Reverse Engineering from Exact and Noisy Data of Objects Defined by Algebraic Surface Patches

Michel Bercovier, Moshe Luzon, Elan Pavlov*

School of Computer Science and Engineering

The Hebrew University

Jerusalem 91904, Israel

Tel: 972-2-6584961; Fax: 972-2-6758958

{berco,moshelu,elan}@cs.huji.ac.il

Abstract

A robust and efficient algorithm for the identification of algebraic surface patches from noisy data is detailed here. The aim is to reconstruct the B-rep of solids made of engineering forms constructs. There is no apriori knowledge on the number of patches. Algebraic surfaces are embedded on hyperplanes in R^A, where A is the number of points sufficient to determine an algebraic surface. A probabilistic algorithm and graph theory are used to cluster the patches. Non trivial examples of quadric surface patches are given.

Keywords: Reverse Engineering, Noisy Data, Algebraic Surfaces, Quadrics, Hyperplanes, Clustering, Probabilistic Algorithm.

Introduction

In many applications in CAD, Reverse Engineering and Computer Vision, $3D$ objects are given by measurements in space (see [1],[2]). Such objects are made of bounded patches, with patches being defined as (part of) planes, algebraic surfaces or parametrized (free form) surfaces. The measurement process performed on these objects, results in a large number of $3D$ points. The points produced are given up to an unknown (but usually bounded) error, moreover most of

*Authors names are arranged in alphabetical order

the time the data is unorganized. The problem is to reconstruct the objects, which were originally made of smooth patches bounded by contours, from the given unorganized set of points. Approximation of free form patches has been extensively studied, but most of the methods are of very high complexity (see [3]). Engineering shapes are very often made of simple algebraic patches and more precisely quadrics and planar facets, hence one can try to identify such patches. In a preceding work the problem of reconstructing planes, from an unorganized measured set of points, which is an essential step in the general context of object reconstruction (see [4]), was solved in details. The present work consider the case of reverse engineering of algebraic surface patches from noisy data. The method developed here is general but all practical examples will consist of quadric surface patches.

Standard methods to solve this problem are based on different variants of least squares approximation (for example see the survey article [5]), random sampling [7], and Hough Transform ([8], [9], [10]).

Least squares approximation and random sampling do not always provide a robust solution, in addition even when the answer is satisfying the cost is relatively high ([5] is in $O(n^3)$ and for the case of an algebraic surface [7] is in $O(nk^A)$, where n is the total number of points, k is the number of original algebraic surfaces, and A is the number of points sufficient to determine each algebraic surface. Most of the time the actual implementation is for a single patch.

The Hough Transform method heavily depends on parameter settings, such as the maxima threshold for choosing the viable hypotheses, moreover there exist examples in which it leads to incorrect results. In practice it implies that the space of parameters is divided in a grid of size n^A where n must be large enough to separate the patches. Most of the examples are limited to $2D$ cases.

Another approach is based on the Least-squares fitting of algebraic spline surfaces (see [11]), this novel approach simultaneously approximates points and associated normals, which are estimated from the data, one obtains a method which is $O(n^2 + h^2)$, where n is the number of points, and h is the number of unknown coefficients, because the system of equations is sparse. However the crucial step in the surface fitting process, the estimation of normals has to be possible. Thus, there sould be no creases in the data, the distribution of the data should be not too irregular and the level of numerical noise should be relatively small.

The present work suggests a new approach for determining the algebraic surface patches from an unorganised exact data or noisy data. The data set is unorganised, there is no apriori knowledge as to which patch a measured point may belong, and part of the problem is to identify the patches. The only input required is an upper bound on the number of patches to be approximated. To solve the present problem one has to implement clustering methods in an efficient way. The core of the work is a probabilistic algorithm that clusters a random subsets selection of the initial data.

The paper is divided into three sections. The first section contains the formulation and solution of the problem with exact data, the solution is based on exhaustive evaluation of all possible patches and thus very expensive. Its main purpose is to provide an insight to problem caused by noisy data. In the following section the problem with noisy data is introduced. To solve it, a probabilistic clustering algorithm of low complexity, based on proximity and clustering, is introduced. This algorithm can be used for the solution of both problems. The third section discusses some experimental results which use quadric surfaces as an example. Finally conclusions and future research directions are outlined.

1 Problem Definitions

Given a set of points in R^3, $\{P^i\}_{i=1}^n$, such that there exist unique k (unknown) algebraic surfaces $S_1, ..., S_k$ and k unknown index sets :

$$\{A_j\}_{j=1}^k, |A_j| = i_j,$$

with i_j being the greatest integer numbers which satisfy :

$$\{\cup_{l \in A_j} P^l\} \in S_j, j = 1, ..., k.$$

The problem is to determine the algebraic surfaces $S_1, ..., S_k$ and the partition (possibly with repetitions)

$$\{P^i\}_{i=1}^n = \cup_{j=1}^k \{\cup_{l \in A_j} \{P^l\}\}.$$

1.1 Algorithm for solving the problem with exact data

Assume that each algebraic surface of degree d is represented by an implicit equation of the form:

$$\sum_{|\vec{i}|=0}^{d} a_{ijk} x^i y^j z^k = 0$$

where $\vec{i}$ is the index triplet $\vec{i} = (i, j, k)$, $|\vec{i}| = i + j + k$;

Note that some of the coefficients might be zero. Each group of A points where $A = (d/6)(d^2 + 6d + 11)$, which is not degenerate, determines an algebraic surface of degree d in a unique fashion up to a multiplicative factor. The algorithm is as follows.

Step 1: Given the points $\{P^i\}_{i=1}^n$, select all of the subsets consisting of exactly A (distinct) points, where A is noted above. For each subset find the algebraic surface it defines. If the

algebraic surface is degenerate discard the corresponding set. The execution time of this step is then, $O(\binom{n}{A}) = O(n^A)$

Note 1. Algebraic surfaces, computed from groups of A points belonging to the same algebraic surface, coincide. Most of the algebraic surfaces which were defined by A points belonging to different algebraic surfaces, are different from each other.

Step 2: Define a sample on the algebraic surfaces obtained. Select the most frequent algebraic surfaces (for instance, with comparison to a function of the average number of times an algebraic surface is obtained), then define this collection of algebraic surfaces $\{S_i\}$ as the target set. Define the partition, by unions of the points indices of all the groups of A points defining each resulting algebraic surface :

$$\{P^i\}_{i=1}^n = \cup_j \cup_{l_1,l_2,...l_A} \{\{P^{l_1}, P^{l_2}, ..., P^{l_A}\} | \{P^{l_1}, P^{l_2}, ..., P^{l_A}\} \in S_j\}.$$

This step can easily be performed in time $O(n)$, thus, the total cost of this algorithm is $O(n^A)$. Although the complexity of this algorithm is relatively high, it provides a new insight, which will be the basis for data with the presence of errors (noisy data).

1.2 Formulation of the problem with points coordinates given up to an error

Let $\{P^i\}_{i=1}^n$, $P^i = O^i + \overline{\varepsilon}_i$, where $\overline{\varepsilon}_i$ is an unknown error vector, be a set of points in R^3, and suppose that there exist unique k (unknown) algebraic surfaces $S_1, ..., S_k$ and k unknown index sets :

$$\{A_j\}_{j=1}^k, |A_j| = i_j,$$

with i_j being the greatest integer numbers which satisfy :

$$\{\cup_{l\in A_j} O^l\} \in S_j, j = 1, ..., k.$$

The problem is to determine an approximation to the algebraic surfaces $S_1, ..., S_k$.

2 A probabilistic algorithm for the solution of both problems

2.1 Hyperplane Embedding

The first step is derived from observing that working in the space of algebraic surfaces is quite difficult, hence one embeds each algebraic surface in an hyperplane in R^A, where A is given by : $A = (d/6)(d^2 + 6d + 11)$, d - the degree of the surface. The embedding actually consists of replacing each monomial of the algebraic surface by a new independent variable thus neglecting the dependency in the algebraic surface axes and getting an hyperplane with the same coeficients. More formally, given an algebraic surface defined by :

$$\sum_{|\vec{i}|=0}^{d} a_{ijk} x^i y^j z^k = 0$$

where $\vec{i}$ is the index triplet $\vec{i} = (i, j, k)$, $|\vec{i}| = i + j + k$; map,

$$s(\vec{i}) = l, |\vec{i}| \neq 0, l = 1, .., (d/6)(d^2 + 6d + 11), \forall(i, j, k),\ x^i y^j z^k \longrightarrow \Psi_l$$

Now the problem is transformed into one of approximation in the space of hyperplanes. In order to solve it, it is necessary to introduce an appropriate measure of proximity between two hyperplanes.

2.2 Proximity Measures

In [12] Pottmann and Peternell introduced a Euclidean metric in the affine space obtained by removing from the dual projective space all the planes parallel to the z-axis. The metric is

defined by first expressing each plane as a function of x, y, then taking the $L^2(\mu)$ (μ is the Lebesgue measure $dxdy$ times the characteristic function of the *region of interest*) norm of the difference between the two functions obtained. This defines a *distance* of two planes within some *region of interest*. Another metric between planes can be defined by taking the Euclidean distance between the footpoints of the normals to the planes, taken from the barycentric center of the points' cloud. Both approaches depend roughly on the area of interest which is assumed to be close enough to their intersection. Moreover the latter one is sensitive to noise.

Since the points coordinates are obtained up to measurement errors, one needs a *relation* which is relatively robust to errors in the coordinates of the points. The angle between the unit normals of two hyperplanes is a good candidate. It is not sufficient since if two intersecting hyperplanes with a small angle are close to each other just near their intersection, their distance becomes arbitrarily large if one moves away from the intersection. To overcome this, a three step *relation* between hyperplanes is introduced.

Given a collection of hyperplanes ${\pi_i}_{i=1}^m$ in R^n, in an area of interest T, define the *relation* between each different pair of them as an ordered triplet (θ, d_1, d_2), where:

Definition 1.

(a.1) θ is defined as the angle between the unit normals, in the same direction, of $\pi_{i,}\pi_j$ ($0 \leq \theta \leq \pi/2$), $1 \leq i \neq j \leq m$;

(a.2) $d_1 = min\{d(x, \pi_j), d(y, \pi_i) : x \in \pi_i \cap T, y \in \pi_j \cap T\}$, $1 \leq i \neq j \leq m$;

(a.3) $d_2 = (d_1 - \overline{d_1})^2$, where the average , $\overline{d_1}$, is taken over all the distances between each pair of different hyperplanes as defined in the definition of d_1, (cf. figure 1).

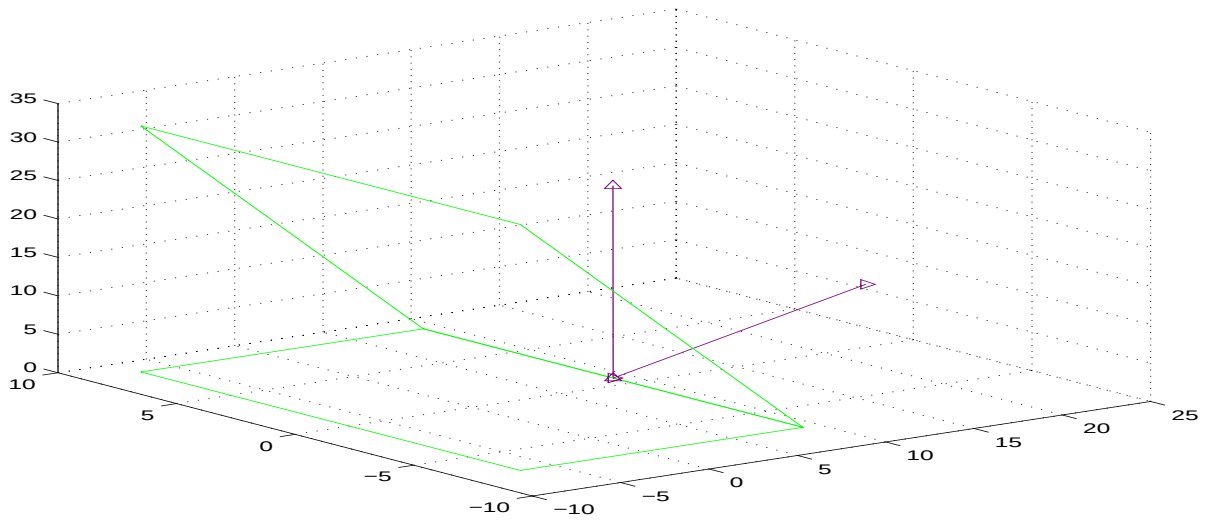

Figure 1: *An example of the angle between two planes. Note that the unit normals are always taken in the directions which define the angle* θ *to satisfy,* $0 \leq \theta \leq \pi/2$*;*

Note 2. Computations for (a.2) are expensive, hence in practice it is replaced by the following estimation:

$$min\{d(x_l, \pi_j), d(y_l, \pi_i) : x_l \in \pi_i \cap T, y_l \in \pi_j \cap T\}, 1 \leq l \leq A, 1 \leq i \neq j \leq m.$$

Each of components (a.1), (a.2) defines a semi-pseudometric, that is it does not satisfy the triangle inequality ("semi"), nor the strict positiveness axiom $d(X, Y) = 0$ implies $X = Y$.
For component (a.3), one does not have necessarily $d_2(X, X) = 0$, its role is to sharpen the separaration criteria.

This approach does not suppose any assumptions on the relation between the hyperplanes in the *region of interest*. The hyperplanes can intersect inside the *region of interest*, outside this region, they can be parallel, or coincide, moreover from a computational aspect, the *relation* between the hyperplanes can be calculated without having to identify their intersection. Note that the order in the triplet (θ, d_1, d_2) is not arbitrary.
Once a measure between hyperplanes is defined, the next step is to build clusters and to select the relevant ones.

2.3 Clustering Method

Let G be a weighted undirected graph, $G = (V, E)$ whose vertex set V,, consists of (algebraic surfaces) hyperplanes, and whose edges set is $E \subseteq V \times V$, with weights (edges-costs) equal to the calculated *relation*-component.

Definition 2. *a bridge* is a vertex (surface) in a min-cost spanning tree (MST) of the graph G, connecting clusters while not belonging to any of these clusters.

(See [13], or any book on algorithms, for a full discussion of MST)

The method is then as follows:

(b.1) Find a min-cost spanning tree (MST) T, of the graph G (e.g using Prim's algorithm).

(b.2) Compute the average cost of arcs in the MST.

(b.3) Define two neighboring hyperplanes to be in the same cluster if the *relation*-component between them is less than (or equal to) a threshold parameter t, defined by a function of $Mean(T)$ to be chosen separately with respect to each *relation*-component used, otherwise define the nodes to be in different clusters.

(b.4) Repeat stage (b.1) on each cluster accepted by the previous stage.

(b.5) Repeat stage (b.2) on each MST accepted by the latter stage, with excluding the large edges from the average computation.

(b.6) Discard *bridges* from each MST by repeating stage (b.3), with a modified threshold parameter t, defined by a new function of the average cost.

Note 3. Stages (b.4)-(b.6) are designated for discarding the *bridges*. Using this method the *bridges* aggregate in tiny clusters to be removed later.

2.4 Selection of Clusters

Given a set of clusters $\{C_i\}_{i=1}^m$, discard the irrelevant clusters while retaining only suitable clusters by choosing all the clusters of size larger that a selection parameter s defined by a percentage of the maximal cluster size, or some other statistical measure defined on the cluster size, or merely a constant value.

2.5 Filtering of Clusters

Given a set of clusters $\{C_i\}_{i=1}^m$, augment the internal proximity of each cluster $\{C_i\}_{i=1}^m$ by filtering each cluster $\{C_i\}_{i=1}^m$ and retaining only a fixed number of vertices (algebraic surfaces) inside it which conform to a given criterion. Consider the criterion of retaining a predefined number (parameter) l of algebraic surfaces of each cluster, (l will be called the size parameter), with l minimal sums of a predefined *relation*-component parameter RC from all other surfaces, l - is chosen to be a function of the expectation value.

2.6 Algorithm's definitions and abbreviations

Let d be the algebraic surfaces maximal degree (for quadric $d = 2$), the dimension of the hyperplanes space.

Let k be an upper bound on the number of algebraic surfaces *expected* to be detected.

Let NZC - be a vector of union of the indices of the nonzero coefficients over all algebraic surfaces involved in the scene.

Let A be the dimension of NZC.

Note 4. k will not impose a constraint (as in most algorithms) on the number of algebraic surfaces detected (i.e the new algorithm below, may detect more (or less) than k algebraic surfaces), when k is not known it can be increased iteratively till convergence of the results.

Note 5. NZC will be used to reduce the complexity of the method. In the worst case NZC consists of all indices of the coefficients of an algebraic surface of degree d.

Note 6. In a scene containing algebraic surfaces one should assume that all points are far from algebraic surfaces intersections. Moreover if executed once, only with the maximal degree, the algorithm may identify reducible algebraic surfaces, for example $z^2 - 1 = 0$ describes two planes $z - 1 = 0$ and $z + 1 = 0$. To prevent the occurence of this situation the algorithm should be executed with increased d values from $d = 1$ till the maximal degree, each time the constructed surfaces are recorded, a classification method is defined and applied on these surfaces, the points assigned to these surfaces are removed.

2.7 The Probabilistic Algorithm

Define the events as selections of subsets consisting of exactly A (distinct) points. Thus, one can assume a uniform probability distribution on the above defined events. Next the sample space is defined as the space of all possible selections of A distinct points, with uniform probability distribution. Note that there are $\binom{n}{A}$ possible events.

Consider the space of all possible algebraic surfaces defined by A distinct points, there are two categories of algebraic surfaces in that space.

Definition 3. *a "pure"* algebraic surface is an algebraic surface formed by A points belonging to one original algebraic surface.

Definition 4. *a "mixed"* algebraic surface is an algebraic surface formed by A points belonging to more than one original algebraic surface.

2.7.1 The algorithm

Using these definitions the following steps are executed:

Step 1, Randomization :

(1.1) Given the points $\{P^i\}_{i=1}^n$, choose at random according to the uniform distribution, c subsets consisting of exactly A distinct points (c will be determined later).

(1.2) For each subset compute the algebraic surface it defines. If the algebraic surface obtained is degenerate, discard the corresponding set and choose another subset.

The execution time of this step is then, $O(c)$.

For each relation *relation*-triplet component θ, d_1, d_2 do step 2, step 3.

Step 2, Relation's Component Calculation :

(2.1) Select next component of the *relation* defined.

(2.2) Compute the *relation* between each pair of algebraic surfaces (from step 1), defined by this component. Build a weighted undirected graph $G = (V, E)$, whose vertex set V,, consists of the algebraic surfaces found, and the edges set is $E = V \times V$ (i.e G is a complete graph), with weights (edges-costs) equal to the calculated relations.

(2.3) Perform a clustering on the graph using the method mentioned in section 2.3, the threshold parameter t will be chosen separately for each *relation*-component.

Note 7. There exists a high probability of getting a large number of *"pure"* algebraic surfaces, since *"mixed"* algebraic surfaces are actually isolated, *"pure"* algebraic surfaces are close to each other - and thus clustered together. *"Mixed"* algebraic surfaces are far from each other (with high probability) - and therefore separated and found in small clusters. *"Mixed"* algebraic surfaces can be close to *"pure"* algebraic surfaces or close to each other and even closer than *"pure"* ones - in this case they will be clustered together but this is logical since in this case they define a useful approximation to the original algebraic surface, (cf figure 2).

The cost of this step is $O(c^2)$.

Step 3, Clusters Filtering :

(3.1) To discard the clusters consisting of *"mixed"* algebraic surfaces, while retaining only clusters of *"pure"* algebraic surfaces, perform a selection of clusters, using the method mentioned in section 2.4. The selection parameter s will be chosen separately for each *relation*-component.

(3.2) Filter the clusters accepted using the filtering method mentioned in section 2.5, the pre-

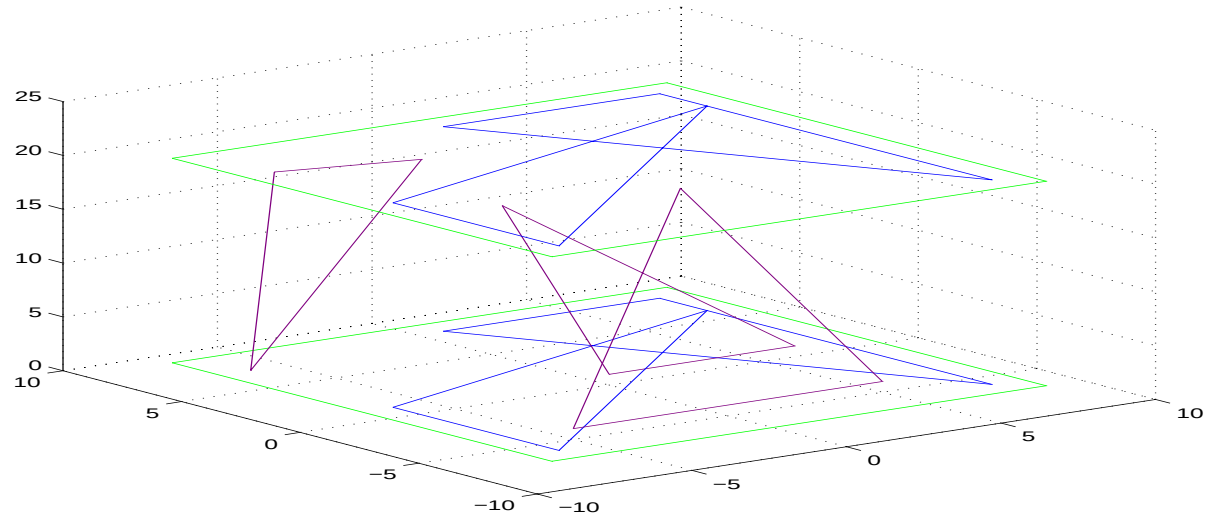

Figure 2: *"pure" and "mixed" algebraic surfaces (the case of planes - $d = 1, A = 3$. The figure illustrates the fact that "pure" planes are typically closer than "mixed" planes, with relation measured by the above defined relation - components.*

defined size parameter l, and the predefined *relation*-component parameter RC, will be chosen seperately for each *relation*-component. Note that in fact this step is not always performed.

The cost of this step is $O(c^2)$.

Step 4, Choosing Representatives :

(4.1) Choose as representative of each cluster its barycentric center in the space of hyperplanes. The representatives constructed by this method are well defined if the hyperplanes are normalized to have the same free coefficient. Instead of the barycentric center, any other acceptable criterion (i.e least squares approximation variants), could be used.

(4.2) Call the representative of cluster i: S_i.

The cost of this step is $O(c)$. Thus, the total cost of this algorithm is $O(c^2)$.

Note 8. The selection of the clusters in step 3 can also be defined by the following methods (the best one depends on the model of noise assumed and the related statistical behavior of the measurements):

1. Select the k largest or k best clusters (in terms of average distance or variance).

2. Select all the clusters that have a small variance (as defined by the present *relation* or by any other relatively robust *relation* defined on the space of hyperplanes).

It remains to determine c. This is done in the next section.

2.7.2 How many subsets are sufficient ?

A natural question arises here : what is the minimal sufficient number of subsets ?

If there are k algebraic surfaces, and if the measured points are equally distributed among the algebraic surfaces, then the probability of selecting a point from a given algebraic surface is $p = k^{-1}$, thus a minimal assumption can be that there exist a lower bound on p, or an upper bound on k. The probability of getting A points from the a given algebraic surface is then k^{-A} (independent events). Hence, the probability of having all A points on the same algebraic surface, is a monotonously decreasing function of the number of algebraic surfaces.

One's objective is to select a sufficient number of *"pure"* algebraic surfaces, therefore one has to choose a large amount of such algebraic surfaces. The number of subsets selected, can be determined by the expectation value of the corresponding binomial distribution (cf [14]), If one chooses a positive constant expectation value E, then the number of subsets to be selected, is typically $c = O(k^A)$. In the experiments, the choice was $E = 25$, thus the number of subsets was $c = 25k^A$. The execution time of the algorithm is then $O(k^{2A})$, a cost which does not

depend on n (the number of data points). For quadrics with full dimensionality one obtains $O(k^{2*9})$.

Data points are assigned to the identified surfaces by the following algorithm:

- Map each point $P^i = (X_i, Y_i, Z_i)$, $i = 1, ..., n$ to the A dimensional point P^i_A by the following mapping :

$$(X_i, Y_i, Z_i) \longrightarrow (X_i^1 Y_i^0 Z_i^0, ..., X_i^j Y_i^k Z_i^l, ...)\ ,\ (j + k + l) = 1, ..., d.$$

- Assign the data point to an algebraic surface if the distance between them in R^A is less than a predefined tolerance.

$$P^i \in S_j \Longleftrightarrow \{Distance(P^i_A, S_j) \leq 10 * Err\}$$

2.8 A Comparison with other methods

Consider two parallel concentric spheres at a small distance. The least min squares algorithm produces a single sphere in the middle, the least med squares [6] will also find a single sphere between the two original ones, finally RANSAC [7] results in one of the spheres. (Obviously, in these methods one could specify that one was looking for two spheres and get both of them BUT if there was actually only one original sphere, one would still get two!). The method introduced here detects two spheres with a distance between them (as most of the computed *"mixed"* spheres will be far from each other), and gives only one sphere if there exists only one. The quality of the approximation produced can be expressed both in terms of *relation*-components and of the surfaces coefficients. The algorithm's probability of success increases with the number of points chosen from each original algebraic surface.

3 Experimental Results and discussion

The probabilistic algorithm was prototyped in MATLAB, and tested on several examples. Recall the expectation value is $E = 25$. The number of subsets selected was $25k^A$, these subsets were all randomized by a uniform distribution. In the implementation we have repeated step 2 and step 3 in the loop twice for each *relation*-component (a.2) and (a.3) each time with different parameters for improving the internal proximity of the clusters accepted and to discard the spurious clusters which may appear.

(1) Clustering with *relation*-component (a.1).

The following parameters were used :

- A threshold parameter $t_1 = Mean(T_1)/4.0$, where T_1 is a min-cost spanning tree (MST).
- A selection parameter $s_1 = (E/5.0)$.
- Filtering step was empty (i.e filtering was not performed in this stage).

Note 9. This stage separates the intersecting algebraic surfaces with large angles between them, without separating the algebraic surfaces which were nearly parallel (as the angle between such algebraic surfaces is small).

(2) Sub-Clustering with the *relation*-component (a.2).

The following parameters were used :

- A threshold parameter $t_2 = Mean(T_2) * 10.0$, where T_2 is a min-cost spanning tree (MST).

- A selection parameter $s_2 = 3$.
- Filtering step with a size parameter $l_1 = 1.2 * E$ and a *relation*-component parameter $RC_1 = (a.2)$.

Note 10. This stage tries to separate the nearly parallel algebraic surfaces which did not overlap. However it results in clusters which contains (nearly parallel) surfaces (*bridges*) which conform to *relation*-components (a.1) and (a.2), but are far from the other (original) surfaces causing the average, computed for *relation*-component (a.3), to be increased. Thus these surfaces have to be discarded from the clusters, for this purpose stage 3 is added.
After this stage the results include spurious clusters (Clusters of *"mixed"* algebraic surfaces). The internal proximity of these clusters is inferred by *relation*-components (a.1) and (a.2). The coefficients of the clusters' elements are not necessarily close to each other, consequently additional clustering stages are needed.

(3) Discarding the *bridges* by sub-clustering with the *relation*-component (a.2).

The following parameters were used :

- A threshold parameter $t_3 = Mean(T_3) * 6.0$ (Excluding the bridges), , where T_3 is a min-cost spanning tree (MST).
- A selection parameter $s_3 = 3$.
- Filtering step was empty (i.e filtering was not performed in this stage).

Note 11. This stage discards the (nearly parallel) surfaces (*bridges*) which conform to *relation*-components (a.1) and (a.2), but are far from the other (original) surfaces. Thus it separates the

nearly parallel surfaces which do not overlap.

Spurious clusters still exist, consequently additional clustering stages are needed.

(4) Sub-Clustering with the *relation*-component (a.3).

The following parameters were used :

- A threshold parameter $t_4 = 10^{-5}$.
- A selection parameter $s_4 = 3$.
- Filtering step with a size parameter $l_2 = 0.6 * E$ and a *relation* component parameter $RC_2 = (a.1)$.

Note 12. The purpose of this stage is to decompose spurious clusters of previous stage into tiny clusters, and to filter clusters in order to retain clusters of high internal proximity.

Spurious clusters still exist, this stage is to be repeated with different parameters.

(5) Sub-Clustering with the *relation*-component (a.3).

The following parameters were used :

- A threshold parameter $t_5 = 10^{-6}$.
- A selection parameter $s_5 = 0.4 * E$.
- Filtering step was empty (i.e filtering was not performed in this stage).

Note 13. The threshold parameter used is of one magnitude lower than the latter parameter, this leads to a better decomposition and a higher internal proximity is achieved.

Note 14. All MSTs of the graphs generated, were calculated using Prim's algorithm.

3.1 Examples

(1) Randomizing points on the following two uniformly distributed quadrics, in the $3d$ cube $[-100, 100]\times[-100, 100]\times[-100, 100]$, each point was perturbed with a different random uniform error in the range $[-10^{-3}, 10^{-3}]$:

The two original quadrics were :

- $-42.1095X + 30.3433Y + 26.5586Z + -0.7186X^2 + -0.3253XY + 0.7513XZ + 0.5098Y^2 + -0.0877YZ + -0.0088Z^2 = 1000$
- $-99.4043X + -36.4223Y + -70.9303Z + -0.3335X^2 + 3.7163XY + -2.4585XZ + 1.7068Y^2 + 1.7150YZ + -1.4570Z^2 = 1000$

All stages were necessary, the following surfaces were reconstructed (cf figure 7):

- $-42.1088X + 30.3428Y + 26.5578Z + -0.7186X^2 + -0.3252XY + 0.7512XZ + 0.5098Y^2 + -0.0877YZ + -0.0088Z^2 = 1000$
- $-99.4047X + -36.4260Y + -70.9299Z + -0.3344X^2 + 3.7154XY + -2.4597XZ + 1.7065Y^2 + 1.7144YZ + -1.4574Z^2 = 1000$

The angles between the corresponding pairs of hyperplanes are : $\theta_1 = 0.00031228$, $\theta_2 = 0.00159669$.

(2) Randomizing points on the following 12 "parallel" quadrics [the term "parallel" refers to the hyperplanes embedding of the surfaces (i.e the hyperplanes are parallel)], in the $3d$

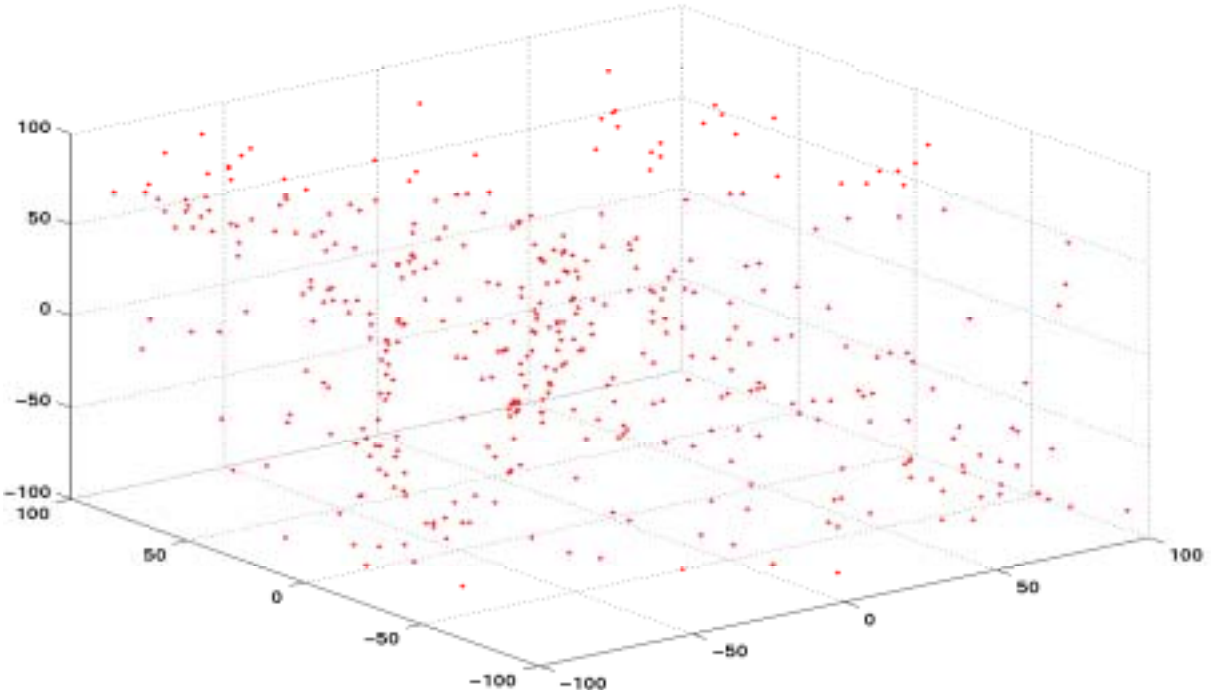

Figure 3: *Two randomly distributed quadrics in the* $3d$ *cube* $[-100, 100] \times [-100, 100] \times [-100, 100]$. *200 points were sampled from each quadric, each point was perturbed with different random uniform error in the range* $[-10^{-3}, 10^{-3}]$.

cube $[-4, 4] \times [-4, 4] \times [-4, 4]$, each point was perturbed with a different random uniform error in the range $[-10^{-3}, 10^{-3}]$:

The 12 original quadrics were :

- $X^2 + Y^2 - Z^2 = C_i$ where $C_i = 0, .., 11.$

The first three stages were sufficient, after all stages were performed the following surfaces were reconstructed (cf figure 8):

- $X^2 + 0.9949Y^2 - 0.9976Z^2 = 0.0017$
- $X^2 + 1.0004Y^2 - 1.0000Z^2 = 1.0009$
- $X^2 + 0.9997Y^2 - 0.9996Z^2 = 2.0010$

- $X^2 + 1.0003Y^2 - 0.9998Z^2 = 3.0021$
- $X^2 + 0.9992Y^2 - 1.0002Z^2 = 3.9973$
- $X^2 + 0.9998Y^2 - 0.9997Z^2 = 4.9981$
- $X^2 + 0.9996Y^2 - 0.9998Z^2 = 5.9990$
- $X^2 + 1.0005Y^2 - 1.0009Z^2 = 6.9995$
- $X^2 + 1.0003Y^2 - 1.0005Z^2 = 8.0000$
- $X^2 + 1.0000Y^2 - 0.9998Z^2 = 8.9999$
- $X^2 + 1.0002Y^2 - 0.9998Z^2 = 10.0019$
- $X^2 + 1.0003Y^2 - 1.0004Z^2 = 11.0008$

The resulting surfaces were normalized to obtain a unit coefficient of X^2.

The angles between the corresponding pairs of hyperplanes are : $\theta_1 = 0.10433026$, $\theta_2 = 0.00931846$, $\theta_3 = 0.00822263$, $\theta_4 = 0.01012355$, $\theta_5 = 0.02014782$, $\theta_6 = 0.00572290$, $\theta_7 = 0.00831907$, $\theta_8 = 0.01746868$, $\theta_9 = 0.01105816$, $\theta_{10} = 0.00416654$, $\theta_{11} = 0.00821341$, $\theta_{12} = 0.00934327$.

(3) Randomizing points not uniformly (with ratio 2:1) on the unit sphere and a canonical one sheeted hyperboloid, in the $3d$ cube $[-1, 1] \times [-1, 1] \times [-1, 1]$, each point was perturbed with random uniform error in the range $[-10^{-3}, 10^{-3}]$:

The quadrics equations were :

- $X^2 + Y^2 + Z^2 = 1$

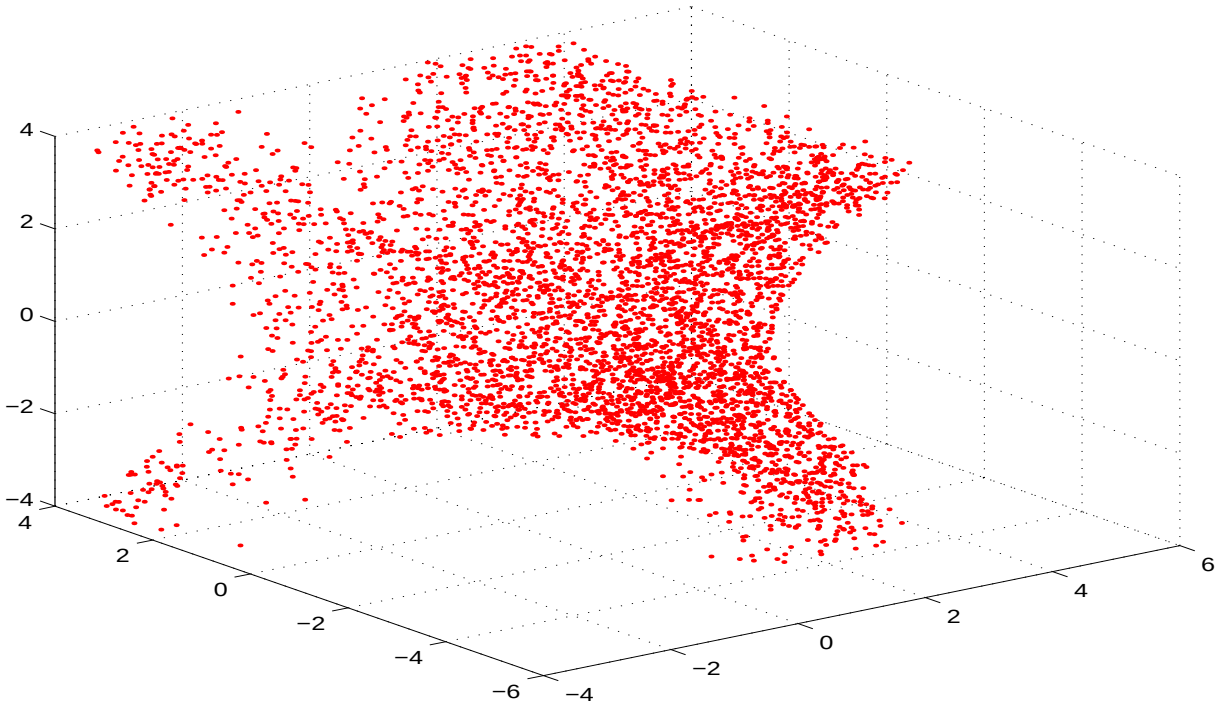

Figure 4: *12 "parallel" quadrics :* $X^2+Y^2-Z^2=C_i, C_i=0,..,11$ *. 400 points were sampled in the* $3d$ *cube* $[-4,4]\times[-4,4]\times[-4,4]$*,each point was perturbed with different random uniform error in the range* $[-10^{-3},10^{-3}]$.

- $X^2+Y^2-Z^2=1$

A lower bound on the probability of point belonging to a surface is $p=1/3$ therefore the algorithm was executed with $k=3$. All stages were essential, the following surfaces were reconstructed (cf figure 9):

- $X^2+0.9993Y^2+1.0002Z^2=0.9998$
- $X^2+0.9987Y^2-1.0012Z^2=0.9982$

The angles between the corresponding pairs of hyperplanes are : $\theta_1=0.01958166$, $\theta_2=0.05030227$.

(4) Sampling points on a hemisphere on top of a $3d$ cube. The cube has the center: $(60,-20,30)$ and edge length: 200. The hemisphere has a center: $(60,-20,130)$ and radius: 80. The

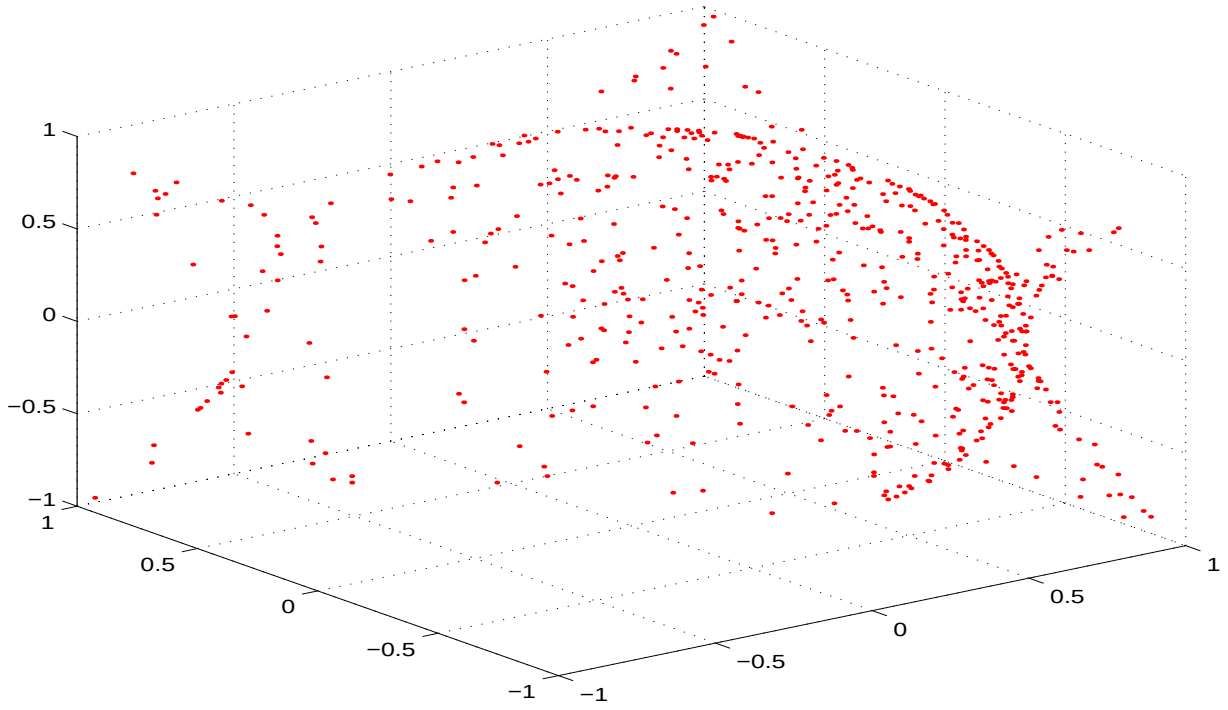

Figure 5: *400 Points were sampled the unit sphere and 200 points were sampled from a canonical one sheeted hyperboloid, in the* $3d$ *cube* $[-1,1]\times[-1,1]\times[-1,1]$*, each point was perturbed with different random uniform error in the range* $[-10^{-3},10^{-3}]$.

points were sampled from an internal grid of the boundary of the scene, each point was perturbed with random uniform error in the range $[-10^{-3},10^{-3}]$:

The quadrics equations were :

- $X=-40$
- $X=160$
- $Y=-120$
- $Y=80$
- $Z=-70$
- $Z=130$

- $-120X + 40Y - 260Z + X^2 + Y^2 + Z^2 = -14500$

The algorithm was executed for $d = 1$ and $d = 2$. For $d = 1$, a lower bound on the probability of a point belonging to a plane facet is $p = 760/8760$, therefore the algorithm was executed with $k = 12$. The first four stages were sufficient, the following surfaces were reconstructed (cf figure 10):

- $X - 0.0000Y - 0.0000Z = -39.9999$
- $X - 0.0000Y + 0.0000Z = 160.0002$
- $0.0000X + Y + 0.0000Z = -120.0000$
- $0.0000X + Y + 0.0000 * Z = 80.0004$
- $-0.0000X - 0.0000Y + Z = -70.0002$
- $0.0000X - 0.0000Y + Z = 130.0005$

The points assigned to these surfaces were removed from the points of the scene. The algorithm was run on the remaining points with $d = 2$ and $k = 1$, one stage only was sufficient, the following surface was reconstructed (cf figure 10):

- $-120.0000X + 39.9982Y - 260.0174Z + 1.0000X^2 + 0.9999Y^2 + 1.0000Z^2 = -14502.3000$

The angles between the corresponding pairs of hyperplanes are : $\theta_1 = 0.00005273$, $\theta_2 = 0.00016796$, $\theta_3 = 0.00003814$, $\theta_4 = 0.00010266$, $\theta_5 = 0.00014602$, $\theta_6 = 0.00022863$, $\theta_7 = 0.00143803$.

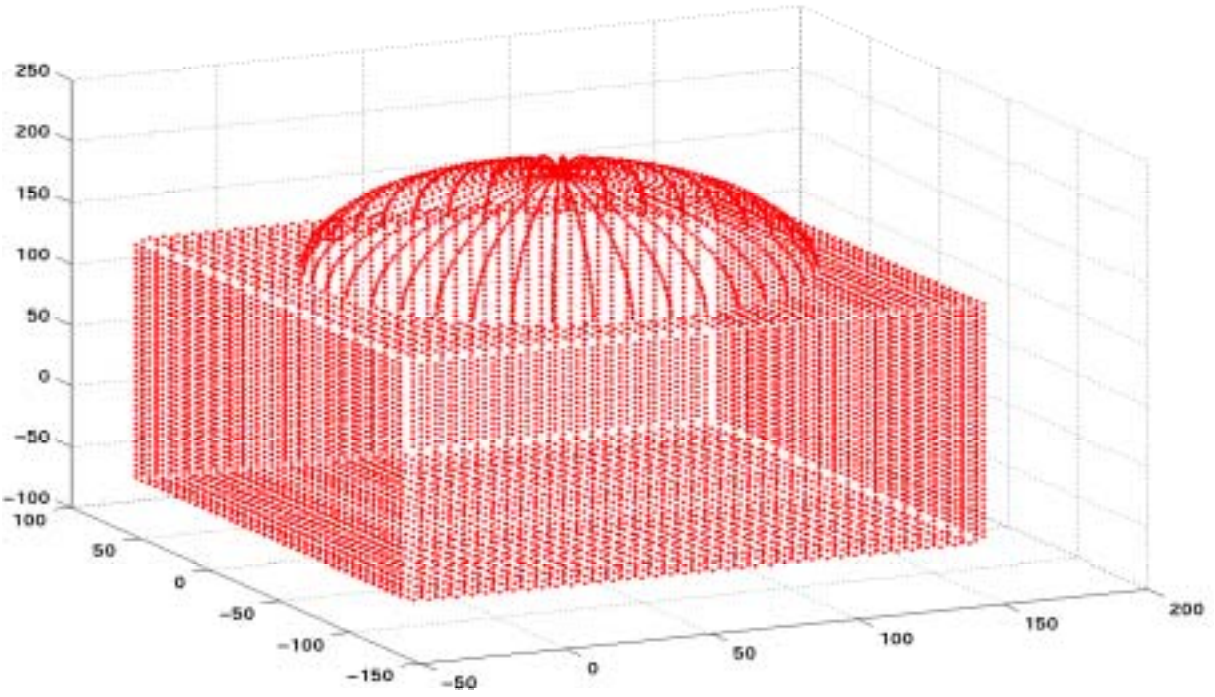

Figure 6: *Hemisphere on top a* $3d$ *cube. 760 points were sampled form the upper cube facet. 1600 points were sampled from hemisphere and from each other cube facet, each point was perturbed with different random uniform error in the range* $[-10^{-3}, 10^{-3}]$.

4 Conclusions

Two new algorithms, for solving the problem of reverse engineering from exact and noisy data of objects defined by algebraic surface patches, were presented. The first algorithm solves the case of data without errors, in an exact manner. Although the complexity of the algorithm is relatively higher than known algorithms, it establishes an interesting approach. The second algorithm is a probabilistic algorithm for both cases with and without data errors. It solves problems which are considered hard to solve and is faster than other known methods, in many cases, it does not assume an exact knowledge of the number of surfaces involved.

The algorithm was tested on several examples and proved to be effective. However there exist cases in which the algorithm reconstructs small amount of spurious surfaces in addition to the original surfaces. These surfaces can be discarded by different postprocessing methods. One

such method is to apply the algorithm several times, and take the intersection of the results, or use proximity and clustering methods to retain the original surfaces solely.

Note that when the error in the data is very large with respect to the objects dimensions, the clustering could lead to “impossible” surfaces such as complex quadrics. In that case the algorithm would not identify some of the algebraic surfaces patches.

The authors are presently working on the extension and complement of the second algorithm , some of the problems in progress being :

(1) Investigate multidimensional clustering methods.

(2) Extending the methods to include boundaries reconstruction of the patches and restore the complete solid B-rep description.

There are many questions brought upon the present work:

How do the final results depend on the *relation* used for comparison, on the clustering method, on the filtering criterion ?

5 Acknowledgements

The authors wish to thank to M. Werman, L. Joskowicz, D. Lischinsky and M. A. Perles for fruitful discussions and encouragement.

References

[1] K.A. Ingle, Reverse Engineering. McGraw-Hill, New York, 1994.

[2] T. Varady, R.R. Martin and J. Cox, Reverse Engineering of geometrical models - an introduction. Computer-Aided Design, 1997, **29** (4), pp.255-268.

[3] R. Chaine, S. Bouakaz and D. Vandorpe, A Graph-Anisotropic Approach to 3-D Data Segmentation. Proceedings of the International Symposium on Computer Graphics, Image Processing and Vision SIBGRAPHI'98, IEEE Computer Society Press, 1998, pp.262-269, IMPA, Rio de Janeiro BRAZIL.

[4] Michel Bercovier, Moshe Luzon, Elan Pavlov, Detecting planar patches in an unorganised set of points in space. Advances in Computational Mathematics, 2002, **17**(1-2), pp. 153-166.

[5] Charles V. Stewart, Robust Parameter Estimation on Computer Vision. Society for Industrial and Applied Mathematics SIAM, **41**(3), pp. 513-537.

[6] P. Roosseeuw, Least median of squares regression. Journal of the American Statistical Association 1993, **58**, pp. 1-22.

[7] M. Fischler & R. Bolles, Random Sample Consensus: A Paradigm for Model fitting. CACM, 1981, **24**(5), pp. 381-395.

[8] J. Illingworth and J. Kittler, A Survey of the Hough Transform. Comput. Vision Graphics Image Process, 1988, **44**, pp. 87-116.

[9] N.Kiryati, Y. Eldar and A.M. Bruckstein, A Probabilistic Hough Transform. The Journal of Pattern Recognition Society, 1991, **24**(4), pp. 303-316.

[10] R.O. Duda and P.E. Hart, Use of the Hough Transformation to Detect Lines and Curves in Pictures. CACM, 1972, **15**(1), pp. 11-15.

[11] Bert Jüttler and Alf Felis, Least-squares fitting of algebraic spline surfaces. Advances in Computational Mathematics, , 2002, **17**(1-2), pp. 135-152.

[12] H. Pottmann, M. Peternell, On Approximation in Spaces of Geometric Objects. The Mathematics of Surfaces IX, R. Cipolla and R. Martin (eds), Springer, 2000, pp. 438-458.

[13] Corman, Rivest, Lieserson, Introduction to Algorithms. The MIT Press Cambridge, Massachussetts London, England. McGraw-Hill Book Company New York St. Louis San Francisco Montreal Toronto, 1990.

[14] Feller, An Introduction to Probability Theory and Its Application. John Wiley & Sons 3rd edition 1, 1968.

[15] POV-Ray software version 3.5, http://www.povray.org/.

The following figures were drawn using POV-Ray software version 3.5 (see [15]).

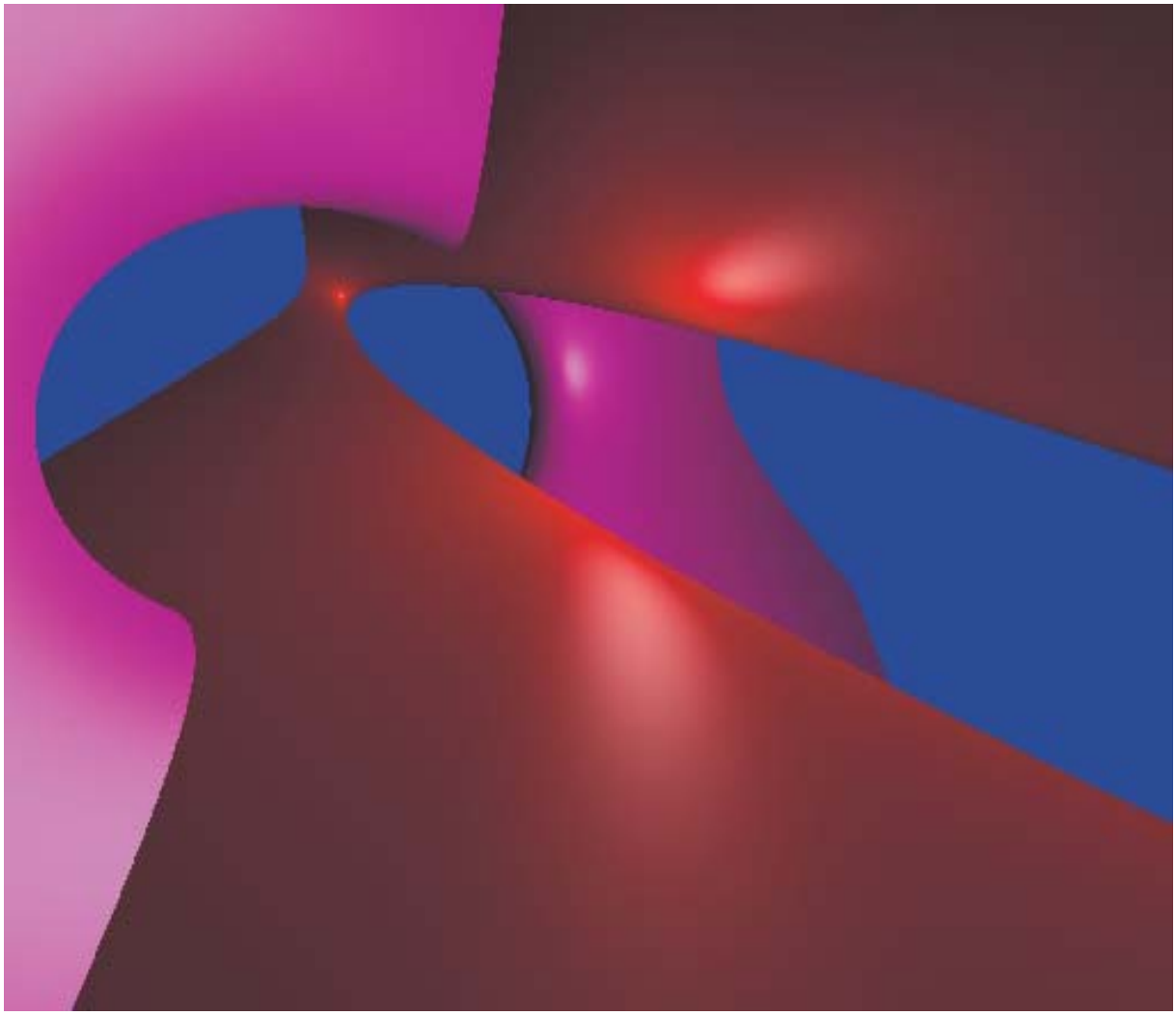

Figure 7: *The two reconstructed surfaces, the first one in purple and the second in red - Example 1.*

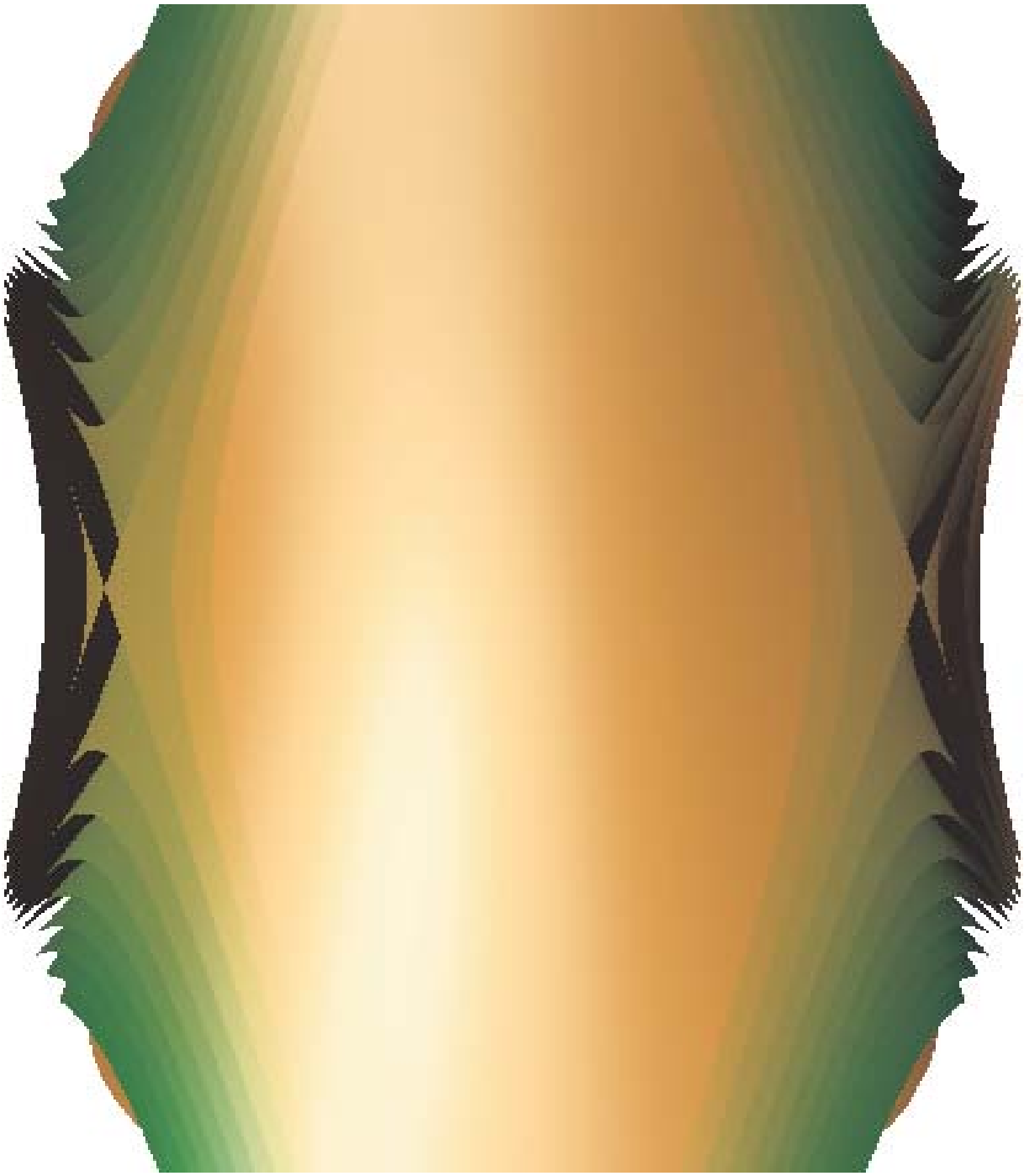

Figure 8: *The reconstructed 12 quadrics - Example 2.*

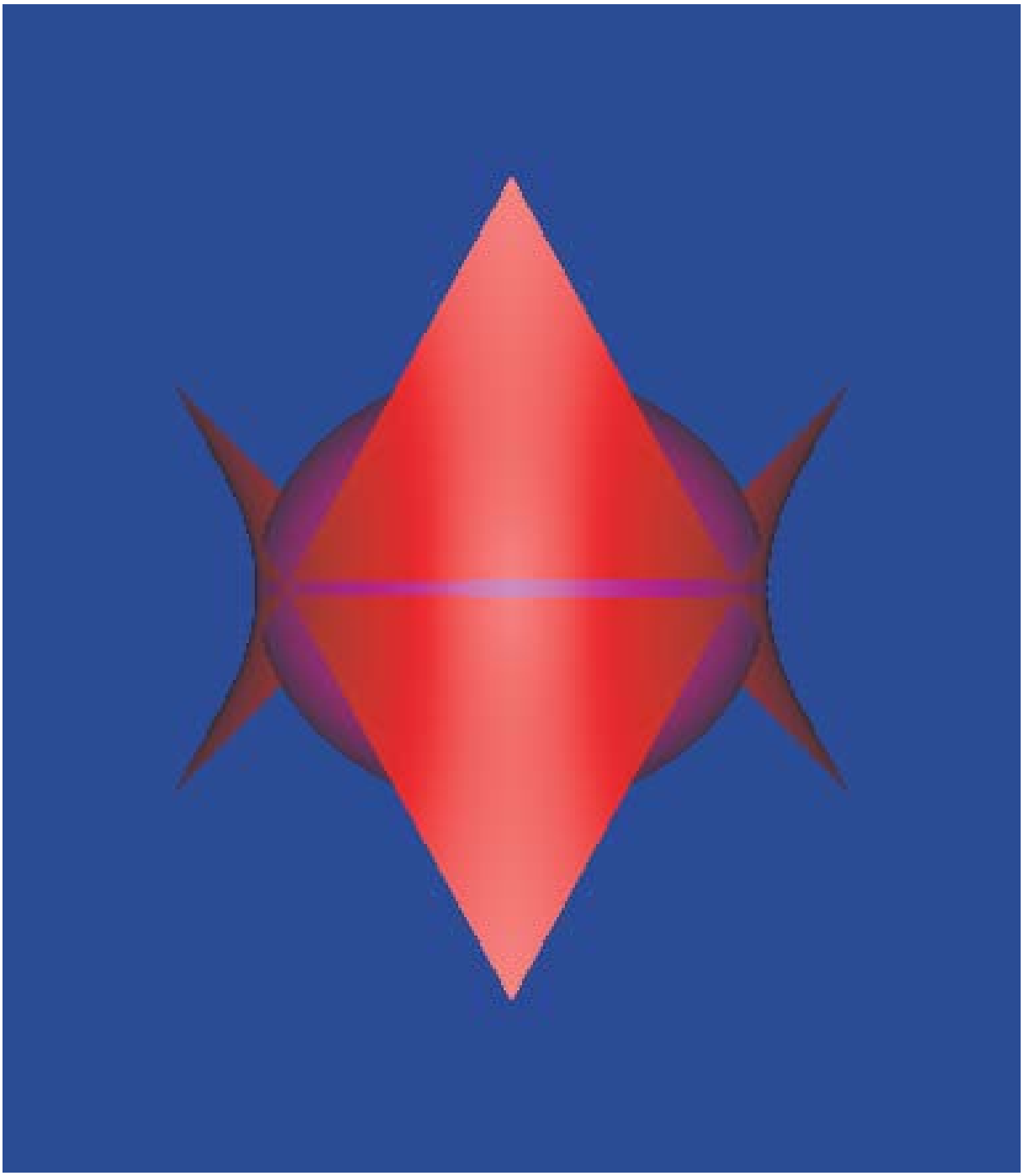

Figure 9: *The reconstructed ellipsoid and one sheeted hyperboloid - Example 3.*

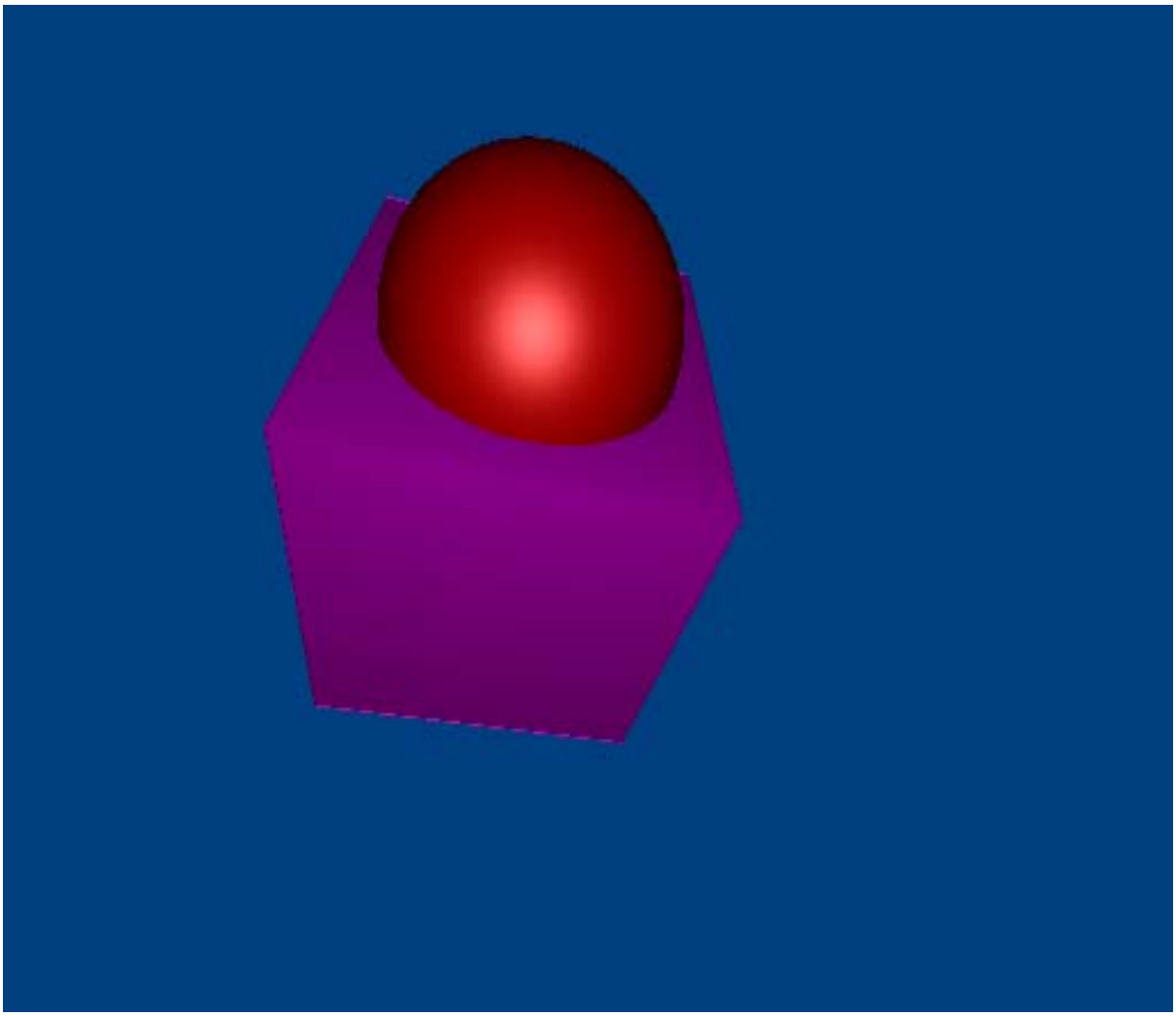

Figure 10: *The reconstructed hemisphere on top of a* $3d$ *cube - Example 4.*