Correspondence

On Recovering Hyperquadrics from Range Data

Senthil Kumar, Song Han, Dmitry Goldgof, and Kevin Bowyer

Abstract—**This paper discusses the applications of hyperquadric models in computer vision and focuses on their recovery from range data. Hyperquadrics are volumetric shape models that include superquadrics as a special case. A hyperquadric model can be composed of any number of terms and its geometric bound is an arbitrary convex polytope. Thus, hyperquadrics can model more complex shapes than superquadrics. Hyperquadrics also possess many other advantageous properties (compactness, semilocal control, and intuitive meaning). 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. Our proposed algorithm starts with a rough fit using only six terms in 3D (four in 2D) and adds additional terms as necessary to improve fitting. Suitable constraints are used to ensure proper convergence. Experimental results with real 2D and 3D data are presented.**

Index Terms—**Object modeling, hyperquadrics, surface fitting, object representation, volumetric models.**

I. Introduction

Several researchers have used superquadrics for object modeling and motion analysis [1], [2], [4], [5], [6], [7], [14], [15], [17], [19]. Raja and Jain [17] explore the use superquadrics for modeling a subset of geons [3]. They use superquadrics to model and successfully discriminate 12 of the 36 geon classes. Their experiments indicate that qualitative shape attributes can be reliably inferred from superquadric models and hence superquadrics can be quite useful in object recognition.

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. Solina and Bajscy [19] 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 is 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 (see for example, the ventricular shapes (2D and 3D) and the body of the duck shown in Fig. 6 and Fig. 8c). Also, a large subclass of geons, i.e. geons with asymmetric cross-sections, cannot be modeled by simple deformable superquadrics.

Hyperquadrics were first proposed for computer graphics applications by Hanson [9]. Hyperquadrics can model shapes that are not necessarily symmetric. Their geometric bounds are arbitrary convex polyhedra, as opposed to superquadrics whose geometric bounds are simple Cartesian rectangular parallelopipeds. This paper proposes using hyperquadric models for shape recovery from range data. We define an error-of-fit function that is minimized using Levenberg-Marquardt optimization [16]. The hyperquadric equation can contain an arbitrary number of terms. We start with four terms in 2D (six terms in 3D) and then add additional terms to improve the fitting, if necessary. Each additional term is introduced by splitting an appropriate term into two separate terms. This procedure is repeated until we reach the desired level of precision. As explained in Section IV, the error-of-fit function has multiple global minima with zero error, most of which do not correspond to any meaningful shape. We introduce constraints in the minimization process to ensure proper convergence.

The rest of the paper is organized as follows. Hyperquadrics and their properties are defined in Section II. In Section III.A, we present two error-of-fit functions utilized for the minimization. In Section III.B, we discuss how to obtain a good initial guess and in Section III.C, we describe how to add more terms to improve the fitting. In Section IV, we discuss the constraints that are needed to ensure proper convergence. In Section V, we give the experimental results of shape recovery from 2D and 3D synthetic and real data.

Due to space constraints, the size of this paper has been cut down considerably. The original version is available on WWW [11]. Some preliminary research was presented in [8], [12].

II. Hyperquadric Models

A. The Hyperquadric Equation

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

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

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

$$\left|\frac{x}{a}\right|^{\gamma_1} + \left|\frac{y}{b}\right|^{\gamma_2} + \left|\frac{z}{c}\right|^{\gamma_3} = 1 \tag{2}$$

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 three 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.

Consider the hyperquadric equation. Each term in the summation is positive and the terms add to one. Therefore, no individual term can ever exceed one. Since the γ_is are positive, this implies that

$$|A_i x + B_i y + C_i z + D_i| \leq 1\ \forall i = 1, 2, \ldots, N \tag{3}$$

The above equation describes a pair of parallel planes for each i. These planes define 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 [10]. For example, if the planar expression $A_i x + B_i y + C_i z + D_i$ is replaced with an expression for a sphere, the resulting bounding volume would be the intersection of a set of spheres.

In order to get a geometric feeling for the effect of the various parameters on the resulting shape, let us consider the 2D hyperquadric[1]:

$$\sum_{i=1}^{N} |A_i x + B_i y + D_i|^{\gamma_i} = 1 \tag{4}$$

Manuscript received Feb. 18, 1993; revised Mar. 28, 1995. Recommended for acceptance by T.E. Boult.

The authors are with the Department of Computer Science and Engineering, University of South Florida, Tampa FL 33620 U.S.A.; email: kumar@figment.csee.usf.edu, goldgof@gasparilla.csee.usf.edu.

IEEECS Log Number P95093.

1. We use 2D hyperquadrics for illustration purposes.

Instead of bounding planes, we now have bounding lines. Note that each term gives a pair of parallel lines. The parameters A_i and B_i in term i give the slope of the lines corresponding to that term. The distance S_i between the lines is given by $\frac{2}{\sqrt{A_i^2+B_i^2}}$. One line is at a distance $\frac{D_i-1}{\sqrt{A_i^2+B_i^2}}$ from the origin and the other is at a distance $\frac{D_i+1}{\sqrt{A_i^2+B_i^2}}$. The distance T_i between the origin and the center-plane between the two bounding planes is given by $T_i = \frac{D_i}{2S_i}$. Fig. 1 shows three hyperquadrics, each with two terms. The bounding lines are also shown. If a particular value of γ is high, the shape is pulled toward the corresponding pair of bounding lines. When all the γs are 2, the resulting shape is an ellipse. When the γs are 1, the shape is a rectangle and when a γ becomes less than one, the shape becomes concave. Fig. 2 shows some 3D hyperquadrics.

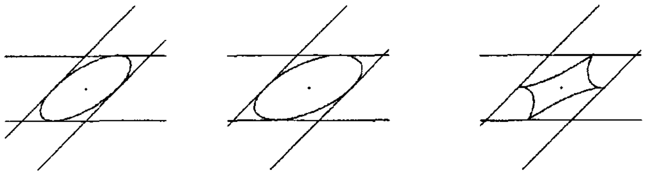
(a)$\gamma_1 = \gamma_2 = 2$
(b)$\gamma_1 = \gamma_2 = 2$; Lines farther apart
(c)$\gamma_1 = 0.9$; $\gamma_2 = 0.5$

Fig. 1. 2D Hyperquadrics: Inclined lines correspond to term 1; horizontal lines to term 2.

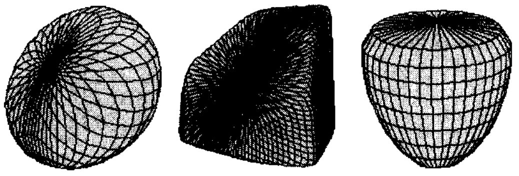

Fig. 2. 3D Hyperquadrics (four terms each).

The parameters A_i and B_i determine the slopes of the bounding lines and also the distance between them. By moving a pair of bounding lines farther apart while keeping everything else the same, we can increase the width of the curve in a direction perpendicular to that pair. The parameter D_i, along with A_i and B_i, determines the distances of the two lines from the origin and the parameters γ_i, $i = 1, 2, \cdots, N$ determine the "squareness" of the shape.

Since the hyperquadric equation can have any number of terms, the first several terms can be used to capture the global shape and more and more terms can be added to define small details. In the hyperquadric equation, terms with smaller size S_i and smaller exponents γ_i have greater effect on the shape. Because of the absolute value sign, each term will affect the shape on both sides. In order to adjust the shape in only one side, the term's size S_i should be big enough and the translation T_i is adjusted to move one bounding plane close to the shape and the other bounding plane far away from the shape.

Hyperquadrics also possess "semilocal control." When adjusting a term in hyperquadrics, a significant change in shape occurs only in the direction specified by this term. The semilocal property of hyperquadrics is due to the fact that when we adjust one term to push in or pull out the bounding plane or change their directions, the other $N-1$ bounding planes will not move at all, and thus the shape does not change significantly in the directions specified by those $N-1$ terms.

Consider the function $f_{io}(x, y, z)$ defined by

$$f_{io}(x, y, z) = \sum_{i=1}^{N} \left|A_i x + B_i y + C_i z + D_i\right|^{\gamma_i} \tag{5}$$

Note that from (1), if a point (x, y, z) lies on the hyperquadric then $f_{io}(x, y, z) = 1$, if the point lies inside the hyperquadric, $f_{io}(x, y, z) < 1$ and if the point lies outside the hyperquadric, $f_{io}(x, y, z) > 1$. Due to the exponent in each term of $f_{io}(x, y, z)$, the function changes too quickly when the point (x, y, z) moves. A small exponent p can be added to the function to obtain an approximately uniform function:

$$f_{io}(x, y, z)^p = 1 \tag{6}$$

where we usually take $p = 1/5$, since the exponents in f_{io} are usually around 5. The above equation is rewritten as

$$F_{io}(x, y, z) = 1 \tag{7}$$

and we refer to F_{io} as the inside-outside function of the hyperquadric. The inside-outside function gives us a measure of the relative position of a data point with respect to the model (i.e., the surface) and it can be used as an error-of-fit measure in the shape recovery procedure.

III. Fitting Hyperquadrics to Range Data

A. Error-of-Fit Functions and Minimization Method

To fit a hyperquadric to range data, we first define an error-of-fit function to measure the difference between a modeled shape and the given data set. Then we define a procedure to minimize the error-of-fit. For generic hyperquadrics, there is no closed-form error-of-fit function based on the true Euclidean distance. However, we can easily define our first error-of-fit function based on the inside-outside function:

$$EOF_1 = \sum_{i=1}^{Ndata} \left(1 - F_{io}(x_i, y_i, z_i)\right)^2 \tag{8}$$

This function is biased, especially for oblong objects. So we define a better measure similar to those defined in [18], [20]. For a data point (x_i, y_i, z_i) on the isosurface

$$F_{io}(x, y, z) = w_i \tag{9}$$

where $w_i = F_{io}(x_i, y_i, z_i)$, the orthogonal distance can be approximated by the distance d_i along the direction of the gradient. Along this direction, the corresponding point on the hyperquadric is (x, y, z), and we have

$$F_{io}(x, y, z) \approx F_{io}(x_i, y_i, z_i) + d_i \left\|\nabla F_{io}(x_i, y_i, z_i)\right\|. \tag{10}$$

Thus

$$d_i \approx \frac{1 - F_{io}(x_i, y_i, z_i)}{\left\|\nabla F_{io}(x_i, y_i, z_i)\right\|} \tag{11}$$

so the squared error over the whole set of data points is defined as

$$EOF_2 = \sum_{i=1}^{Ndata} \frac{1}{\left\|\nabla F_{io}(x_i, y_i, z_i)\right\|^2} \left(1 - F_{io}(x_i, y_i, z_i)\right)^2 \tag{12}$$

EOF_2 can be regarded as a weighted version of EOF_1, where the weight is introduced to alleviate the bias in EOF_1.

Obviously, directly implementing the minimization based on EOF_2 will be computationally too expensive, so an iterative algorithm is used:

1) minimize EOF_1 and obtain F_{io};
2) compute the weight for each data point: $w_i = \frac{1}{\|\nabla F_{io}\|^2}$
3) use the weight w_i computed in 2) to minimize EOF_2 and obtain F_{io}
4) if the fitting F_{io} is not good enough, go to 2); else, stop.

In the above procedure, the minimization is accomplished by the Levenberg-Marquardt non-linear optimization method [16].

B. Initial Guess

The initial guess is chosen to be an ellipsoid in 3D and an ellipse in 2D. We start with six terms in 3D and four terms in 2D. The initial values of these parameters are chosen as follows. Refer to Fig. 3 which shows a 2D hyperquadric with three terms. Due to the fact that each term of the hyperquadric corresponds to two parallel lines and because of the way in which these lines are arranged in the figure, the resulting shape is inherently symmetric. Every segment of the curve has a "parallel" segment somewhere else on the curve. The problem is because 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 [10] has suggested, the problem can be minimized by moving one of the partners far away so that its effect on the curve is negligible. Such a partner is called a "hidden partner." We therefore choose the minimum distance S_{min} between the planes to be about 1.5 times the maximum diameter of the object. This places the parallel planes (in 3D) sufficiently far from each other so that only one of them influences the hyperquadric surface.

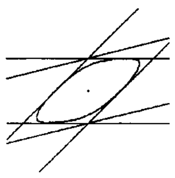
Fig. 3: 2D HQ with three terms.

The 2D initialization is illustrated in Fig. 4a. The data points are shown as dots. There are four lines corresponding to the four terms. The partners of the visible lines are moved far away and are not visible in the figure. Two lines are perpendicular to the x-axis and are placed to the left and right of the object. The other two lines are oriented similarly in the perpendicular direction. All the γs are initialized to 2. The initial curve is also shown. The final model along with the four lines corresponding to the four terms is shown in Fig. 4b. 3D initialization is similar.

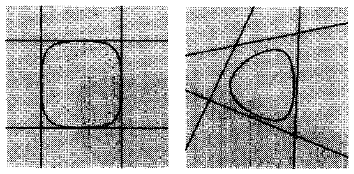
(a) Initial shape (b) Final shape

Fig. 4: 2D estimation.

C. Fitting More Terms

Unlike superquadrics, hyperquadrics can have an arbitrary number of terms in equation. During minimization, if the current fitting is not good enough, additional terms can be added to refine the fit. Due to space constraints, we have removed the discussion on adding new terms during the fitting process. Please refer to [11] for details.

IV. Constraints

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. Thus, 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. Clearly, we need some restrictions on the range of values the parameters can take. 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 in the x, y, and z directions. We can also estimate the minimum width of the object. We use this information to produce four constraints. These four constraints will assure a convergence that is close to the correct minimum.

As we mentioned earlier, each term $|A_i x + B_i y + C_i z + D_i|^{\gamma_i}$ corresponds to the pair of planes: $A_i x + B_i y + C_i z + D_i = +1$ and $A_i x + B_i y + C_i z + D_i = -1$. 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 to each other and enclose a small volume 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 γs 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 minimum diameter of the object. We therefore set the lower limit on S_i, S_{min}, to be slightly larger than the minimum diameter of the object. We set the upper limit on S_i, S_{max}, is chosen arbitrarily as 1.5 times the (estimated) maximum diameter of the object. Thus we have $\forall i = 1, 2, \cdots N$,

$$S_{min} \le S_i \le S_{max} \tag{13}$$

$$\Rightarrow S_{min} \le \frac{2}{\sqrt{A_i^2 + B_i^2 + C_i^2}} \le S_{max} \tag{14}$$

$$\Rightarrow \mu_1 \le A_i^2 + B_i^2 + C_i^2 \le \mu_2 \tag{15}$$

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 γs too. When γ_i is very high, it has a tendency to increase more and more. Also when the γs 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:

$$\gamma_i \ge \gamma_{min}\ \forall i = 1, 2, \cdots N \text{ and } \gamma_i \le \gamma_{max}\ \forall i = 1, 2, \cdots N \tag{16}$$

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 incorporate these inequality constraints into the minimization process using the penalty method [13].

Consider the minimization problem

$$\textit{minimize}\ f(\mathbf{x})\ \textit{subject to}\ \mathbf{x} \in S \tag{17}$$

where f is a continuous function on E^n and S is a constraint region. The penalty method replaces the above problem by an unconstrained problem of the form

$$\textit{minimize}\ f(\mathbf{x}) + cP(\mathbf{x}) \tag{18}$$

where c is a positive constant and P is a penalty function on E^n such that

1) P is continuous,
2) $P(\mathbf{x}) \ge 0\ \forall \mathbf{x} \in E^n$, 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 (18) will converge to the solution of (17) as $c \to \infty$ [13]. The penalty method asks us to solve (18) 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 (17).

With hyperquadrics, we found experimentally that it is sufficient to solve (18) for a single large value of c. We set c to 100 and solve (18). Its solution satisfies (17). The above value of 100 was selected arbitrarily and was experimentally found to yield good results.

Given a single inequality constraint $g(\mathbf{x}) \le 0$, a penalty function that satisfies the stated conditions is[2] $P(\mathbf{x}) = (max[0, g_i(\mathbf{x})])^2$. We can thus 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

$$P_i = \left(max\left[0, A_i^2 + B_i^2 + C_i^2 - \mu_2\right]\right)^2 + \left(max\left[0, \mu_1 - A_i^2 - B_i^2 - C_i^2\right]\right)^2 + \left(max\left[0, \gamma_i - \gamma_{max}\right]\right)^2 + \left(max\left[0, \gamma_{min} - \gamma_i\right]\right)^2 \tag{19}$$

And the problem becomes

$$minimize \sum_{i=1}^{Ndata} \frac{1}{\left\|\nabla F_{io}(x, y, z)\right\|^2}\left(1 - F_{io}(x, y, z)\right)^2 + c\sum_{i=1}^{N} P_i \tag{20}$$

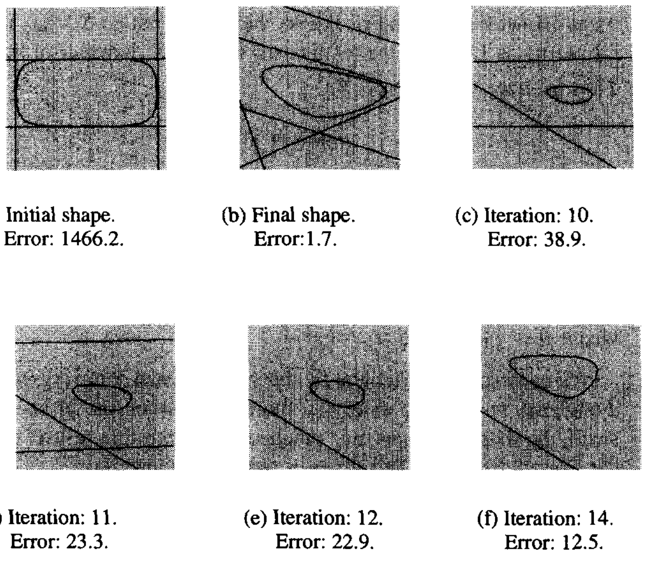

Fig. 5. Constrained and unconstrained minimization. Figure (b) is the result of constrained minimization. Figures (c) through (f) show the path of unconstrained minimization.

Fig. 5 illustrates the need for constraints. Fig. 5a shows the data and the initial contour (data points are shown as dots and the hyperquadric as a solid curve). Fig. 5b shows the result of constrained minimization. The final hyperquadric has four terms (for the third term, both the bounding lines are visible). Figs. 5c through 5f 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 keeps 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 will not converge.

V. Experimental Results and Conclusions

In this section, we present our experimental results. In Figs. 6 a and b, the left-ventricular boundaries were extracted manually and then hyperquadrics were fit to the extracted data. Figs. 6c and d show the fit to a volumetric data of the left ventricle. The duck, spoon, and hand images (Figs. 7, 9, and 10) were segmented manually and hyperquadrics were fitted to the individual parts. Figs. 6, 8c, and 9c show some shapes that cannot be modeled by superquadrics with simple deformations.

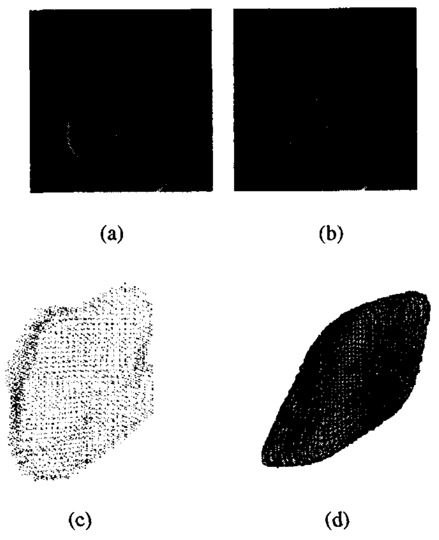

Fig. 6. (a and b) 2D fits to ventricular boundaries, (c) original volumetric data, (d) HQ fit (six terms).

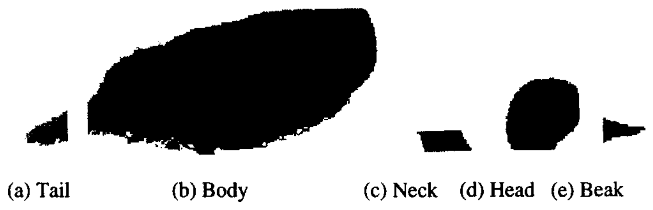

Fig. 7. Manually segmented duck range data.

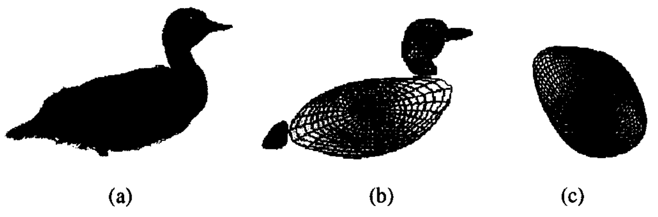

Fig. 8. (a) Original range data, (b) HQ fit (six terms per part), (c) a different view of duck's body. The duck was manually segmented into five parts as shown in the previous figure.

2. Note that all our constraints can be written in this form.

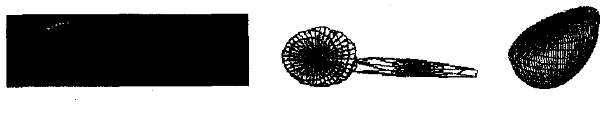

Fig. 9. (a) Original range data, (b) HQ fit (six terms per part), (c) another view of the spoon head. The spoon was manually segmented into two parts: handle and head.

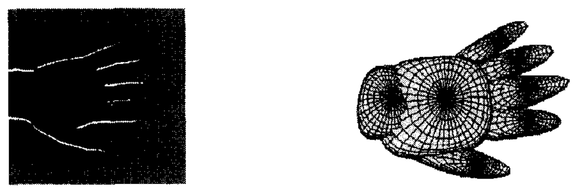

Fig. 10. Hyperquadric fit to the range image of a hand (six terms per part). The data was manually segmented into seven parts: wrist, palm, and five fingers.

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 are more suitable for object modeling. However, there are a number of steps to be taken before we can use hyperquadrics for modeling complex objects. Several researchers have reported successful results in modeling complex objects using superquadrics (see, for example, Ferrie, Legarde and White [6] and Gupta [7]) and it might be possible to extend their approaches to hyperquadrics. Simple concave objects can be modeled by hyperquadrics as long as their curvature is small, but large concavities present problems. One way to model concave objects is to use bending (see Solina [19] and Barr [2]).

The fact that hyperquadrics can model a diverse range of shapes with a small number of parameters makes them attractive for the purpose of recognition. However, much work needs to be done. The hyperquadric parameters of a given shape are not unique. Also many simple shapes have infinite sets of parameters. For the sphere, for instance, any hyperquadric equation with two terms that give a cubical bounding box just enclosing the sphere would be an exact model as long as the γ_is are 2. Leaving such pathological cases aside, we want to determine some way of representing complex objects in a unique way. Clearly, we need to add constraints on the possible values of the parameters but exactly what constraints must be imposed is not clear. Our final goal is to have a set of techniques that use hyperquadrics efficiently for modeling, nonrigid motion analysis and recognition.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grant IRI-90-100357 and in part by a grant from Siemens.

REFERENCES

[1] A.H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer Graphics and Applications*, vol. 1, no. 1, pp. 11-23, 1981.
[2] A.H. Barr, "Global and local defromations of solid primitives," *Computer Graphics*, vol. 18, no. 3, 1984.
[3] I. Biederman, "Recognition-by components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 1, pp. 115-147, 1987.
[4] T.E. Boult and A.D. Gross, "Recovery of superquadrics from 3D information," *SPIE Intelligent Robots and Computer Vision: Sixth in a Series*, vol. 848, 1987.
[5] C.W. Chen, T.S. Huang, and M. Arrott, "Modeling, analysis, and visualization of left ventrical shape and motion by hierarchical decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 342-356, Apr. 1994.
[6] F.P. Ferrie, J. Lagarde, and P. White, "Darboux frames, snakes, and superquadrics," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 8, pp. 771-784, 1993.
[7] A. Gupta, "Surface and volumetric segmentation of complex 3D objects using parametric shape models," PhD thesis, Dept. of Computer and Information Science, Univ. of Pennsylvania, 1991.
[8] S. Han, D. Goldgof, and K. Bowyer, "Using hyperquadrics for shape recovery from range data," *Proc. Fourth ICCV*, pp. 492-296, Berlin, 1993.
[9] A. Hanson, "Hyperquadrics: Smoothly deformable shapes with convex polyhedral bounds," *CVGIP*, vol. 44, pp. 191-210, 1988.
[10] A. Hanson, "Implicit functions for modeling arbitrary deformable shapes," technical report, Computer Science Dept., Indiana Univ., 1990.
[11] S. Kumar and D. Goldgof, "On recovering hyperquadrics from range data," www:http://marathon.csee.usf.edu.
[12] S. Kumar and D. Goldgof, "A robust technique for the estimation of hyperquadrics parameters from range data," *Proc. 12th ICPR*, vol. I, pp. 74-78, Jerusalem, 1994.
[13] D.G. Luenberger, *Linear and Nonlinear Programming*. Reading, Mass.: Addison-Wesley, 1984.
[14] A.P. Pentland, "Perceptual organization and the representation of natural form," *Artificial Intelligence*, pp. 293-331, 1986.
[15] A.P. Pentland, "Recognition by parts," *Proc. First ICCV*, pp. 612-620, United Kingdom, 1987.
[16] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C*. Cambridge, England: Cambridge Univ. Press, 1992.
[17] N.S. Raja and A.K. Jain, "Recognizing geons from superquadrics fitted to range data," *Imaging and Vision Computing*, pp. 179-190, 1992.
[18] P.D. Sampson, "Fitting conics to very scattered data: An iterative refinement of Bookstein algorithm," *CVGIP*, vol. 18, pp. 97-108, 1982.
[19] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case for superquadrics with global deformations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 131-147, 1990.
[20] G. Taubin, "Estimation of planar curves, surfaces, and nonplanar curves defined by implicit equations with applications to edge and range image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, 1991.