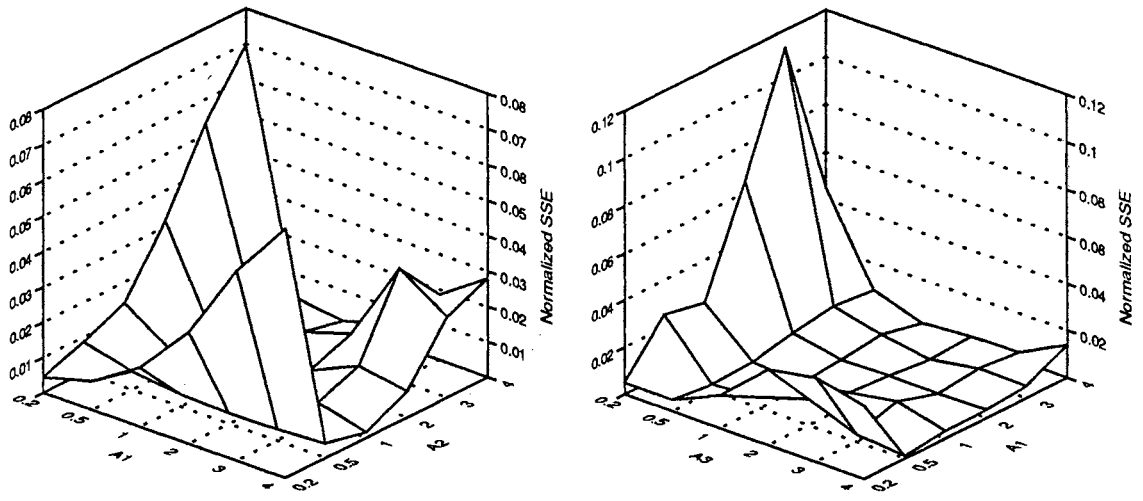

Figure 29: SSE_N vs. Aspect Ratio

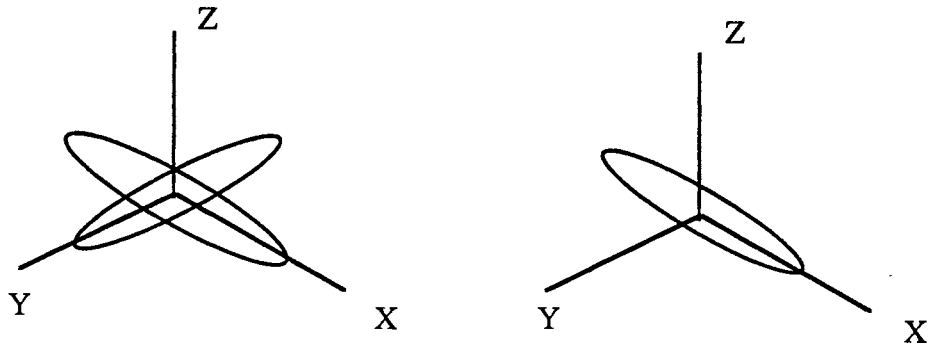

Figure 30:Appearance of SQs in Aspect Ratio Test

Experiment 4: Shape Invariance - SQ Shape Parameters

The previous experiment measured the effects of aspect ratio on the accuracy of SQNET2's estimates. We also need to know if the technique displays any significant sensitivity to the shape parameters of the SQs it is trying to model. The input data for this experiment was a single unit SQ positioned at the origin. Both e_1 and e_2 were varied from 0.2 to 2.0 for a total of 25 runs. The results, presented in figure 31, again show that the estimation error increases as the SQ shape gets further away from spherical. The errors are greater than those for high aspect ratios. The most probable cause for these errors was also mentioned in the previous chapter, namely the lack of information provided by flat faces. All four corners of the figure below correspond to objects with flat faces.

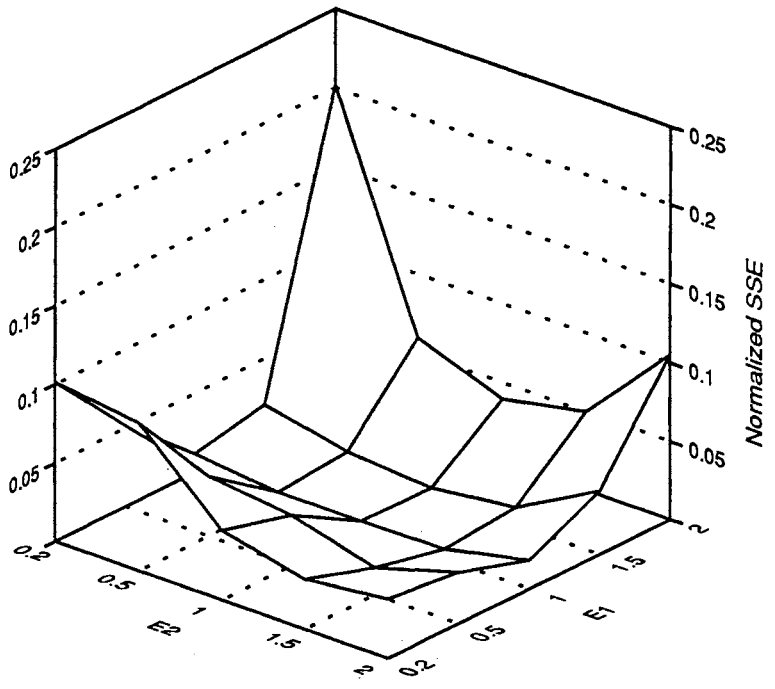

Figure 31: SSE_N vs. SQ Shape Parameters

Experiment 5: Viewpoint Invariance

For SQNET1, the viewing direction was always down the y-axis. SQNET2 was run on datasets that did not have this restriction. Therefore, this experiment was designed to measure any sensitivity the technique might have to viewpoint. The dataset for this experiment consisted of a single SQ at the origin whose shape parameters were varied from 0.2 to 2.0. This means its shape varied from approximately cubical, through spherical, to an octahedron. Each shape was viewed from seven positions in the first octant whose longitude and latitude were both equal to the viewing angle specified in figure 32. Since the SQs are symmetric, this is considered to be adequate coverage. Viewing directions directly along axes are avoided because of the principle of general position. The results of the experiment show that there is a sensitivity to viewing position, especially for cubical and octahedral shapes. This is expected due to the results of the previous experiment. The smaller sensitivity to viewpoint noted for the spherical shape is attributed to the coarse polyhedral approximation to a sphere actually used.

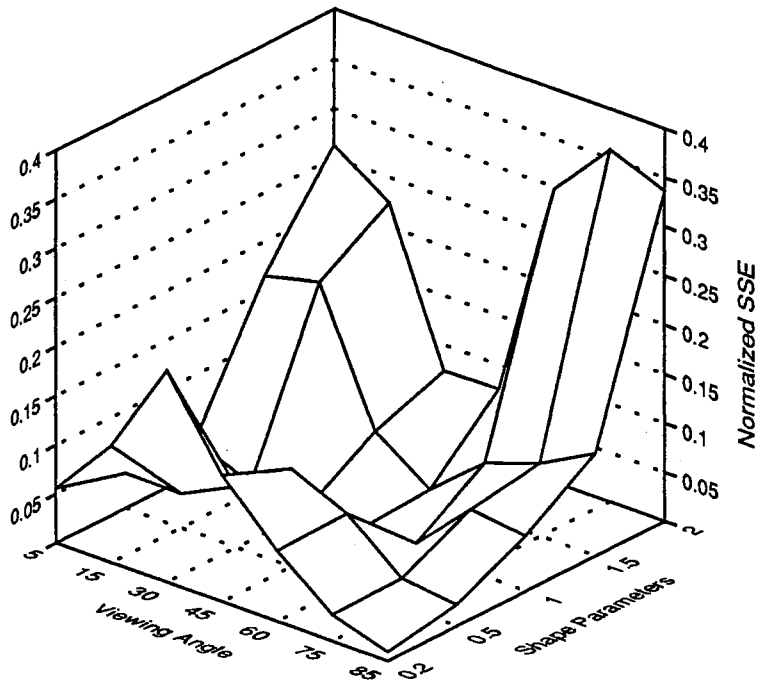

Figure 32: SSE_N vs. Viewing Angle and Shape Parameters

CHAPTER VI

CONCLUSIONS

Summary

This dissertation began by mentioning some of the motivations for machine vision and briefly reviewing some of the 3D object modeling techniques used in the field. In Chapter II we described an interesting geometric solid, the superquadric (SQ), and reviewed the work done using it as a modeling primitive in machine vision. Special attention was paid to the work of Solina [58], and several advantages and disadvantages of his technique were noted. We cited a unique parallel implementation of his method [14] to show that one of the disadvantages, speed, could be overcome. We also noted that he was not exploiting the very powerful property of image coherence. Chapter III reviewed the basics of neural networks, giving details on the operation of a few example networks. Close attention was paid to a neural network developed by Christof Koch and his colleagues [35]. Its operation was discussed and results presented. We also noted the results of a unique comparison between his network and a median filter [15].

The remainder of the dissertation described attempts to use a modified Koch network to exploit image coherence in an attempt to overcome the problems noted with Solina's approach to SQD. Chapter IV described initial efforts which used a new vector extension to Koch's 1D network. The objective function of this network (SQNET1), which incorporates a term to model 2D superellipses, was another of the unique developments in this thesis. The objective function proved too difficult to minimize by the simple gradient crawl of the Koch network, so a hybrid minimization algorithm was developed. This new minimization algorithm was able to minimize the objective function, although certain

limitations were noted. One of these limitations, recovery of squares, was expected. The other, an inability to find the boundary between nearby SQs, was disappointing since it was a major motivation for the technique. Chapter V described the extension of SQNET1 to a 2D network (SQNET2) for modeling 3D superellipsoids. This unique network, which was the goal of the thesis, was also successful in estimating the SQ parameters in most cases. However, the problems evident in SQNET1 were even more noticeable in SQNET2.

Suitability of Koch Network Approach

As was noted in Chapter IV, a straightforward modification of the Koch network was not able to minimize the portion of the objective function that was based on the SQ inside/outside function (3). Recent discussions with neural network researchers at Cambridge have shed additional light on the difficulties experienced [1, 2, 21]. Hopfield networks can be analyzed in terms of the subspaces generated by the eigenvectors of their connection matricies. These subspaces take the form of hyperplanes within the N-dimensional hypercube of possible solutions, where N is the number of units in the network. If the network is attempting to solve a combinatorial problem, the solution is further constrained to lie at one corner of the hypercube. This additional constraint is quite powerful, and can be used to markedly improve the performance of the network. Without this extra constraint, Hopfield networks are poorly suited to non-quadratic, non-combinatorial optimization problems. Its simple gradient crawl leaves it open to the classic problem of long shallow valleys.

Suitability of Hybrid Method

While the Koch network was not capable of minimizing the SQ portion of the objective function, I still thought that the Koch network's approach to simultaneously discovering image regions and image edges was interesting. I also thought that the objective function it had inspired (32) was a natural way for exploiting image locality in

order to overcome the problems noted with Solina's network. Therefore I investigated an approach as close as possible to the spirit of the Koch network, but used the more powerful Levenberg-Marquardt (LM) minimization technique to handle the SQ portion of the objective function. By exploiting the natural parallelism in the problem, the hybrid approach also addresses the storage problem associated with minimization algorithms that compute or approximate the Hessian matrix. While neural network researches would like to use such methods to minimize their objective functions, they frequently have so many parameters to estimate that such methods are impractical.

The hybrid method proved that it was capable of minimizing the objective function and obtaining good parameter estimates. The characteristics of SQNET1 were tested to find out its sensitivity to noise in the data, sampling density, aspect ratio, etc. The technique was usually successful in finding quite good estimates for the underlying superquadrics. Some problems were noted as shapes tended toward squares, due to the loss of information from the sides of the SQ. We also saw that the technique was not able to discriminate SQs if they were positioned very close together. This was disappointing, since it was one of the main motivations for investigating the technique.

Despite these problems, SQNET1 appeared promising enough to warrant investigating its extension to recovering the parameters for 3-D SQs. SQNET2 was implemented and tested to discover its invariance to position, size, aspect ratio, shape parameters and viewing position. SQNET2 was also successful in estimating the SQ parameters under most conditions, but exhibited greater sensitivity to the situations which gave SQNET1 trouble. Basically, the performance fell off as the SQ shape got further away from a sphere. There were two causes for this. The first was that the cross-shaped neighborhood performed poorly if the orientation of the SQ avoided most of the pixels in the neighborhood. This is natural, but it does indicate that the cross-shaped neighborhood is not a good choice for general purpose use. The second problem was that SQs with flat rather than curved faces were more difficult to recover. Most of this difficulty was ascribed

to a lack of curvature information. This network's sensitivity to viewing position is related to this lack of information. Performance is best from viewpoints that show as many faces as possible. The rest of the difficulty was ascribed to an increasingly non-quadratic error function as the SQ exponents were moved away from 1.0. These exponents make the SQ function difficult to minimize. Careful handling of the $x = x_0$, $y = y_0$, and $z = z_0$ planes was necessary in order to avoid numerical difficulties.

Within the limitations noted above, the hybrid technique worked well. However, this is mostly because of the capabilities of the LM algorithm. The difference in convergence order between the LM algorithm and the gradient crawl algorithm means that the interpolation term does not have enough time enforce smoothness of the parameter estimates before the LM algorithm begins to overfit.

Comparison with Solina's Technique

The goal of the research was to improve Solina's method by using Koch's network as a framework to take advantage of image coherence. Earlier we noted three particular problems with Solina's method. The first was speed, the second was the limited number of SQs that could be recovered, and the third was that the recovery time was serial in the number of SQs recovered. How well did the hybrid approach meet the goals of the project? Let us examine the problems in order.

First, speed. The hybrid network is not faster then Solina's technique, in fact it's computational complexity is greater. As mentioned earlier, for each iteration of the LM algorithm, the network requires MN evaluations of the objective function and its partial derivatives, where M is the number of active data points and N is the size of the local neighborhood. Solina's only requires M evaluations per LM step. The hybrid approach also has the expense of the interpolation and discontinuity terms. However, both approaches are very amenable to parallel implementation, so the problems with speed are not serious.

The second problem with Solina's method was that his multiple object recovery technique severely limited the number of SQs that could be estimated. The SQ network approach can handle large numbers of SQs in an image. However, this is not the large improvement it initially appears to be. Solina's technique can sometimes disambiguate SQs that overlap in image space, or even interpenetrate in object space. The hybrid technique has not ever managed to form the continuous line of discontinuity elements needed to separate SQs that touch. While the hybrid approach can handle many more objects than Solina's technique, they must be surrounded by a border of background pixels, or very easy to separate in a preprocessing step.

The third problem with Solina's approach was that it recovered the SQs serially, with the largest one being estimated first. The hybrid approach can recover all the SQs in parallel, but again, this is a minor improvement. If the regions had been separated in a preprocessing step, Solina's method could be extended to recover them in parallel. However, this preprocessing step could well be a mixed blessing. While Solina's technique would be able to discover if the preprocessing step had incorrectly grouped the data from two SQs into one region, it would not be able to do the converse.

In summary, the SQNET2 technique works, but not as well as I had initially hoped. It does exploit image coherence, which gives it some advantages over Solina's approach. However, these advantages could be matched and exceeded if Solina's approach were developed further to use pre-segmented range data while not excluding the possibility of merging regions. The actual SQ parameter estimation of SQNET2 works quite well for many situations, but appears inferior to that of Solina's for objects with flat faces. The small neighborhood size limits the information available to estimate any one parameter vector. This means that pixels in the middle of a face do not have enough information to make reliable estimates. The difference in convergence order between the LM and the gradient crawl algorithms means that there is not enough time for information to propagate across regions and improve those estimates before the LM technique begins to overfit.

Guidelines for Future Research

Several simplifying assumptions were made in the course of this research. For example, the objective functions in this study do not attempt to model rotated or deformed SQs. Removing these restrictions would be a subject for a straightforward extension of the present research. However, they served their purpose in allowing an examination of the suitability of an approach to object modeling before getting bogged down in a mass of details. Another obvious extension would be to use real rangefinder data rather than the artificial data used in this study.

Another possible course of research which closely follows this one would be to pull the interpolation term into the LM minimization algorithm, while still using a different method to estimate the breakpoints. This would overcome any problems due to the difference in convergence order between the LM and gradient crawl methods. The product terms which exclude data points could also be reformulated to use other than cross-shaped neighborhoods.

Only slightly further afield from this study would be to investigate network organizations other than one parameter vector per pixel. Other spatially organized neural networks, such as Kohonen's Self-Organizing Map [37], might provide a better model for this purpose. Certainly, spatial coherence should be used in the preprocessing stages. However, the object modeling stage should be able to overcome errors in the preprocessing algorithm's assignment of points to candidate SQs. Extensions to Solina's method are a topic that might prove fruitful.

A topic of more general interest, and one that I think will be pursued by may people who will never know of this thesis, is the development of minimization algorithms that can handle large numbers of parameters while not sacrificing all the benefits of methods that approximate the Hessian matrix. The hybrid network developed during this study was able to do so by taking advantage of parallelism inherent in the problem, parallelism that was

due to image coherence. The search for general methods should keep many researchers busy for several years, and a successful method would prove a great benefit to many fields.

Finally, as a more general guideline to vision researchers, it appears that attempting to have an object modelling technique simultaneously perform data segmentation is a bad idea. While it is an attractive idea to avoid the "chicken and egg" problem that has bedeviled segmentation for so long, this does not appear to be the way to do it. However, the limitations of the standard low level to medium level to high level pipeline remain. The object modeling phase must be able to overcome bad decisions made in preprocessing stages. Pavlidis [44] believes that this layered approach is one of the main impediments to progress in machine vision. Finding algorithms to overcome it is therefore crucial to significant advances in this field.

BIBLIOGRAPHY

[1] Aiyer, S.V.B., Niranjan, M., and Fallside, F., "Theoretical Investigation into the Performance of the Hopfield Model," Tech. Rep., CUED/F-INFENG/TR 36, 1990.

[2] Balakrishnan, S.V., personal communication.

[3] Ballard, D.H. and Brown, C.M., *Computer Vision.* Englewood Cliffs, N.J.: Prentice-Hall, 1982.

[4] Barr, A.H., "Superquadrics and Angle-Preserving Transformations," *IEEE Comp. Graphics and Applications*, vol. 1, no. 1, pp. 11-23, Jan. 1981.

[5] Barr, A.H., "Global and Local Deformations of Solid Primitives," *Computer Graphics*, vol. 18, no. 3, pp. 21-30, July 1984.

[6] Besl, P.J. and Jain, R.C., "Three-Dimensional Object Recognition," *Computing Surveys*, vol. 17, no. 1, pp. 75-145, March 1985.

[7] Biederman, I., "Recognition-by-components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 2, pp. 115-147, 1987.

[8] Binford, T.O., "Visual Perception by Computer," in *Proceedings IEEE Conf. on Systems and Control,* Miami, FL, 1971.

[9] Boult, T.E. and Gross, A.D., "Recovery of Superquadrics from Depth Information," in *Proceedings of Spatial Reasoning and Multi-Sensor Fusion Workshop,* pp. 128-137, 1987.

[10] Braid, I.C., "The Synthesis of Solids Bounded by Many Faces," *Communications of the ACM*, vol. 18, no. 4, pp. 209-217, April 1975.

[11] Carpenter, G.A., Cohen, M.A., and Grossberg, S., "Computing with Neural Networks: Technical Comments," *Science*, vol. 235, pp. 1226-1227, March 1987.

[12] Carpenter, G.A. and Grossberg, S., "A Massively Parallel Architecture for a Self-Organizing Pattern Recognition Machine," in *Neural Networks and Natural Intelligence*, Grossberg, S., Ed. MIT Press, 1988.

[13] Daniel, Jr., R.E., "Superquadric Description on Large Arrays of Bit-Serial Processors," Master's thesis, K. Teague, Advisor, Okla. State Univ., July 1987.

[14] Daniel, Jr., R.E., "Parallel Nonlinear Optimization," in *Proceedings of the Fifth Distributed Memory Computing Conference,* 1990.

[15] Daniel, R.J. and Teague, K., "A Connectionist Technique for Data Smoothing," in *Proceedings of the Fifth Distributed Memory Computer Conf.*, 1990.

[16] Davies, E.R., *Machine Vision: Theory, Algorithms, Practicalities*. San Diego, CA: Academic Press, 1990.

[17] Fahlman, S.E., "An Empirical Study of Learning Speed in Back-Propagation Networks," Carnegie-Mellon University Tech. Rep., CMU-CS-88-162, 1988.

[18] Fahlman, S., "The Cascade-Corellation Model," Carnegie-Mellon University Tech. Rep., CMU-CS-90-100, 1990.

[19] Ferrie, F.P., Lagarde, J., and Whaite, P., "Darboux Frames, Snakes, and Super-Quardics: Geometry from the Bottum-Up," in *Proceedings Workshop on Interpretation of 3D Scenes,* IEEE Comp. Soc. Press, 1989, pp. 170-176.

[20] Foley, J.D. and van Dam, A., *Fundamentals of Interative Computer Graphics*. Reading, Ma: Addison-Wesely Publishing Co., 1985.

[21] Gee, A., personal communication.

[22] Geman, S. and Geman, D., "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. PAMI-6, no. 6, pp. 721-741, Nov. 1984.

[23] Grossberg, S., "Competitive Learning: From Interactive Activation to Adaptive Resonance," in *Neural Networks and Natural Intelligence*, Grossberg, S., Ed. MIT Press, 1988.

[24] Gupta, A., Bogoni, L., and Bajcsy, R., "Quantitative and Qualitative Measures for the Evaluation ot the Superquadric Models," in *Proceedings Workshop on Interpretation of 3D Scenes,* IEEE Comp. Soc. Press, 1989, pp. 162-169.

[25] Healey, G. and Binford, T.O., "Local Shape from Specularity," *Computer Vision, Graphics, and Image Processing*, vol. 42, pp. 62-86, 1988.

[26] Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554-2558, April 1982.

[27] Hopfield, J.J., "'Unlearning' has a stabilizing effect in collective memories," *Nature*, vol. 304, pp. 158-159, July 1983.

[28] Hopfield, J.J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, pp. 3088-3092, May 1984.

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

[30] Horn, B.K.P., *Robot Vision.* MIT Press, 1989.

[31] Hummel, J.E. and Biederman, I., "Dynamic Binding in a Neural Network for Shape Recognition," Univ. of Minnesot Dept. of Psychology, Image Understanding Lab Tech. Rep., 90-5, August 1990.

[32] Ikeuchi, K. and Horn, B.K.P., "Numerical Shape from Shading and Occluding Boundaries," *Artificial Intelligence*, vol. 17, pp. 141-184, 1981.

[33] Ikeuchi, K., "Shape from Regular Patterns," *Artificial Intelligence*, vol. 22, pp. 49-75, 1984.

[34] Johansson, E.M., Dowla, F.V., and Goodman, D.M., "Backpropagation Learning for Multi-Layer Feed-Forward Neural Networks Using the Conjugate Gradient Method," Lawrence Livermore Natl. Lab, Preprint UCRC-JC-104850, 1990.

[35] Koch, C., Marroquin, J., and Yuille, A., "Analog 'neuronal' networks in early vision," *Proc. Nat. Acad. Sci. USA*, vol. 83, pp. 4263-4267, June 1986.

[36] Koenderink, J.J. and Van Doorne, A.J., "The Internal Representation of Solid Shape with respect to Vision," *Biological Cybernetics*, vol. 32, pp. 211-216, 1979.

[37] Kohonen, T., *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 1984.

[38] Kriegman, D.J. and Ponce, J., "Computing Exact Aspect Graphs of Curved Objects: Solids of Revolution," in *Proceedings of the Workshop on Interpretation of 3D Scenes*, IEEE Computer Society Press, 1989, pp. 116-122.

[39] Lee, C.H. and Rosenfeld, A., "Improved Methods of Estimating Shape from Shading Using the Light Source Coordinate System," *Artificial Intelligence*, vol. 26, pp. 125-143, 1985.

[40] Lippmann, R.P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[41] Lowe, D.G., "The Viewpoint Consistency Constraint," *International Journal of Computer Vision*, vol. 1, no. 1, pp. 57-72, 1987.

[42] Metaxas, D. and Terzopoulos, D., "Constrained Deformable Superquadrics and Nonrigid Motion Tracking," in *Proceedings International Conference on Computer Vision and Pattern Recognition (CVPR'91)*, 1991, pp. 337-343.

[43] Minsky, M. and Papert, S., *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[44] Pavlidis, T., "Why progress in machine vision is so slow," *Pattern Recognition Letters*, vol. 13, pp. 221-225, April 1992, North-Holland.

[45] Pentland, A.P., *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*. Norwood, N.J.: Ablex, 1986.

[46] Pentland, A.P., "Shading into Texture," *Artificial Intelligence*, vol. 29, pp. 147-170, 1986.

[47] Pentland, A.P., "Perceptual Organization and the Representation of Natural Form," SRI Intl. Tech. Rep., 357 (Revised), July 29 1986.

[48] Pentland, A.P., "A New Sense for Depth of Field," *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. PAMI-9, no. 4, pp. 523-531, 1987.

[49] Pentland, A.P., "Toward an Ideal 3-D CAD System," in *SPIE Conf. on Mach. Vision and the Man-Machine Interface,* 1987.

[50] Pentland, A.P. and Bolles, R., "Learning and Recognition in Natural Environments," SRI Intl. Tech. Note 421, 1987.

[51] Pentland, A.P. and Williams, J., "Perception of Non-Rigid Motion: Inference of Shape, Material, and Force," in *Proceedings International Joint Conference on Artificial Intelligence (IJCAI '89),* 1989.

[52] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge, England: Cambridge University Press, 1988.

[53] Requicha, A.A.G.V.H.B. and Voelcker, H.B., "Boolean Operations in Solid Modeling: Boundry Evaluation and Merging Algorithms," *Proceedings of the IEEE*, vol. 73, no. 1, pp. 30-44, Jan. 1985.

[54] Roberts, L.G., "Machine Perception of three-dimensional solids," in *Optical and Electro-optical Information Processing*, Tippet, J.P. *et al*, Eds. Cambridge, MA: MIT Press, 1965.

[55] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation," in *Parallel and Distributed Processing: Explorations in the Microstructure of Cognition*, Rumelhart, D.E. and Hinton, G.E., Eds. Cambridge, MA: MIT Press, 1986.

[56] Rumelhart, D.E., Hinton, G.E., and Group, P.P.R., *Parallel and Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press, 1986.

[57] Scales, L.E., *Introduction to Non-Linear Optimization.* New York: Springer, 1985.

[58] Solina, F., "Shape Recovery and Segmentation with Deformable Part Models," Ph.D. thesis, also published as U. Penn. Dept. Comp. and Info. Sci. GRASP Lab Tech. Rept. 128, Univ. of Penn., Dec. 1987.

[59] Solina, F. and Bajcsy, R., "Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations," *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. PAMI-12, no. 2, pp. 131-147, Feb. 1990.

[60] Tank, D.W. and Hopfield, J.J., "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a LinearProgramming Circuit," *IEEE Trans. Circuits and Systems*, vol. 33, no. 5, pp. 533-541, May 1986.

[61] Terzopoulos, D. and Metaxas, D., "Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics," in *Proceedings of the Third International Conference on Computer Vision (ICCV'90),* 1990, pp. 606- 615.

[62] Terzopoulos, D. and Metaxas, D., "Dynamic 3D Models with Local and Global Deformations: Dynamic Superquadrics," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, pp. 703-714, July 1991.

[63] Terzopoulous, D., Witkin, A., and Kass, M., "Energy Constraints on Deformable Models: Recovering Shape and Non-Rigid Motion," in *Proceedings American Assoc. Artif. Intell. (AAAI-87)*, 1987, pp. 755-760.

[64] Tilove, R.B., "Set Membership Classification: A Unified Approach to Geometric Intersection Problems," *IEEE Trans. Computers*, vol. C-29, no. 10, pp. 874-883, Oct. 1980.

[65] Tou, J.T. and Gonzalez, R.C., *Pattern Recognition Principles*. Reading, MA.: Addison-Wesley, 1974.

[66] Ullman, S., *The Interpretation of Visual Motion*. MIT Press, 1979.

[67] Ullman, S. and Basri, R., "Recognition by Linear Combinations of Models," A.I. Memo 1152, MIT AI Lab, Cambridge, MA., August 1989.

[68] Waltz, D., "Understanding Line Drawings of Scenes with Shadows," in *The Psychology of Computer Vision*, Winston, P.H., Ed. New York: McGraw-Hill, 1975.

[69] Yngvesson, J. and Wallin, I., *SIPP 2.0 - A 3d Rendering Package*, available on comp.sources.misc archive sites.

VITA

Ronald Ellison Daniel Jr.

Candidate for the Degree of

Doctor of Philosophy

Thesis: A MASSIVELY PARALLEL APPROACH TO MODELING 3D OBJECTS IN MACHINE VISION

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Springfield, Missouri, September 8, 1960, the son of Ronald E. and Barbara L. Daniel.

Education: Graduated from Putnam City High School, Oklahoma City, Oklahoma, in May 1978; received Bachelor of Science Degree in Electrical and Computer Engineering from Oklahoma State University in May, 1985; received Master of Science Degree from Oklahoma State University in July, 1987; completed requirements for Doctor of Philosophy degree at Oklahoma State University in May, 1993. Member of Eta Kappa Nu, Tau Beta Pi. Life Member of the Cambridge University Cognitive Science Society.

Professional Experience: Teaching Assistant, Department of Electrical and Computer Engineering, Oklahoma State University, August 1984 to December 1985; Graduate Research Assistant, Department of Electrical and Computer Engineering, Oklahoma State University, August 1985 to August 1986; Teaching Assistant, Department of Electrical and Computer Engineering, Oklahoma State University, August 1986 to May 1987; Graduate Research Associate, Department of Electrical and Computer Engineering, Oklahoma State University, May 1987 to September 1990; Research Associate, Department of Engineering, Cambridge University, October 1990 to present.