

# **PROGRESS IN ROBOTICS AND INTELLIGENT SYSTEMS**

## **VOLUME 2**

**GEORGE W. ZOBRIST  
C.Y. HO**  
Editors

Tp242-6  
p964  
v.2

9661679



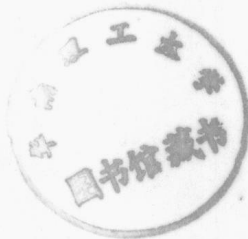
E9661679

**PROGRESS**  
**in**  
**ROBOTICS**  
**and**  
**INTELLIGENT SYSTEMS**

**VOLUME 2**

edited by  
**George W. Zobrist**  
and  
**C.Y. (Pete) Ho**

**University of Missouri—Rolla**



**Ablex Publishing Corporation**  
**Norwood, New Jersey**

Copyright © 1996 by Ablex Publishing Corporation

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, microfilming, recording or otherwise, without permission of the publisher.

Printed in the United States of America

Library of Congress Cataloging-in-Publication Data  
Progress in robotics and intelligent systems.

Includes bibliographical references.

1. Robotics. 2. Artificial intelligence.


I. Zobrist, George W. (George Winston), 1934–

II. Ho, C. Y. (Chung You), 1933–

TJ211.P76 1994 670.42'72 89-17746

ISBN 0-89391-593-9 (v. 2)

Ablex Publishing Corporation  
355 Chestnut Street  
Norwood, New Jersey 07648



## Series Preface

This series is intended for those individuals involved in robotics and intelligence systems research, design, development, and scholarly activities.

This volume is concerned with state-of-the-art developments in robotics and intelligent systems by providing insight and guidance into specific techniques vital to those concerned with design and implementation of robotics and intelligent system applications.

The material contained in this volume discusses motion learning and vision-based robotics, control algorithms, accuracy issues, networking robots, and robotic programming techniques.

The editors wish to thank the contributing authors for making available the information contained in this book.

George W. Zobrist  
C.Y. Ho  
University of Missouri-Rolla

May 1994

# Contents

Series Preface	vii
1 Experimental Analysis of Convex Volumes Enclosing Parametric Surfaces	1
<i>Chaman L. Sabharwal and Thomas G. Melson</i>	
2 Motion Learning in Robotized Mechanical Systems	27
<i>Guiseppe Casalino and Michele Aicardi</i>	
3 Vision-Based Robotic Assembly System	61
<i>Z. Bien, I.H. Suh, S.-R. Oh, and B.-J. You</i>	
4 Robot Accuracy Issues and Methods of Improvement	90
<i>Chia P. Day</i>	
5 Intensity Blending of Computer Image Generation-Based Displays	109
<i>E.A. Reidelberger and Daniel C. St. Clair</i>	
6 Long-Range Adaptive Control Algorithms for Robotics Applications	134
<i>J.M. Lemos, F. Coito, P. Shirley, P. Conceição, F. Garcia, C. Silvestre, and J.S. Sentiero</i>	
7 Research Problems in Computer Networking for Manufacturing Systems	196
<i>Asok Ray and Shashi Phoha</i>	
8 Reduced Protocol Architecture for Factory Applications	214
<i>Luigi Ciminiera, Claudia Demartini, and Adriano Valenzano</i>	
9 A Programming Methodology for Robotic Arc Welding	253
<i>Kristinn Andersen, George E. Cook, Saleh Zein-Sabattou, Robert Joel Barnett, and Kenneth R. Fernandez</i>	
Author Index	293
Subject Index	297

# Experimental Analysis of Convex Volumes Enclosing Parametric Surfaces

**Chaman L. Sabharwal**

University of Missouri—ROLLA

**Thomas G. Melson**

Computer Aided Technology, MCAIR

## 1. INTRODUCTION

Computing intersections between geometric objects in particular surfaces [1] is a key capability of CAD/CAM systems and many other geometric modeling systems. Parametric equations completely separate the roles of independent and dependent variables, both geometrically and algebraically, and allow for any number of variables (i.e., there is a natural extension from two- to three-dimensional space). Parametrically defined objects are inherently bounded because the parameter space is normalized to a unit square. There is no need to carry additional data to define boundaries. The calculation and intersection of bounding volumes for parametric surfaces is used, in a wide range of applications, as a scaffold to mitigate the agony of complex computations. The rectangular parallelepiped bounding volumes are also referred to as bounding boxes, or simply boxes. Bounding boxes are interchangeably used for rectangular parallelepiped bounding volumes. Bounding volumes are useful for parametric surface intersections [2, 3], collision detection [4], and partitioning of polyhedral objects into nonintersecting parts [5].

The intersection between objects and the collision detection problem arises in CAD/CAM applications where one needs to cull disjoint objects before computationally intensive analysis is performed on the objects. Human eyes can detect intersection easily and instantaneously, but it is a difficult problem for the computer to visualize this phenomena. The culling process is done by first enclosing the two objects in bounding volumes and determining whether or not the bounding volumes intersect. The cost of detecting the intersecting bounding volumes is significantly lower than the cost of intersecting the objects. The bounding volumes may intersect, but the objects may not. In that case, a more computation-intensive analysis is performed.

This problem arises also in other areas such as Robotics, where interference detection is to be determined; and Computer Graphics, where hidden surface removal and ray tracing take place. For intersection between surfaces, one subdivides the larger surfaces into smaller and simpler surface pieces. This subdivision process is a selective subdivision in which the surfaces are enclosed in bounding volumes and the bounding volumes are tested for intersection conditions to detect the need for further subdivision of the surfaces. It is easier to detect the intersection condition between the bounding volumes than between the surfaces themselves. Further, in some applications, because the bounding boxes are used for the culling process only, actual intersection between the bounding volumes is not required: Only a flag value is used to indicate the existence or nonexistence of intersection. Note that computation of actual intersection between the bounding volumes leads to a more complex problem of intersection between  $C^0$  surfaces. The designer of an algorithm has two goals at hand: (a) The algorithm should be understandable, easy to code, and easy to debug. (b) It should make optimal use of computer resources with respect to both storage space and execution time. We have these two goals in mind in the analysis of the methods to be considered. Several methods have been used to calculate the bounding volumes for surfaces and to detect the intersection between them. This chapter singles out one method that is mathematically sound, numerically less prone to computational errors, computationally efficient, and easier to understand and implement. The source code and load segments make efficient use of computer resources. An execution time efficiency analysis is performed to determine its suitability for use in production modules.

## 2. DISCUSSION OF METHODS

The bounding volumes for the surfaces can be computed in several ways. An important and much-debated issue is how to calculate the enclosing volume. The object of this presentation is to put this issue to rest for a long time. The methods for calculating the bounding volumes for surfaces can be classified as: (a) axis-oriented parallelepipeds [2, 3], (b) surface-oriented parallelepipeds [6], (c) convex hulls [7], (d) ellipsoids [8], and (e) spheroids [9, 10]. There is a need for an ideal method guaranteeing that the surface remains entirely within its bounding volume. However, this task is impossible for an arbitrary parametric surface with no additional structural information. In the absence of additional information, only the sampled points are used to compute the bounding volumes. There is no way to guarantee the behavior of the function values where the surface is not sampled, if samples are the only information available. It is, therefore, desirable to minimize the difference between the computed bounding volume enclosing the surface and the actual geometric volume containing the surface. At the same time, there is a need to strike a balance to achieve the best out of the available methods for bounding volumes.

Analytical comparison shows that methods (b), (c), (d), and (e) are poor choices for one or more of the following reasons: the excessive computation time for the calculation of bounding volumes, the convex hull property of the surfaces, smoothness constraints on surfaces, and the excessive performance time for intersecting the bounding volumes. The torus represents an example of a surface where none of the above-mentioned bounding volume methods works well. The simplicity of a box calculation with axis-oriented parallelepipeds and the associated simplicity in box intersection makes the axis-oriented method more acceptable than other methods.

To keep the comparisons simple, the details of the methods have been simplified. The spirit of the techniques has been retained in order to point out the complexity of the methods. For execution time analysis, run-time tests are performed on the axis-oriented and surface-oriented methods that are currently used at the McDonnell Douglas Corporation. These methods can be characterized as follows: (a) use only the position value, 0-dimensional information, to calculate axis-oriented bounding volumes or (b) use a position value and three direction vectors, 1-dimensional information, to calculate the surface-oriented bounding volumes.

## 2.1. Details of Axis-Oriented Method

This method is designed to compute approximate bounding boxes, which are oriented along the axes of the coordinate system. Two different methods for computing the axis-oriented bounding volumes, considered here, depend on the nature of the surfaces (e.g.,  $C^0$  and  $C^2$  surfaces). However, the method for  $C^0$  surfaces will still be applicable to  $C^2$  surfaces.

### 2.1.1. Axis-Oriented Method for $C^0$ Surfaces.

This method is the same as implemented in the original surface/surface intersection algorithm [2, 3]. Since it was used in the surface/surface intersection algorithm as implemented in 1981, it is referred to as SURF81. Since the implementation of this algorithm was revised in 1987 to eliminate the unnecessary calculations and reduce the repeated evaluator calls, the same method in the revised version is referred to as SURF87. In this method, it is easier to calculate the bounding volume and simpler to detect the intersection between the bounding volumes. For computation-intensive practical applications, it is not necessary to have exact bounding volumes because the approximate bounding volumes can be used without loss of information. Sample  $N^2$  points on the surface  $R(u, v)$ , equally spaced parametrically, calculate the Maximum and Minimum points over the sampled data. It gives an approximate bounding volume for the surface  $R(u, v)$ . Since the computations are over the sampled points on the surface, there is no guarantee that the surface lies entirely inside the bounding volume. Extensive empirical evidence indicates that, on the average, if the boxes over a surface

with  $D$ -degree curvature is expanded by 0.D percent, the resulting bounding volumes would normally contain the surface. This is sufficient for practical applications where the numerical accuracy of the bounding volumes is not necessary or required. However, this or any other numerical method can be frustrated by creating sufficiently unrealistic examples. Thus, the bounding volume for a surface  $R(u, v)$  is given by two points (Min and Max), such that

$$\text{Min}(i) \leq R(u, v, i) \leq \text{Max}(i)$$

for  $i = 1, 2, 3$ ;  $LWU \leq u \leq UPU$  and  $LWV \leq v \leq UPV$ .

Here  $LWU$  and  $LWV$  are the lower bounds on the parameters  $u$  and  $v$ ,  $UPU$  and  $UPV$  are the upper bounds on the parameters  $u$  and  $v$ , and  $i = 1, 2$ , and 3 refer to the  $x$ -,  $y$ -, and  $z$ -coordinates of the points.

Determining whether the axis-oriented bounding volumes intersect (or not) is as easy as calculating these volumes. Two bounding volumes ( $\text{Min}_1, \text{Max}_1$ ) and ( $\text{Min}_2, \text{Max}_2$ ) are disjoint provided there exists an  $i$ ,  $1 \leq i \leq 3$ , such that

$$\begin{aligned} \text{Max}_1(i) &< \text{Min}_2(i), \text{ or} \\ \text{Max}_2(i) &< \text{Min}_1(i). \end{aligned}$$

It shows that it is easy to calculate the bounding volumes and it is even easier to detect the intersection condition between the axis-oriented bounding volumes.

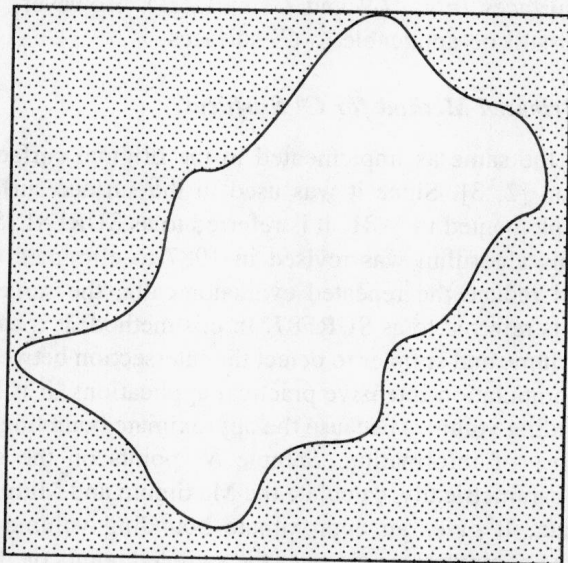


FIGURE 1.1. Axis-Oriented Bounding Box

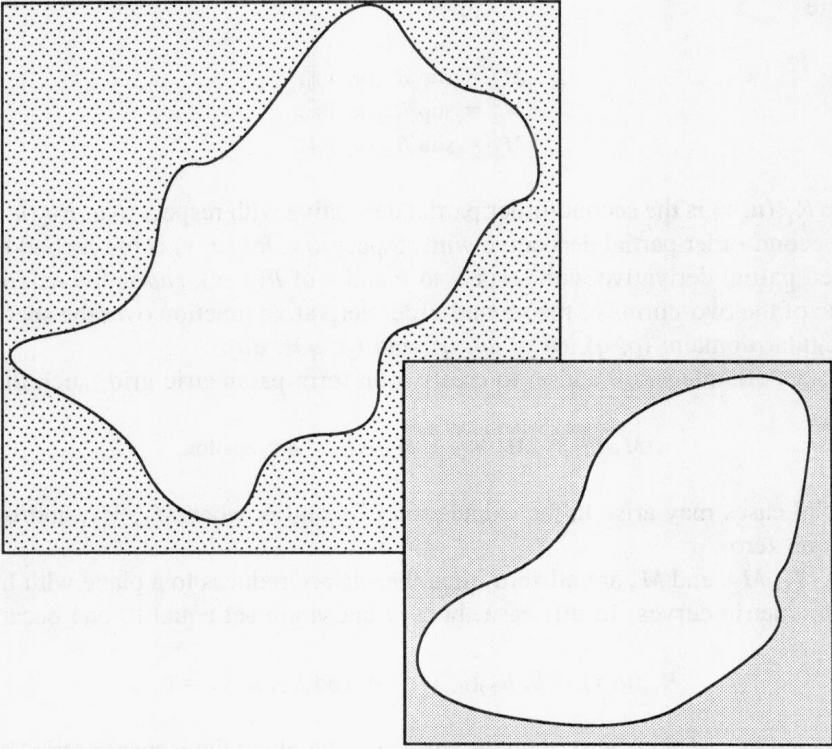


FIGURE 1.2. Axis-Oriented Boxes Intersect, Surfaces Do Not Intersect

### 2.1.2. Axis-Oriented Method for $C^2$ Surfaces.

This method applies to  $C^k$ ,  $k \geq 2$ , surfaces only and was developed at Automation Technology Products [11]. This technique depends on the ability to calculate the second-order partial derivatives of the surface with respect to the surface parameters and the bounds on these derivatives. The discussion of this method warrants the discussion of some terminology. Let  $(p, q)$ ,  $(p + dp, q)$ , and  $(p, q + dq)$  be three points in the parameter space of the surface  $R(u, v)$ . The surface  $R(u, v)$  can be linearly approximated by  $L(u, v)$  over the nondegenerate triangle  $(p, q)$ ,  $(p + dp, q)$ , and  $(p, q + dq)$  within a specified tolerance epsilon. The approximation function is given by:

$$L(u, v) = R(p, q) + (u - p)(R(p + dp, q) - R(p, q))/dp \\ + (v - q)(R(p, q + dq) - R(p, q))/dq$$

provided

$$(dp^2 M_1 + 2 dp dq M_2 + dq^2 M_3) < 8 * \text{epsilon}$$

where

$$\begin{aligned} M_1 &= \sup \|R_{11}(u, v)\|, \\ M_2 &= \sup \|R_{12}(u, v)\|, \\ M_3 &= \sup \|R_{22}(u, v)\|. \end{aligned}$$

Here  $R_{11}(u, v)$  is the second-order partial derivative with respect to  $u$ ,  $R_{22}(u, v)$  is the second-order partial derivative with respect to  $v$ ,  $R_{12}(u, v)$  is the second-order mixed partial derivative with respect to  $u$  and  $v$  of  $R(u, v)$ ;  $\sup$  is the maximum value of the two-norms of the second-order derivative function over the specified triangular domain:  $(p, q)$ ,  $(p + dp, q)$ , and  $(p, q + dq)$ .

Calculate integers  $n$  and  $m$  to create a uniform parametric grid, such that

$$(M_1/n^2 + 2M_2/mn + M_3/m^2) < 8 * \text{epsilon}.$$

Special cases may arise in the calculation of  $n$  and  $m$  when  $M_1$ ,  $M_2$ , and/or  $M_3$  become zero.

If  $M_1$ ,  $M_2$ , and  $M_3$  are all zero, then the surface reduces to a plane with linear isoparametric curves. In this case, both  $m$  and  $n$  are set equal to one because

$$R_{11}(u, v) = 0, R_{22}(u, v) = 0, \text{ and } R_{12}(u, v) = 0.$$

If  $M_1$  and  $M_3$  are both zero, then the surface is flat along the isoparametric curves and both  $m$  and  $n$  are treated as equal. This accounts for linearity of isoparametric curves used in the surface/surface intersection problem.

If  $M_1 = 0$ , then the surface is flat in the  $u$ -direction and  $n$  is set equal to 1.

If  $M_3 = 0$ , then the surface is flat in the  $v$  direction and  $m$  is set equal to 1.

If  $M_1$  and  $M_3$  are both nonzero, then the calculated mixed partial derivative term is rolled proportionally in  $u$  and  $v$  steps based on  $M_1/M_3$  (i.e.,  $n/m$  is set to be  $M_1/M_3$ ).

This method depends on the direct evaluation of second-order partial derivatives and their 2-norms. There are two problems associated with this method. First, the second-order derivatives may not exist for the surface, as in the case of  $C^1$  surfaces. Secondly, even if the derivatives exist, it may be time consuming to compute these derivatives and the bounds on the 2-norms of these derivatives. No doubt, it is easy to implement the calculation of the derivatives in the case of polynomial surfaces. In particular, these derivatives for cubic parametric surfaces reduce to linear terms and it is trivial to calculate bounds on linear expressions.

Let  $K$  be defined as

$$K = (M_1/n^2 + 2M_2/mn + M_3/m^2)/8.$$

To calculate the bounding volume for a surface, first the term  $K$  is computed and then  $\text{Max}'$  and  $\text{Min}'$  are calculated as the maximum and minimum values of  $R(p, q)$ ,  $R(p + dp, q)$ ,  $R(p, q + dq)$ , and  $R(p + dp, q + dq)$ —four corners of a surface piece. The bounding volume of this surface piece is then obtained by using  $K$  as the expansion factor:

$$\begin{aligned}\text{Min}(i) &= \text{Min}'(i) - K, \\ \text{Max}(i) &= \text{Max}'(i) + K, \quad \text{for } i = 1, 2, \text{ and } 3.\end{aligned}$$

This technique guarantees that the entire surface piece lies inside the bounding volume. It is based on the ability to calculate the second-order derivatives, to calculate the bounds on them, and finally to calculate the maximum and minimum of the corner points. This method is very successful with polynomial surfaces, specifically with cubic polynomial surfaces. However, this technique is not practical at all for general parametric surfaces. The difficulty lies in the calculation of the second-order derivatives for general parametric surfaces where the derivatives may not exist. Even when the derivatives exist, it is not easy to compute the norms on these derivatives. This problem makes such a method very clumsy to use in real time applications. In general, these norms are not used in CAGD [11].

## 2.2. Details of Surface-Oriented Method

This method differs from the axis-oriented method, SURF81/SURF87, discussed in Section 2.1.1., because the bounding volumes are not oriented along the coordinate axes. Rather, they are oriented along the surface involving the positioning of the surface. Such bounding volumes are supposed to yield smaller geometric volumes enclosing the surfaces [6]. For computation-intensive applications it is not only desirable but also necessary to minimize the number of bounding volumes. Thus, it was assumed that, in general applications, the surface-oriented bounding volumes will be fewer in number than the axis-oriented bounding volumes.

A local coordinate system for the surface piece for the surface  $R(u, v)$  is calculated on the parameter rectangle  $[u_i, u_t] \times [v_i, v_t]$ , where  $u_i, v_i$  are the initial values of the parameters and  $u_t, v_t$  are the terminal values of the parameters. Define a unit vector  $e_1$  in terms of the position values along the  $v = v_i$  or  $v = v_t$  parametric curves, if possible. Otherwise, consider  $e_1$  to be the unit vector along the positive direction of the x-axis of the coordinate system. Similarly, if possible, define a unit vector  $e_2$  in terms of the position values along the  $u = u_i$  or  $u = u_t$  parametric curves. Otherwise, consider  $e_2$  to be the unit vector along the positive direction of the y-axis of the coordinate system. Once two noncollinear vectors are determined, the third unit vector is determined by the relation

$$e_3 = (e_1 \times e_2) / |e_1 \times e_2|$$

where “ $\times$ ” denotes the cross-product between the vectors.

Since  $e_1$  and  $e_2$  are not necessarily orthogonal, the unit vector  $e_2$  is recalculated as

$$e_2 = (e_3 \times e_1) / |e_3 \times e_1|.$$

The resulting vectors  $e_1$ ,  $e_2$ , and  $e_3$  form a local orthonormal system. Relative to this orthonormal system, Min and Max are calculated as in Section 2.1.1. The surface-oriented bounding volume is defined in terms of three components:

1. Min is the anchor point.
2. The sides are oriented along the direction vectors of the local orthonormal system and they emanate from the anchor point.
3. The lengths of the sides of the bounding volume are determined by using the values of Min and Max computed above.

The intersection condition between two bounding volumes is determined after performing the following two steps:

1. Transform one of the bounding volumes in such a way that the anchor point coincides with the origin and the sides are oriented along the positive directions of x-, y-, and z-axis.

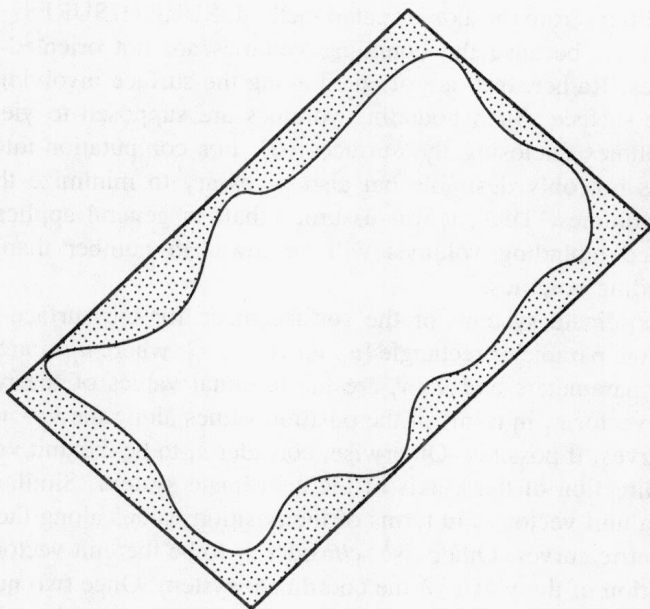
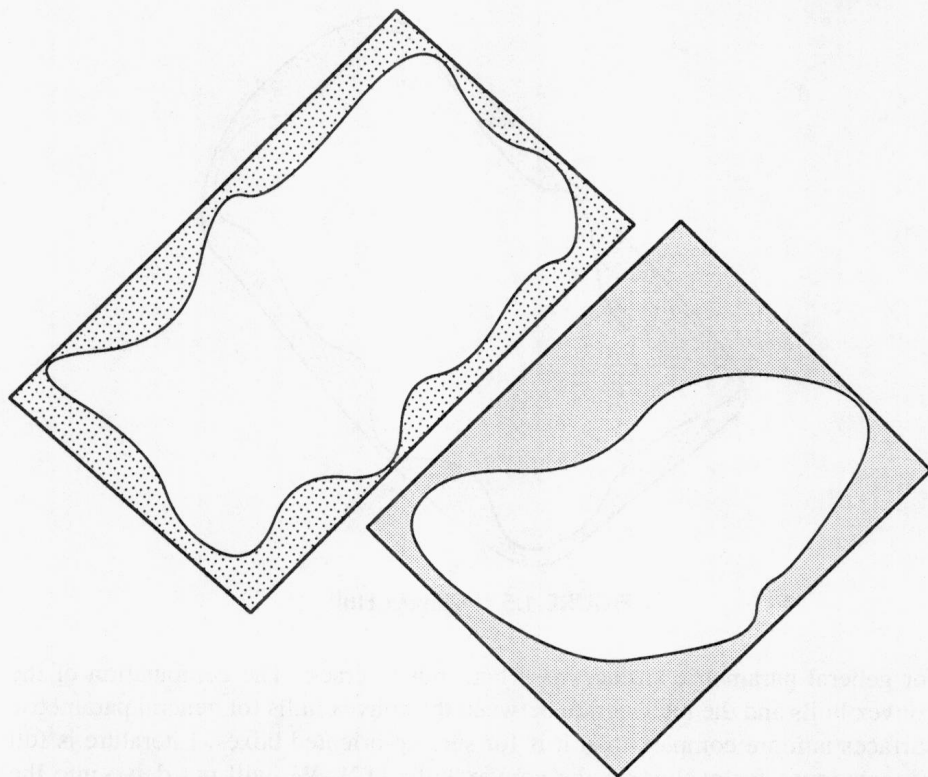


FIGURE 1.3. Surface-Oriented Bounding Box



**FIGURE 1.4. Surface-Oriented Boxes Do Not Intersect, Surfaces Do Not Intersect**

2. Apply the same transformation to the second bounding volume. The intersection condition is evaluated by performing the intersection of the edges of one box with the plane faces enclosing the other bounding volume and vice versa.

### 2.3. Details of Convex Hull Method

This method differs from methods discussed in Sections 2.1.1, 2.1.2, and 2.2 because a convex hull is not necessarily a rectangular parallelepiped. Rather, a convex hull is a systematic collection of planes wrapping around the surface such that the enclosing volume is minimal. There is less deviation between the volume of a convex hull and the geometric volume of a surface than with the previous methods.

The calculation of convex hulls is a function of control points of such specialized surfaces as B-spline and Bezier surfaces. The calculation of convex hulls

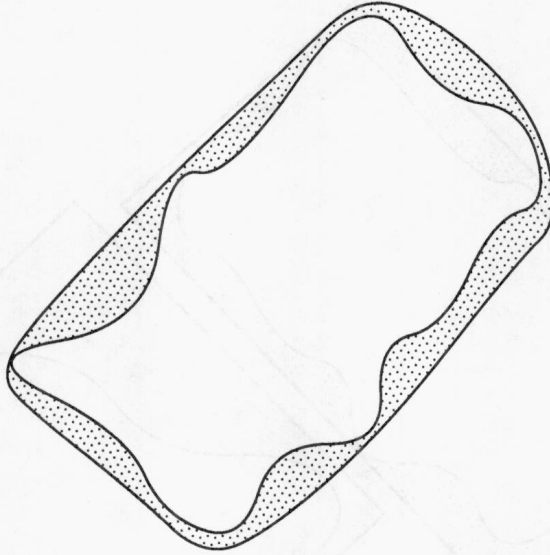


FIGURE 1.5. Convex Hull

for general parametric surfaces is a hard nut to crack. The computation of the convex hulls and the intersection between the convex hulls for general parametric surfaces is more complex than it is for surface-oriented boxes. Literature is full of techniques for evaluating the convex hulls [12]. We will not delve into the details of convex hull calculations because it is not the way to approach bounding volumes for arbitrary parametric surfaces.

#### 2.4. Details of Ellipsoidal Method

This method is applicable to  $C^k$ ,  $k \geq 1$ , surfaces only [8]. This procedure depends on the ability to calculate the first-order partial derivatives of the surface and the bounds on these derivatives. This technique differs from all the methods discussed in Sections 2.1.1, 2.1.2, 2.2, and 2.3, where the bounding volumes are enclosed by planar faces. Here the bounding volumes themselves are curved surfaces and they are ellipsoidal or spheroidal in nature.

The discussion of this method warrants the discussion of some terminology. Let the surface  $R(u, v)$  be defined over the rectangle  $[u_i, v_i] \times [u_t, v_t]$ , where  $u_i, v_i$  are the initial values of the parameters, and  $u_t, v_t$  are the terminal values of the parameters  $u, v$ . The 2-norm of  $A$  is denoted by  $\|A\|$  and is defined by the square root of the sum of the squares of the components of  $A$ . Let  $K$  be the Lipschitz constant for the surface  $R(u, v)$ , such that

$$\|R(u, v) - R(u_1, v_1)\| \leq K\|(u, v) - (u_1, v_1)\|.$$

Since

$$\|(u, v) - (u_1, v_1)\| \leq (|u - u_1| + |v - v_1|)$$

the Lipschitz condition can be replaced by a simpler modified condition

$$\|R(u, v) - R(u_1, v_1)\| \leq K(|u - u_1| + |v - v_1|).$$

This modified condition is used to calculate the ellipsoidal bounding volume with two foci at  $R(u_i, v_i)$  and  $R(u_r, v_r)$ , and with the length of major axis as

$$L = K(|u_i - u_r| + |v_i - v_r|).$$

The resulting ellipsoid becomes

$$\|R(u, v) - R(u_i, v_i)\| + \|R(u, v) - R(u_r, v_r)\| = L.$$

The calculation [8] of the Lipschitz constant,  $K$ , is found for  $C^1$  continuous surfaces as  $K =$

$$\sup(\|R_1(u, v)\| + \|R_2(u, v)\|).$$

Here  $R_1(u, v)$  is the first-order partial derivative vector with respect to  $u$ , and  $R_2(u, v)$  is the first-order partial derivative vector with respect to  $v$  of  $R(u, v)$ ;  $\sup$  is the maximum value of the sum of the 2-norms of the first-order derivatives of  $R(u, v)$  over the defining rectangle

$$[u_i, v_i] \times [u_r, v_r].$$

If  $R(u_i, v_i) = R(u_r, v_r)$  then ellipsoids reduce to spheroids. For brevity, the ellipsoid with foci  $P_1, P_2$ , and major axis length  $L_P$  is denoted by  $E_P = (P_1, P_2, L_P)$ .

The ellipsoidal technique is feasible for surfaces such as bicubics or low-order polynomials because local maximums of the parametric derivatives are easy to evaluate. If the surface is piecewise continuously differentiable, the maximum of each piece may be used. The application must know, in advance, the number of pieces used in the definition. This method at least guarantees that the entire surface will lie inside the bounding volume. Many applications, such as surface/surface intersection, use the bounding volumes for the culling process and thus do not require this guarantee of numerical accuracy. In such cases, sampled positional data may be used for calculating the derivatives [13].

The intersection condition can be determined by using the foci and Lipschitz constant for one bounding volume and the sampled positional data of the second bounding volume. That is, two ellipsoidal bounding volumes,

$$E_P = (P_1, P_2, L_P) \text{ and}$$

$$E_Q = (Q_1, Q_2, L_Q),$$

intersect if there exists a point  $P$  on  $E_P$  such that

$$\|P - Q_1\| + \|P - Q_2\| \leq L_Q \text{ or}$$

there exists a point  $Q$  on  $E_Q$ , such that

$$\|Q - P_1\| + \|Q - P_2\| \leq L_P.$$

This method is simpler than the surface-oriented technique and more complex than the axis-oriented method for the calculation and intersection of bounding volumes. However, it depends on the availability of calculations for the first-order partial derivatives and the norms on them. Hence, this method is not applicable to arbitrary parametric surfaces. Since axis-oriented and surface-oriented methods both use positional data, they are applicable to a more general class of parametric surfaces. These methods will be the prime candidates for run-time analysis in this discussion.

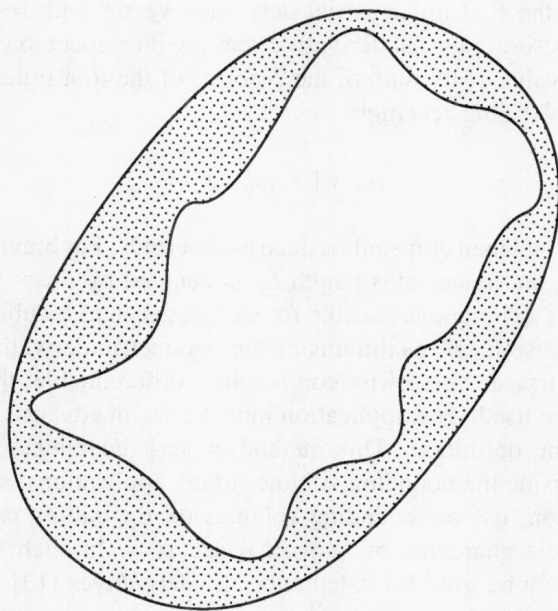


FIGURE 1.6. Ellipsoidal Bounding Volume