A Robust Technique for the Estimation of the Deformable Hyperquadrics from Images*

Senthil Kumar and Dmitry Goldgof
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33613
email: kumar,goldgof@figment.csee.usf.edu

Abstract

In this paper, we present a robust technique for the estimation of deformable hyperquadrics from images. Hyperquadrics are volumetric shape models that include superquadrics as a special case. Recovering hyperquadric parameters is difficult not only due to the existence of many local minima in the error function but also due to the existence of an infinite number of global minima (with zero error) that do not correspond to any meaningful shape. An algorithm that minimizes the error-of-fit function without using techniques similar to those presented here will often find itself stuck in "meaningless" minima, even with good initialization. Our algorithm exhibits good convergence behavior and is largely insensitive to initialization.

1 Introduction

The problem of object modeling using superquadrics has received considerable attention in the computer vision literature. Superquadrics are volumetric shape models that can represent a wide range of objects with a small number of parameters. Several researchers have used superquadrics for object modeling, motion analysis and object recognition (Barr [1, 2], Solina and Bajcsy [12], Boult and Gross [3], Gupta [5], Ferrie, Lagarde and White [4] and Raja and Jain [11]). While superquadrics can model a diverse set of objects, they are intrinsically symmetric about the coordinate axes. Barr [2] applied simple deformations like tapering, bending and twisting to superquadrics so as to model a wider range of shapes and in [12], Solina and Bajscy discuss the recovery of such deformable superquadrics from range images. While it is possible to apply more complex deformations to superquadrics, their recovery from sensed data would be very difficult except in the case of simple deformations. In short, while superquadrics are powerful, there exist many naturally occurring objects that they cannot model accurately.

Hyperquadrics are volumetric shape models that are more powerful than superquadrics and include superquadrics as a special case. They were first proposed for computer graphics applications by Hanson [7]. Hyperquadrics can model shapes that are not necessarily symmetric and their geometric bounds are arbitrary convex polyhedra as opposed to superquadrics whose geometric bounds are simple cartesian rectangular parallelopipeds. The power of hyperquadrics arises from the fact that they can model some shapes that cannot be described by deformable superquadrics.

Recently, Han and Goldgof published an article that used hyperquadrics for shape recovery from range data [6]. The hyperquadric parameters were estimated from range data by minimizing an error-of-fit function. In this paper we improve the convergence behavior of the above estimation. We incorporate inequality constraints on the parameters to improve the convergence behavior. With these constraints, even when we start far from the correct minimum, the parameters are pulled very quickly towards the correct minimum and the algorithm converges with a meaningful set of parameters.

2 Hyperquadrics: A Brief Review

A hyperquadric is a surface defined by the set of points (x, y, z) satisfying the hyperquadric equation:

$$\sum_{i=1}^{N} | A_i x + B_i y + C_i z + D_i |^{\gamma_i} = 1 \qquad (1)$$

Here N is any arbitrary number and $\gamma_i \geq 0 \quad \forall i$. We note that a superquadric as defined by

$$| \frac{x}{a} |^{\gamma_1} + | \frac{y}{b} |^{\gamma_2} + | \frac{z}{c} |^{\gamma_3} = 1 \qquad (2)$$

is a special case of the hyperquadric with $N = 3$, $A_1 = \frac{1}{a}$, $B_2 = \frac{1}{b}$, $C_3 = \frac{1}{c}$ and all other parameters (except γ_i) zero. Whereas the superquadric has only 3 terms the hyperquadric has an arbitrary number of terms. The hyperquadric can model a much broader range of shapes than the superquadric. Due to the absolute values in its equation, the superquadric is symmetric about the three axes and hence, without deformations, can model only symmetric objects. Hyperquadric models need not be symmetric.

*Supported in part by the NSF(IRI-90-10357), the Whitaker Foundation and Siemens

Consider the hyperquadric equation. Each term in the summation is positive and all the terms add up to one. Therefore, each of the individual terms can never exceed one. *i.e.*, $\mid A_i x + B_i y + C_i z + D_i \mid^{\gamma_i} \leq 1 \quad \forall i$. Since γ_i's are positive, this inequality implies that

$$\mid A_i x + B_i y + C_i z + D_i \mid \leq 1 \quad \forall i = 1, 2, \ldots, N \quad (3)$$

The above equation describes a pair of parallel planes for each i. The intersection of all these $2N$ planes defines a convex polytope and the hyperquadric is constrained to lie within this polytope. More complex bounding volumes can be defined by using more complex expressions within each term [8].

To fit a hyperquadric model to range data we first need an error-of-fit (EOF) function to measure the difference between the modeled shape and the given data. Once we have an EOF function, the fitting process amounts the finding the set of parameters that would minimize the function. In [6], an EOF is defined by

$$EOF = \sum_{i=1}^{Ndata} \frac{1}{\mid\mid \nabla F_{io}(x, y, z) \mid\mid^2} (1 - F_{io}(x, y, z))^2 \quad (4)$$

where

$$F_{io}(x, y, z) = f_{io}(x, y, z)^p \quad (5)$$

and

$$f_{io}(x, y, z) = \sum_{i=1}^{3} \mid A_i x + B_i y + C_i z + D \mid^{\gamma_i}. \quad (6)$$

For points close to the hyperquadric, EOF is a first order approximation to the true Euclidean distance.

The above EOF is minimized[1] using the Levenberg-Marquardt optimization method[10] with an ellipsoid as an initial guess. Once the algorithm converges, the fit can be improved by introducing more terms into the hyperquadric equation followed by further minimization. New terms are introduced by splitting an existing term into two.

While the above algorithm runs well on some data, it has some serious problems with convergence. Specifically, consider the hyperquadric equation. If we set $A_i = B_i = C_i = 0 \ \forall i$, the equation can still be satisfied by infinitely many choices of D_i and γ_i irrespective of the value of (x, y, z). Should we stumble upon such a set of parameters, the error-of-fit would be zero for all (x, y, z). Further, such a set of parameters does not correspond to any meaningful shape. In other words, the EOF function has an infinite number of global minima with zero error and only some of them correspond to the shape being modeled. This is in addition to the existence of many local minima that might produce shapes very different from the desired shape. With the algorithm given in [6], even when we start with a good initialization, we may end up in a meaningless global minimum. Clearly, we need some restrictions on the range of values the parameters can take. The next section discusses constraints that will force the final solution to be close to the correct solution.

[1] We omit some minor details here. Please refer to [6] for a complete discussion.

3 Adding Constraints

The number of constraints that can be imposed depends on the amount of *a priori* knowledge we have about the object being modeled. In this paper, we assume no *a priori* knowledge. Our only information about the object is in the form of a range image. From this information, we can extract the extent of the object. We can also estimate approximately the maximum width of the object. We use this information to produce four constraints. These four constraints will assure convergence to a point close to the correct minimum.

Each term of the hyperquadric equation corresponds to the pair of planes:

$$A_i x + B_i y + C_i z + D_i = \pm 1 \quad (7)$$

The model is constrained to lie within the region bounded by these planes. The distance between the planes (or the "size" of the above region) is $S_i = \frac{2}{\sqrt{A_i^2 + B_i^2 + C_i^2}}$. The pathological case $A_i = B_i = C_i = 0, \ \forall i$, occurs when the distance between the planes is infinity. Clearly, this can be avoided by setting an upper limit on the size S_i. Another problem arises when the planes get too close each other such that the volume bounded by the planes is very small and lies inside the object. If such a situation were to correspond to a local minimum the original algorithm would readily converge to that set of parameters. In such a case even if the γ's are high (which means that the shape of the object is the shape of the bounding box), the resulting shape is far from the desired shape. This brings us to the second constraint: the distances ($S_i \ \forall i$) between the bounding planes should be greater than the maximum diameter of the object. We therefore, set the lower limit on S_i to be S_{min} where S_{min} is slightly larger than the maximum diameter of the object. We set the upper limit on S_i to be S_{max}, where S_{max} is chosen arbitrarily as $1.5 \times S_{min}$. Thus we have $\forall i = 1, 2, \cdots N$,

$$S_{min} \leq S_i \leq S_{max} \quad (8)$$

$$\Rightarrow S_{min} \leq \frac{2}{\sqrt{A_i^2 + B_i^2 + C_i^2}} \leq S_{max} \quad (9)$$

$$\Rightarrow \mu_1 \leq A_i^2 + B_i^2 + C_i^2 \leq \mu_2 \quad (10)$$

where $\mu_1 = \left(\frac{2}{S_{max}}\right)^2$ *and* $\mu_2 = \left(\frac{2}{S_{min}}\right)^2$.

We need some restrictions on the γ's too. When γ_i is very high, the minimization process has a tendency

to increase γ_i more and more. Also when the γ's become very small, the shape becomes too "pinched". We therefore restrict γ_i to lie in the interval $[0.3, 30]$. This gives us two more constraints:

$$\gamma_i \leq \gamma_{max} \quad \forall i = 1, 2, \cdots N \tag{11}$$

$$\gamma_i \geq \gamma_{min} \quad \forall i = 1, 2, \cdots N \tag{12}$$

where $\gamma_{max} = 30$ and $\gamma_{min} = 0.3$.

Thus we have a total of four constraints, two on S_i and two on γ_i for each term. We use the penalty method to incorporate these inequality constraints into the minimization process.

The penalty method[9] replaces the problem

$$minimize \quad f(\mathbf{x}) \quad subject\ to \quad \mathbf{x} \in S \tag{13}$$

where f is a continuous function on E^n and S is a constraint region by an unconstrained problem of the form

$$minimize \quad f(\mathbf{x}) + cP(\mathbf{x}) \tag{14}$$

where c is a positive constant and P is a penalty function on E^n such that (1) P is continuous, (2) $P(\mathbf{x}) \geq 0 \ \ \forall \mathbf{x} \in E^n$ and (3) $P(\mathbf{x}) = 0$ if and only if $\mathbf{x} \in S$.

Clearly, for large values of c, the minimum of the above problem will lie in a region where P is small. We expect that as c increases, the corresponding solution point will approach the feasible region S and will minimize f. In other words, we expect the solution point of problem (14) will converge to the solution of problem (13) as $c \rightarrow \infty$.

The penalty method asks us to solve problem (14) with increasing values of c. Let $\langle \mathbf{x_k} \rangle$ denote the sequence of solutions obtained. Then any limit point of the sequence is a solution to the original problem (13).

With hyperquadrics, we found experimentally that it is sufficient to solve problem (14) for a single large value of c. We set c to 100 and solve problem (14). Its solution satisfies problem (13).

How do we find a penalty function satisfying the conditions stated above? Suppose we have a single inequality constraint $g(\mathbf{x}) \leq 0$. (Note that all our constraints can be written in this form). A penalty function that satisfies the stated conditions is

$$P(\mathbf{x}) = (max\,[0, g_i(\mathbf{x})])^2 \tag{15}$$

Thus, we can rewrite our original constrained minimization problem as an unconstrained minimization problem as follows. We have four constraints per term - a total of $4N$ constraints. The penalty function P_i for each term becomes

$$\begin{aligned} P_i &= (max[0, A_i^2 + B_i^2 + C_i^2 - \mu_2])^2 \\ &+ (max[0, \mu_1 - A_i^2 - B_i^2 - C_i^2])^2 \\ &+ (max[0, \gamma_i - \gamma_{max}])^2 + (max[0, \gamma_{min} - \gamma_i])^2 \end{aligned} \tag{16}$$

And the problem becomes: $Minimize$

$$\begin{aligned} &\sum_{i=1}^{Ndata} \frac{1}{|| \nabla F_{io}(x, y, z) ||^2} (1 - F_{io}(x, y, z))^2 \\ &+ c \sum_{i=1}^{N} P_i \end{aligned} \tag{17}$$

4 The Algorithm

First we need to determine the initial values for the parameters. Due to the fact that each term of the hyperquadric corresponds to two parallel lines, the resulting shape still exhibits some symmetry. Every segment of the curve has a "parallel" segment somewhere else on the curve. The problem is due of the fact that if a line affects the curve in a particular way, its "partner" affects the opposite side of the curve in the same way. As Hanson [8] has suggested, the problem can be minimized by moving one of the partners far enough away so that its effect on the curve is negligible. Such a partner is called "hidden partner". We therefore choose the minimum distance S_{min} (equation 8) between the planes to be about 1.5 times the maximum diameter of the object. This places the parallel planes (in 3D) sufficiently far form each other so that only one of them influences the hyperquadric surface.

In the method proposed in [6], the authors started with 3 terms in 3D and 2 terms in 2D. Additional terms were created by splitting as and when necessary. Every additional term requires splitting all the old terms and choosing the term that minimizes the error. We noticed that every non-trivial shape in 2D requires at least 4 terms (at least 6 terms in 3D). Therefore, we start our algorithm with 4 terms in 2D and 6 terms in 3D. This eliminates some of the search and split in the beginning. The 2D case is illustrated in Figure 1(a). The data points are shown as dots. There are 4 lines corresponding to the 4 terms. The partners of the visible lines are moved far away and are not visible in the figure. These lines are perpendicular to the x and y axes and are placed at a short distance to the left/right and top/bottom of the object. All the γ's are initialized to 4. The initial curve is also shown. The final model along with the 4 lines corresponding to the four terms is shown in Figure 1(b).

With the above initialization, the algorithm is:

1. For each data point (x, y, z) compute the weight $\frac{1}{||\nabla F_{io}(x,y,z)||^2}$.
2. Minimize the function given in Equation 17 using the above weights..
3. If the fitting is good, stop. Else, return to step 1.

If necessary, additional terms are added by splitting.

Figure 2 illustrates the need for constraints. Figure 2(a) shows the data and the initial contour. Figure 2(b)

shows the result of constrained minimization. The final hyperquadric has 4 terms (for the third term, both the bounding lines are visible). Figures 2(c) and 2(d) show the path taken by the algorithm when we minimize the EOF without any constraints. We see that the bounding lines are moving farther and farther apart and the error-of-fit s decreasing. Eventually we reach a stage where the parameters A_i and B_i are very close to zero and any value of (x, y, z) gives negligible error.

Due to the use of constraints, the algorithm is somewhat insensitive to initialization. This does not mean that we can use any arbitrary initialization. For example, if the initialization is such that the region bounded by the planes is completely away and to one side of the object, the algorithm might not converge.

5 Experimental Results and Discussions

The duck, spoon and hand images (Figures 3, 4 and 6) were manually segmented and hyperquadrics were fitted to the individual parts. Figure 5 shows 2D and 3D hyperquadric fits to left ventricular data. The ventricular data, the spoon head (Figure 4(c)) and the duck's body (Figure 3(c)) describe some shapes that cannot be modeled by superquadrics with simple deformations. The techniques presented in this paper are part of our on-going research on utilizing hyperquadrics for modeling, identification and motion analysis of multicomponent nonrigid objects. Clearly, hyperquadrics have more representational power than superquadrics and hence more suitable for object modeling. However, there are a number steps to be taken before we can use hyperquadrics for modeling complex multicomponent objects.

References

[1] A. H. Barr. Superquadrics and Angle-Preserving Transformations. *Computer Graphics and Applications*, 1(1):11–23, 1981.

[2] A. H. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18(3), 1984.

[3] T. E. Boult and A. D. Gross. Recovery of Superquadrics from 3-D Information. In *SPIE Intelligent Robots and Computer Vision:Sixth in a Series* , volume 848. 1987.

[4] F. P. Ferrie, J. Lagarde, and P. White. Darboux Frames, Snakes and Super-Quadrics. *IEEE Trans. on PAMI*, 15(8):771–784, 1993.

[5] A. Gupta. *Surface and Volumetric Segmentation of Complex 3-D Objects Using Parametric Shape Models.* PhD thesis, University of Pennsylvania, 1991. Department of Computer and Information Science.

[6] S. Han and D. Goldgof. Using Hyperquadrics for Shape Recovery form Range Data. In *Proceedings of the fourth ICCV, Berlin*, pages 492–496. 1993.

[7] A. Hanson. Hyperquadrics: Smoothly Deformable Shapes with Convex Polyhedral Bounds. *CVGIP*, 44:191–210, 1988.

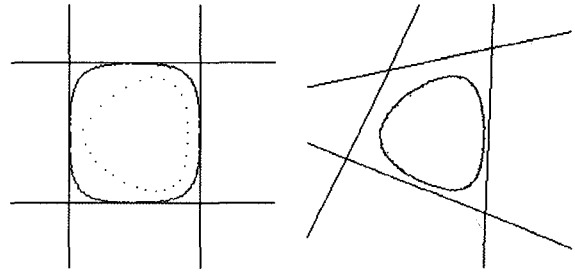
[8] A. Hanson. Implicit Functions for Modeling Arbitrary Deformable Shapes. In *Technical Report, Computer Science Department, Indiana University.* 1990.

[9] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing, 1984.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C, Second Edition.* Cambridge University Press, 1992.

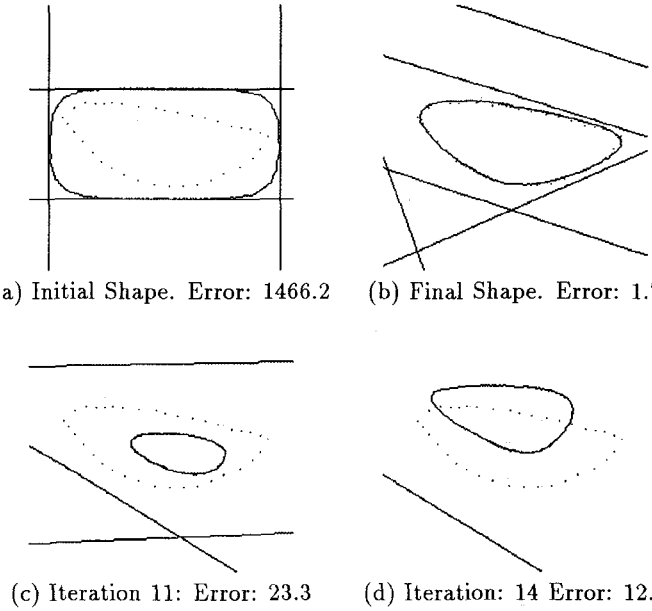
[11] N. S. Raja and A. K. Jain. Recognizing Geons from Superquadrics Fitted to Range Data. *Imaging and Vision Computing*, pages 179–190, 1992.

[12] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE Trans. on PAMI*, 12(2):131–147, 1990.



(a)Initial Shape (b) Final Shape

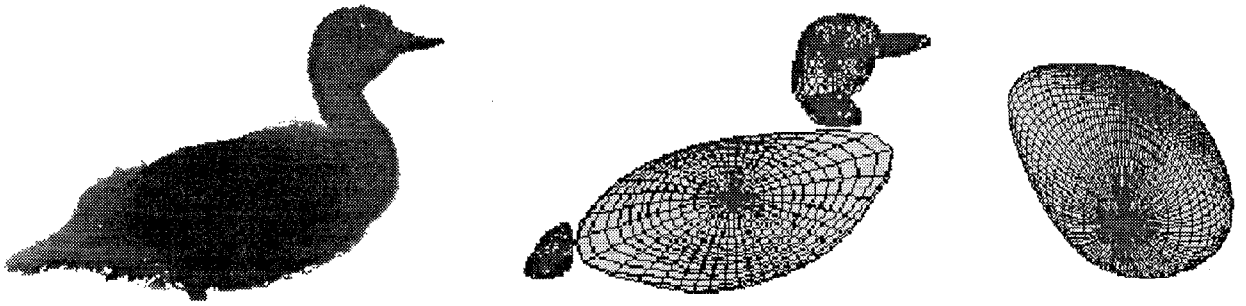
Figure 1: 2D Estimation: Initialization and the final shape



(a) Initial Shape. Error: 1466.2 (b) Final Shape. Error: 1.7

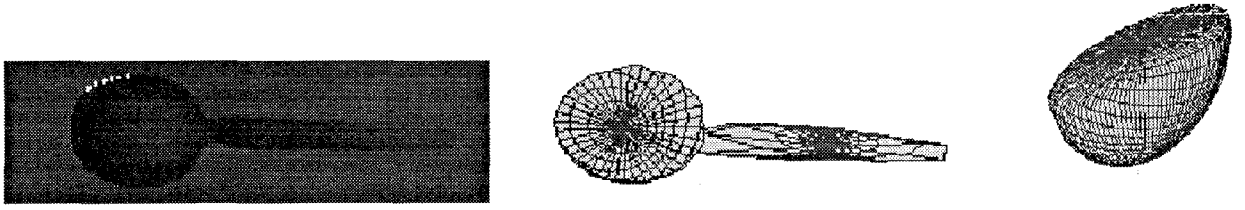
(c) Iteration 11: Error: 23.3 (d) Iteration: 14 Error: 12.5

Figure 2: Constrained and unconstrained minimization.



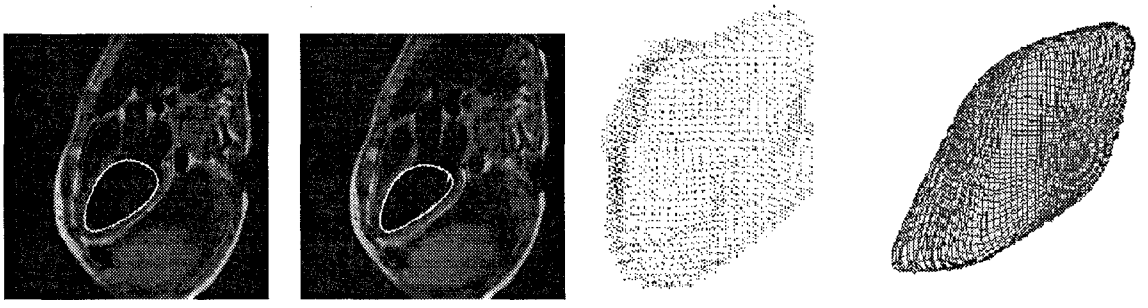
(a) (b) (c)

Figure 3: (a) Original range data (b) HQ fit (6 terms/part) (c) A different view of duck's body.



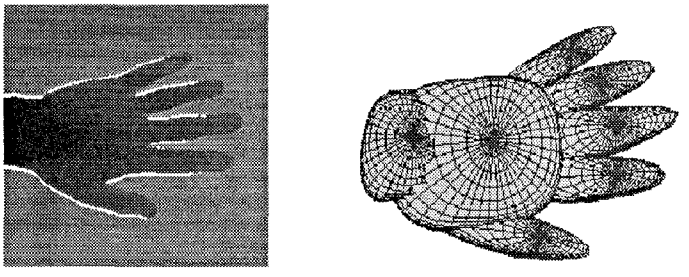
(a) (b) (c)

Figure 4: (a) Original Range Data (b) HQ fit (6 terms/part) (c) Another view of the spoon head.



(a) (b) (c) (d)

Figure 5: (a &b) 2D fits to ventricular boundaries. (c) Original Volumetric Data (d) HQ fit (6 terms).



(a) Original range data (b) Hyperquadric fit (6 terms).

Figure 6: HQ fit to the range image of a hand (6 terms/part).