Reconstructing Surfaces from Unstructured 3D Points*

P. Fua and P. Sander

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
USA

INRIA Sophia-Antipolis
2004 Route des Lucioles
06565 Valbonne Cedex
France

Abstract

Most active and passive range finding techniques yield unstructured and generally noisy 3D points. In order to build useful world representations, one must be able to remove spurious data points and group the remaining into meaningful surfaces.

In this paper, we propose an approach based on fitting local surfaces. Differential properties of these surfaces are first used iteratively to smooth the points, and then to group them into more global surfaces while eliminating errors.

We present results on complex indoor and outdoor scenes using stereo data as our source of 3D information.

1 Introduction

To reconstruct object surfaces, one can start with a number of measuring techniques, for example laser rangefinding, stereo or 3D scanners, which all provide raw information about the location of points in space. These points, however, often form potentially noisy "clouds" of data instead of the surfaces one expects.

Deriving the surfaces from such data is a difficult task because:

- the 3D points may form a very irregular sampling of the space,
- they may have been produced by several sensors or derived from several viewpoints so that it becomes impossible to work only in the imaging plane of any one sensor,
- several surfaces can overlap — the 2 1/2 D hypothesis required by simple interpolation schemes is not necessarily valid,
- the sensors and algorithms make mistakes that must be properly dealt with.

In this paper, we address the problem of determining surfaces from a set of points in space, where the points are assumed to be nonregular samples from underlying imaged surfaces. Most existing approaches to this problem, such as deformable superquadrics [Terzopoulos and Metaxas, 1990], modal representations [Horowitz and Pentland, 1991] or 3D deformable surfaces [Cohen *et al.*, 1991], assume that all data points belong to a single object to which a model can be fit. In other words, they assume that the data is already segmented, even though it is well known that segmentation is hard when the data is noisy and originates from multiple objects in a scene.

To overcome this problem, we propose using local 3D surfaces to smooth the data points which are then grouped into globals surfaces. We proceed by fitting a local quadric patch to a small neighborhood of each 3D point and using the estimated surfaces to iteratively smooth the raw data. We then use these local surfaces to define binary relationships between points: points whose local surfaces are consistent are considered as related, i.e., sampled from the same underlying surface. Given this relation, we can impose a graph structure upon our data and define the surfaces we are looking for as sets of points forming connected components of the graph. The surfaces can then be interpolated using simple techniques such as Delaunay triangulation. In effect, we are both segmenting the data set and reconstructing the 3D surfaces.

In the next section, we introduce our fitting procedure (mathematical details are left to App. A) and describe our implementation for real 3D data. We then demonstrate our technique using stereo depth images corresponding to complex indoor and outdoor scenes. Our 3D data is produced by a stereo algorithm that has been developed in previous work [Fua, 1991a; Fua, 1991b], and is briefly described in App. B. Note, however, that closely related methods have also been applied to magnetic resonance imagery [Sander and Zucker, 1990] and laser rangefinder images [Ferrie *et al.*,].

2 From Points in Space to Global Surfaces

Our goal is to determine explicit object surfaces from "clouds" of unstructured 3D points. In this section, we describe the implementation of our method and show how it allows us to go all the way from raw data sets to triangulated maps. The procedure consists of the four

*Support for this research was partially provided by ESPRIT P2502 (VOILA) and ESPRIT BRA 3001 (INSIGHT) and a Defense Advanced Research Projects Agency contract.

steps described in the following subsections.

1. Iterative smoothing of the points by local surface fitting.
2. Resampling and merging of the points on a regular 3D grid.
3. Computation of an adjacency graph and clustering of connected components into surfaces.
4. Triangulation of the clusters.

We take the local surfaces to be quadric because they allow curvature computation while having few enough parameters to allow for a reasonably stable fitting process.

Our method is inherently local and parallel; the procedures have been designed with a SIMD architecture in mind and have been implemented on a Connection Machine.[1]

2.1 Local Surfaces

We fit local surfaces by iteratively fitting quadric patches around every data point and then using them to move the points themselves, as shown in Figure 1.

For our algorithm to be effective with real data, it must be:

- Orientation independent: we do not assume any particular orientation of the surfaces, and the result should not be affected by rotations of the objects of interest.
- Insensitive to outliers: In the neighborhood of any point, the possibility of finding points that are in gross error or belong to more than one surface always exist and must be addressed since least-squares techniques are notoriously sensitive to such problems.

Orientation. To achieve orientation independence around a point $P_0 = (x_0, y_0, z_0)$, we use all the points in a spherical neighborhood and estimate the orientation of the local tangent plane. To initialize, we simply fit a plane; thereafter we use the quadric fit of the previous iteration. Given this orientation, we define a reference frame whose origin is P_0 itself and whose z axis is perpendicular to the plane; we fit a quadric of the form

$$z = \mathrm{quad}(x, y) = ax^2 + bxy + cy^2 + dx + ey + f \quad , \quad (1)$$

by minimizing a least squares criterion δ,

$$\delta = \sum_i w_i (z_i - \mathrm{quad}(x_i, y_i))^2 \quad , \quad (2)$$

where the $(x_i, y_i, z_i)_{1 \le i \le n}$ are the n neighbors of P_0 and the w_i are associated weights. We then transform P_0 as follows:

$$P_0 - - > (0, 0, f = quad(0, 0)) \quad (3)$$

expressed in the local reference frame.

In effect, we are approximating the 2nd order Taylor expansion of the surface around P. Since we iterate the estimates, this procedure is a form of relaxation (see Appendix A) where the amount of smoothing increases with the number of iterations and the size of the neighborhoods. However, because we use quadric patches as opposed to planar ones, the procedure preserves curvature and does not smooth out relevant features. For typical stereo data sets ($\approx$ 100,000 points), the algorithm converges within five to ten iterations.

Outliers. To deal with outliers, we define a metric d_{quad} that measures whether or not two points appear to belong to the same surface. We take d_{quad} to be

$$d_{quad}(P_1, P_2, quad_1, quad_2) = \max(dist_1, dist_2) \quad (4)$$

where

$$\begin{aligned} dist_1 &= abs(z_1 - quad_2(x_1, y_1)) \\ &\quad \text{expressed in the reference frame of } quad_2 \\ dist_2 &= abs(z_2 - quad_1(x_2, y_2)) \\ &\quad \text{expressed in the reference frame of } quad_1 \ . \end{aligned}$$

We depict d_{quad} graphically in Figure 2. It is zero when the two points belong to the same local surface and increases when their respective local surfaces become inconsistent. It can therefore be used to discount outliers by computing the weighting factor w_i of equation 2 at iteration t to be:

$$\begin{aligned} w_i &= \frac{1}{1 + (dist_i/\sigma)^2} \quad (5) \\ dist_i &= d_{quad}(P_0, P_i, \mathrm{quad}_0^{t-1}, \mathrm{quad}_i^{t-1}) \end{aligned}$$

where the $(quad_i^{t-1})_{0 \le i \le n}$ are the quadrics that had been computed at the previous iterations and σ is an estimate of the variance of the process generating the data points. Note that for processes such as stereo compilation or laser range finding, σ can actually be estimated.

In this manner, as the algorithm progresses, the points that are on the same surface as P_0 gain influence while the others are increasingly discounted. The discriminatory power of the algorithm is substantially increased by fitting the quadrics several times at every iteration and updating the weights without moving the data points. We illustrate this behaviour in Figure 3. The two noisy hemispheres are smoothed without being merged and the points between them are left as outliers; such a result would be difficult to achieve with a simple 2 1/2D interpolation scheme.

Implementation. The main obstacle to performing the relaxation smoothing is the very large amount of data that has to be taken into account. A dense 512x512 depth map represents over 250,000 points and as many local surfaces, and the problem obviously gets worse as more and more views are being merged. One solution is to merge the points as they are acquired [Szeliski, 1990]. In this work, however, we are interested more in the behaviour of the smoothing algorithm than in the possible ways of reducing the amount of computation. We therefore start with *all* the data points and assume that, in the z direction, at most a limited number of surfaces can overlap. This is typically true for ground level scenes if z is the vertical direction because only a finite (and usually small) number of objects are stacked on top of one another. Given this assumption, we can

[1] Trademark of Thinking Machines Inc.

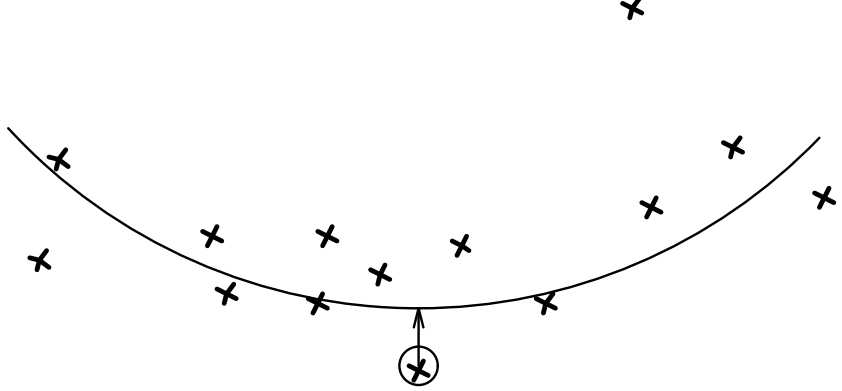
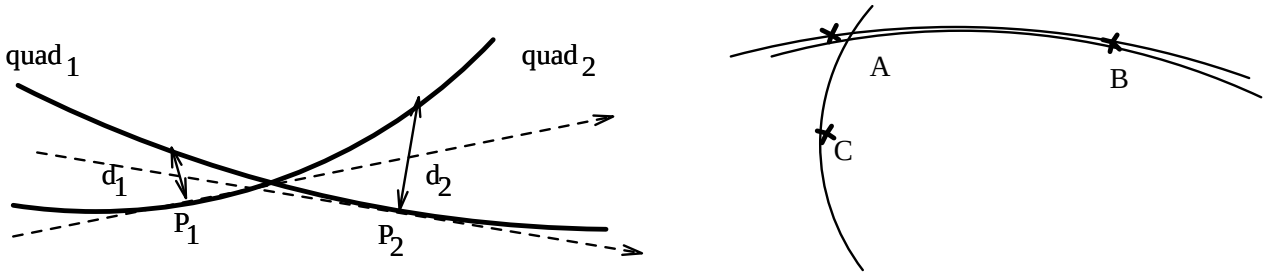


Figure 1: Local surface fitting: the neighbors of the circled point are used to fit a quadric surface onto which the point is then projected.



(a) (b)

Figure 2: (a) The distance between two points is taken to be the maximum value of the distance of one point to the local surface corresponding to the other, represented by the arrows. The dotted lines represent the axes of the reference frames in which the computations are performed. (b) For this metric, point B is "close" to A but C is not, even though their euclidean distances are comparable.

create a cube-shaped data structure by quantizing the x and y coordinate axes and stacking the 3D points with the same quantized x, y values in columns ordered by increasing z values. In this way, we can guarantee that all neighbors can be found in a cubic neighborhood around every point in the data structure; this allows efficient 4-connected (NEWS) access to the neighbors. We can also reduce the z dimension of the cube data structure and make efficient use of the Connection Machine processors.

2.2 Resampling

When the smoothing is done, as shown in Figure 3(c), the data points still form an irregular sampling of the underlying surfaces that is ill suited for the generation of a map. However, to every point is associated a local surface defined by the quad function of equation 1. In order to produce meaningful triangulations, we need a more regularly spaced set of vertices. We therefore pick spatial step sizes δ_x,δ_y and δ_z along an the X,Y and Z axes of an absolute referential. For points whose local surface patch has a normal within 45 degrees of the Z axis, we use the following updating scheme

$$\begin{aligned} x &\quad -- > \quad x_0 = \delta_x \mathrm{mod}(x, \delta_x) \\ y &\quad -- > \quad y_0 = \delta_y \mathrm{mod}(y, \delta_y) \end{aligned}$$

$$z \quad -- > \quad \mathrm{quad}(x_0, y_0).$$

For points whose normals are closer to the X or Y axes, we use the same method but permute the roles of x,y and z. Furthermore, after updating, several points may have two identical coordinates and, provided that their third ones are close enough, be merged. If these points have been seen in different views, we retain that information for later use as will be discussed in Section 3.1. In this manner, points are "resampled" on a regular 3D grid as can be seen in Figure 3(d).

In effect we are replacing a large number of irregularly spaced 3D points by a smaller set of regularly spaced ones and their local surfaces: we are achieving both data organization and compression. This turns out to be an effective way to merge data-points coming from several views or sensors.

2.3 Clustering

To cluster the isolated 3D-points into more global entities, we use again the metric of Equation 4 to define a "same surface" relationship $\mathcal{R}$ as follows:

$$pt_1 \mathcal{R} pt_2 \iff \left| \begin{array}{lcl} d_{quad}(pt_1, pt_2) & < & \max_q \\ d_{\mathrm{eucl}}(pt_1, pt_2) & < & \max_d \end{array} \right. \qquad (6)$$

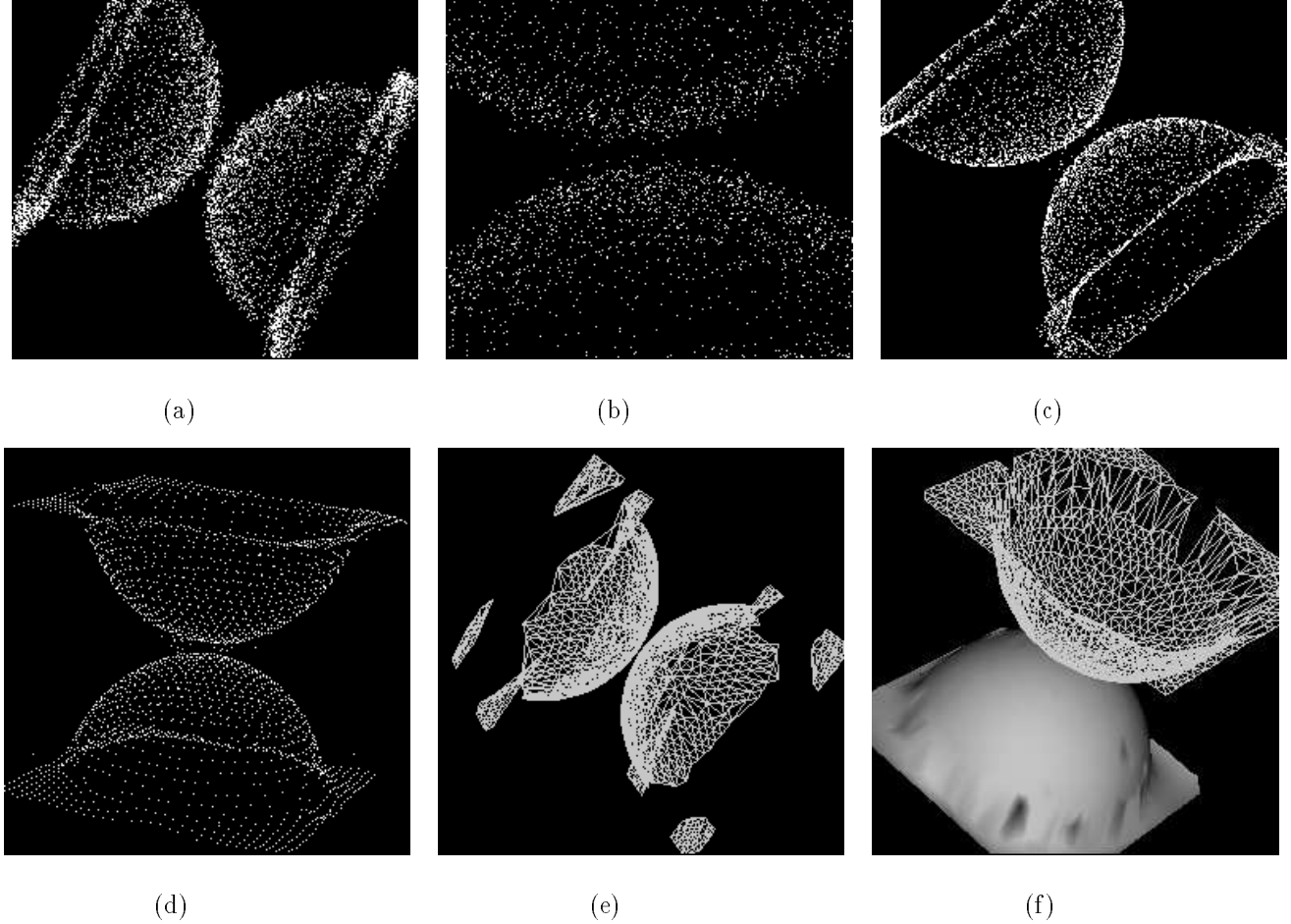


Figure 3: (a) Two superposed and noisy hemispheres. (b) The top of the closest parts of the spheres. (c) Smoothed spheres after several iterations. (d) Resampled points. (e) (f) Two possible segmentations of the data points for two different values of the max_q parameter of Equation 6. In both cases the clusters of points have been triangulated using a 2D Delaunay triangulation in the horizontal plane and in (f) one of them has been shaded.

where d_{eucl} is the euclidean distance max_d and max_q are two thresholds. In other words, two points are assumed to belong to the same surface if their local fits are consistent with one another.

The data set equipped with the relationship $\mathcal{R}$ can now be viewed as a graph whose connected components are the surfaces we are looking for. In practice, there may be erroneous points in the original range data, resulting in situations like the one shown in Figure 4, where legitimate clusters are weakly linked. In such cases, we have found that removing all points that do not have a minimum number of neighbors allows us to throw away the gross errors and generate meaningful clusters. In the case of the two hemispheres of Figure 3, depending on the value the max_q threshold of Equation 6, the algorithm finds either the two obvious clusters or separated the hemispheres themselves and their flat bases because of the sudden change of orientation at the junctions.

2.4 Triangulating

The clusters we have generated so far are collections of points and their associated local surfaces. For many applications, such as robotics or graphics, it is important to be able to unambiguously interpolate the surfaces. Delaunay triangulation is an excellent way of doing this and, furthermore, lets us compute shaded models of our data sets. Alternatively, because our data points are now grouped into sets of points that belong to the same global surface, we could use a number of the other published methods [Cohen *et al.*, 1991; Horowitz and Pentland, 1991; Terzopoulos and Metaxas, 1990].

For applications in which the global surfaces can be projected onto a plane without losing their topology, we do so and use 2D Delaunay triangulation in that plane. This is typically the case for mobile robotics when one needs to model the world as a ground level surface plus obstacles. If this ground level surface has no overhangs, it can be projected onto the horizontal plane. The triangulations of Figure 3(e) and (f) have been computed

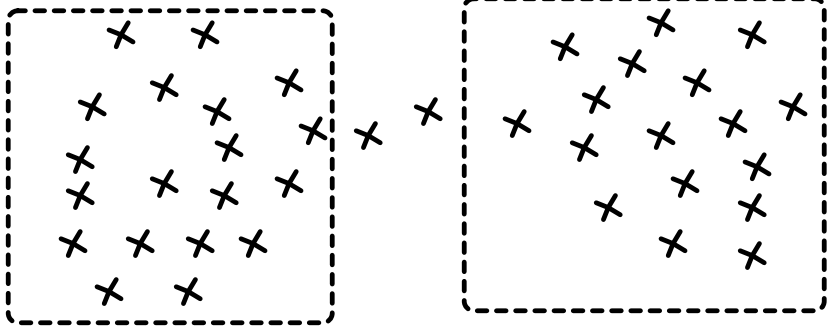


Figure 4: Two legitimate surfaces are can be weakly linked by a few outliers.

in this manner.

For applications where the surfaces can have any orientation, we perform a full 3D Delaunay triangulation that yields the convex hull of the points and use the faces of the tetrahedrons as our surface primitives.

In both cases we retain only the triangles whose orientation is consistent with the local surface patches of the vertices. In the current implementation, we test only the consistency of the surface normals with the triangle orientation, as shown in Figure 2.4.

When the 3D data is sufficiently clean, as will be shown in the results section, this heuristic is sufficient to remove the spurious triangles and "hollow out" the triangulation. However, in more difficult cases, more powerful constraints should be brought to bear:

- Visibility constraint: If a point is visible in a given view but appears to be behind one of the triangles, one of the two must be spurious. Since the resampling operation described in section 2.2 provides such information, it ought to be used.
- Supporting evidence: Given an estimate of the variance of the process generating the points, the σ of Equation 5, one can estimate how much data supports the existence of each triangle.

These extensions will be be the subject of future work.

3 Reconstructing 3D Surfaces from Stereo Data

In this section, we show how our technique can be used to reconstruct 3D surfaces using stereo depth-maps. We first reconstruct natural scenes by merging several maps and then demonstrate the potential of our algorithm on a complex 3D scene containing stacked-up man-made objects.

3.1 Mergeing Results Computed from Several Viewpoints

The stereo algorithm we use [Fua, 1991a; Fua, 1991b] produces semi-dense maps that are 2 1/2D representations of the world. They are necessarily incomplete and, in particular, cannot account for the occluded parts of the scene, such as the back of the rocks in Figure 6(a).

To reconstruct the scene more completely, we have used 5 sets of stereo-pairs corresponding to camera positions between that of figures 6(a) and 6(c). After having registered the stereo results with one another [Zhang and Faugeras, 1990], we can use our technique to generate clusters of triangulated 3D points. The largest one, depicted in Figure 7, accounts for all the large rocks. The erroneous data points have been discarded as outliers.

In Figure 8, we use two stereo pairs to reconstruct the ground surface. Note that the raw points corresponding to the tree and the background wall, as well as the erroneous matches, have been segmented out.

In the two examples above, the depth maps were quite dense and the final surface of interest could be projected onto the horizontal plane to compute a 2D Delaunay triangulation. In the next subsection, we show a more difficult case for which none of these conditions holds.

3.2 Reconstructing a Complex 3D Scene

In Figure 9, we show three objects that have been stacked on a turntable. By rotating the table, we have produced 12 stereo triplets and their corresponding depth maps such as the one of Figure 9 (c). Because the objects are not very textured, the individual maps are neither very dense nor very precise; however by merging them using our smoothing/resampling procedure we can generate the set of points shown in Figure 9(d) and (e) that correctly captures the geometry of the main structures. Unfortunately, there also are a few erroneous points that seem to float in space that must be removed. The shaded models of Figure 10 have been obtained by computing a 3D Delaunay triangulation of the points and retaining only the faces whose orientation is consistent with the local surfaces, as depicted in Figure 2.4. The surfaces corresponding to the three objects are clearly separated but still exhibit a few "holes" due to the lack ot stereo data.

4 Conclusion

In this paper we have presented a method for estimating explicit surfaces from given data consisting of a nonregular sampling of the surfaces. The data was assumed in the form of a general "cloud of points" with no particular structure, and we made no *a priori* assumptions about the form of the underlying surfaces, other than

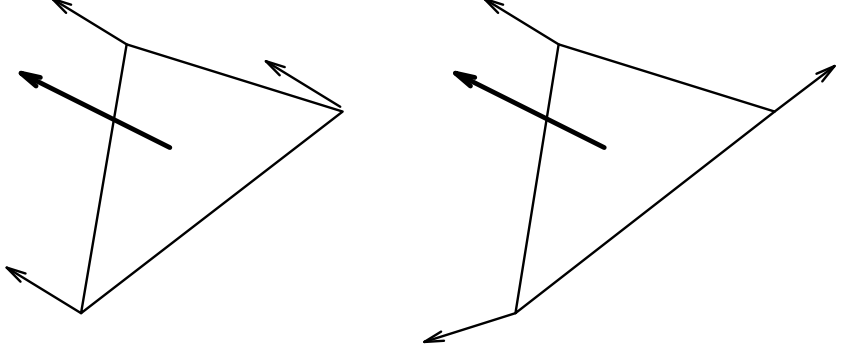


Figure 5: (a) A valid triangle: the normal at the vertices are consistent with the triangle orientation. (b) An invalid triangle.

smoothness. We have shown that we can effectively reconstruct global surfaces under such difficult conditions. However, when additional assumptions or knowledge can be brought to bear, one should obviously do so and our technique can then be used as a source of information about the local surface geometry.

The algorithm consists of four sequential steps: smoothing of the points by iterative local surface fitting; resampling the smoothed points onto a regular grid; computation of an adjacency graph of the points with clustering of the connected components; and triangulation of the clusters. The algorithm is well-suited to parallel implementation and the examples shown were produced on the Connection Machine.

The experiments shown in this paper used depth data acquired from stereopsis by a mobile robot, although we have successfully tested much of the same code on 3D biomedical scanner images as well. It must ne noted that when the quality of the data degrades, the critical step becomes the grouping one. In extreme cases, the heuristics described in this paper are too simple and may fail. To push the nethod forward, it will be necessary to develop more sophisticated ones. In fact, generating the triangulations of section 2.4 could be recast as the problem of finding the best description of the data in terms of a set of triangles knowing the normals and curvatures and every vertex. This problem can be handled within the framework provided by Mininum Description Length encoding [Rissanen, 1987; Leclerc, 1989; Fua and Hanson, 1989]. Such a framework sould provide us with a sound theoretical basis for future work.

Appendices

A The mathematics of recursive fit-and-update

In this appendix, we give some of the details of the method of fitting local surfaces. A comprehensive analysis of the mathematical properties of the algorithm is not given, but will appear in a future publication. Also, to simplify the presentation, we reduce the problem by one dimension and consider fitting local quadratic curves to 2D data points.

Given m initial data points $\{(x_i^0, y_i^0),\ i = 1, \ldots, m\}$, we assume that the iterations k take place on a fixed regular grid so that $x_i^{k+1} = x_i^k = \cdots = x_i^0 = x_i$. We further assume that the fits are local and over the same size neighborhoods of n points so that the neighborhood of (x_i, y_i^k) consists of $\{(-\frac{n-1}{2}, y^k_{i-\frac{n-1}{2}}), \ldots, (0, y_i^k), \ldots, (\frac{n-1}{2}, y^k_{i+\frac{n-1}{2}})\}$. The initial fits to the data points determine coefficients $\zeta_i^0 = (a_i^0, b_i^0, c_i^0)$ of the "best-fitting" quadratic

$$y_i^0(t) = a_i^0 t^2 + b_i^0 t + c_i^0$$

at (x_i, y_i^0).

We now look at the relationship between successive iterations k and $k-1$. The quadratic fit at (x_i, y_i^k) is the least-squares solution of the overdetermined system $A_i^k \zeta_i^k = \beta_i^k$, and for simplicity we consider here that it is given by

$$\zeta_i^k = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \beta_i^k.$$

Note that A_i^k is composed only of linear combinations of x_j and is hence independent of k, and since the $\{x_j\}$ in each neighborhood are the same, it is independent of i as well and we write just A. Thus,

$$\begin{pmatrix} a_i^k \\ b_i^k \\ c_i^k \end{pmatrix} = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \begin{pmatrix} y^k_{-\frac{n-1}{2}} \\ \vdots \\ y^k_{\frac{n-1}{2}} \end{pmatrix},$$

where $(a_i^k, b_i^k, c_i^k) = \zeta_i^k$. Now, the data point y_i^k is updated to $y_i^{k+1} = c_i^k$, and introducing this updating into the above equation gives, at the next iteration,

$$\begin{aligned} \zeta_i^{k+1} &= \begin{pmatrix} a_i^{k+1} \\ b_i^{k+1} \\ c_i^{k+1} \end{pmatrix} = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \begin{pmatrix} y^{k+1}_{-\frac{n-1}{2}} \\ \vdots \\ y^{k+1}_{\frac{n-1}{2}} \end{pmatrix} \\ &= (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} \begin{pmatrix} c^k_{-\frac{n-1}{2}} \\ \vdots \\ c^k_{\frac{n-1}{2}} \end{pmatrix}. \end{aligned}$$

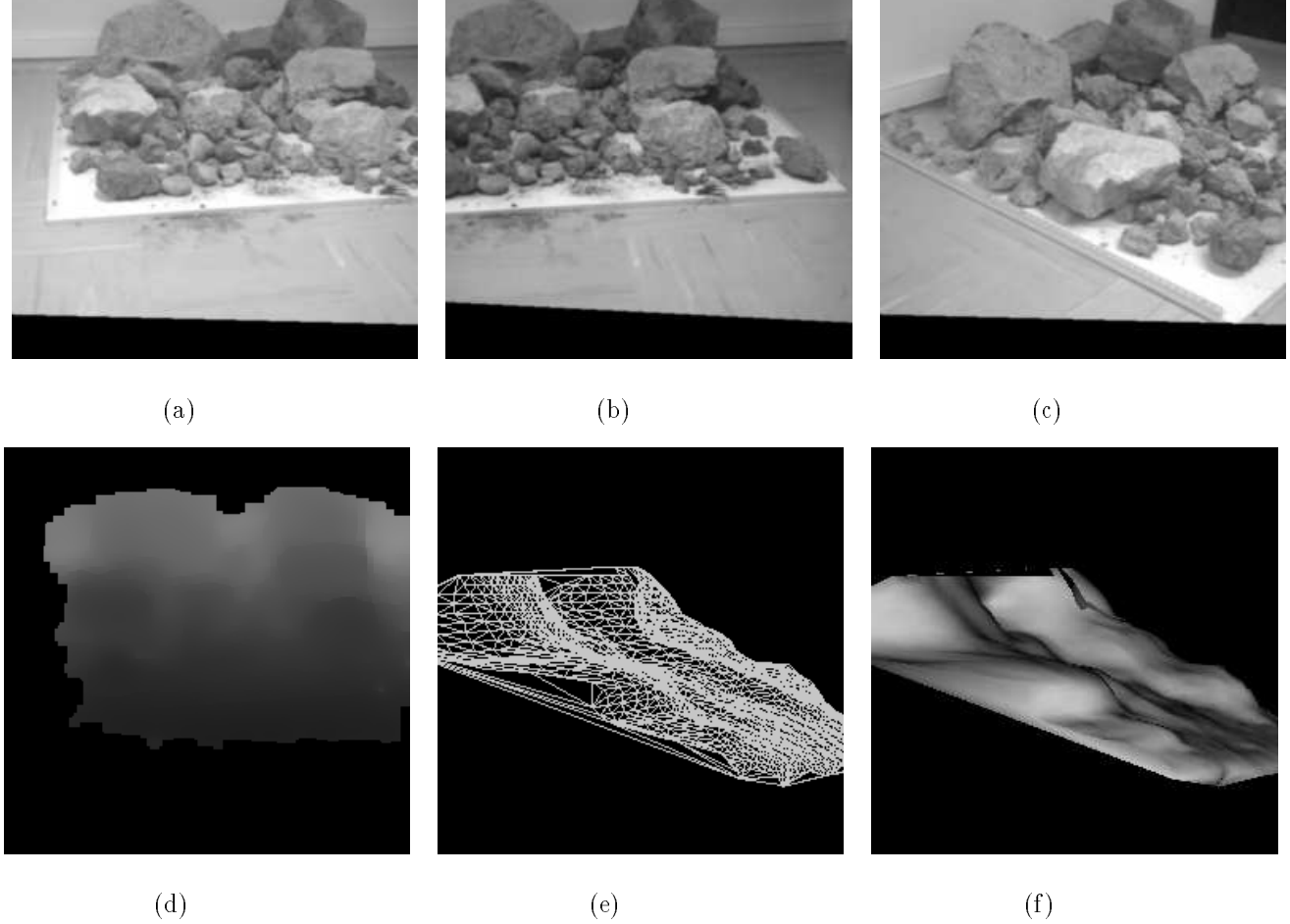


Figure 6: (a) (b) A stereo pair showing a set of rocks. (c) Another image taken from a completely different viewpoint. (d) The stereo map derived by matching (a) and (b). The black areas correspond to textureless areas for which no depth was computed. Elsewhere, the lighter colors correspond to the greater distances (e) Wireframe representation. (f) Shaded representation. Note that the parts of the rocks visible in both (a) and (b) are correctly reconstructed, but that the others are not.

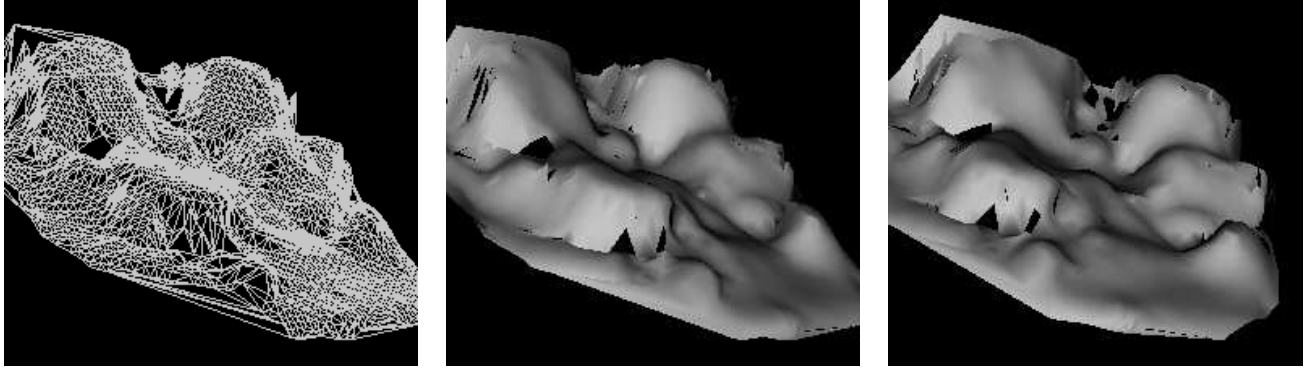
Now, when only the third component of ζ_i^k is considered, we obtain

$$
\begin{aligned}
c_i^{k+1} &= \left\{(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\right\}_{[3,\bullet]} \begin{pmatrix} c^k_{i-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+\frac{n-1}{2}} \end{pmatrix} \\
&= L \begin{pmatrix} c^k_{i-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+\frac{n-1}{2}} \end{pmatrix},
\end{aligned}
$$

where $B_{[3,\bullet]}$ is the third row of a matrix B and L is a $1 \times n$ matrix. Now, we see that considering *all* the data points at iteration $k+1$, we get

$$
\begin{pmatrix} \vdots \\ c^{k+1}_{i-1} \\ c^{k+1}_{i} \\ c^{k+1}_{i+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \ddots & & & & \\ & L & & & \\ & & L & & \\ & & & L & \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ c^k_{i-1-\frac{n-1}{2}} \\ \vdots \\ c^k_{i-1+\frac{n-1}{2}} \\ c^k_{i-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+\frac{n-1}{2}} \\ c^k_{i+1-\frac{n-1}{2}} \\ \vdots \\ c^k_{i+1+\frac{n-1}{2}} \\ \vdots \end{pmatrix},
$$

where each L matrix occupies a $1 \times n$ block of the large matrix. By shifting the L matrices appropriately, the right-hand side can be written more economically as the



(a) (b) (c)

Figure 7: (a) Triangulated ground surface for the rocks of Figure 6 (b) (c) Shaded views. Note that the backs of the foreground rocks of Figure 6(a) are now clearly visible.

product of an $m \times m$ matrix and an $m-$vector,

$$\begin{pmatrix} \vdots \\ c_{i-1}^{k+1} \\ c_i^{k+1} \\ c_{i+1}^{k+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \ddots & & & \\ & L & & \\ & & L & \\ & & & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ c_{i-1}^{k} \\ c_i^{k} \\ c_{i+1}^{k} \\ \vdots \end{pmatrix}$$

$$= \begin{pmatrix} \ddots & & & \\ & L & & \\ & & L & \\ & & & \ddots \end{pmatrix}^{k+1} \begin{pmatrix} \vdots \\ c_{i-1}^{0} \\ c_i^{0} \\ c_{i+1}^{0} \\ \vdots \end{pmatrix}$$

$$= M^{k+1} \begin{pmatrix} \vdots \\ c_{i-1}^{0} \\ c_i^{0} \\ c_{i+1}^{0} \\ \vdots \end{pmatrix},$$

where M is an $m \times m$ banded matrix of width n. Thus we see that with a bit of work the results of the quadratic fit at iteration k can be derived from the results of the initial fit, in turn obtained from the given data observations.

B Area Based Stereography

In this section, we briefly describe the correlation-based stereo algorithm used to produce the 3D data we need. For a more complete description we refer the interested reader to previous publications [Fua, 1991a; Fua, 1991b].

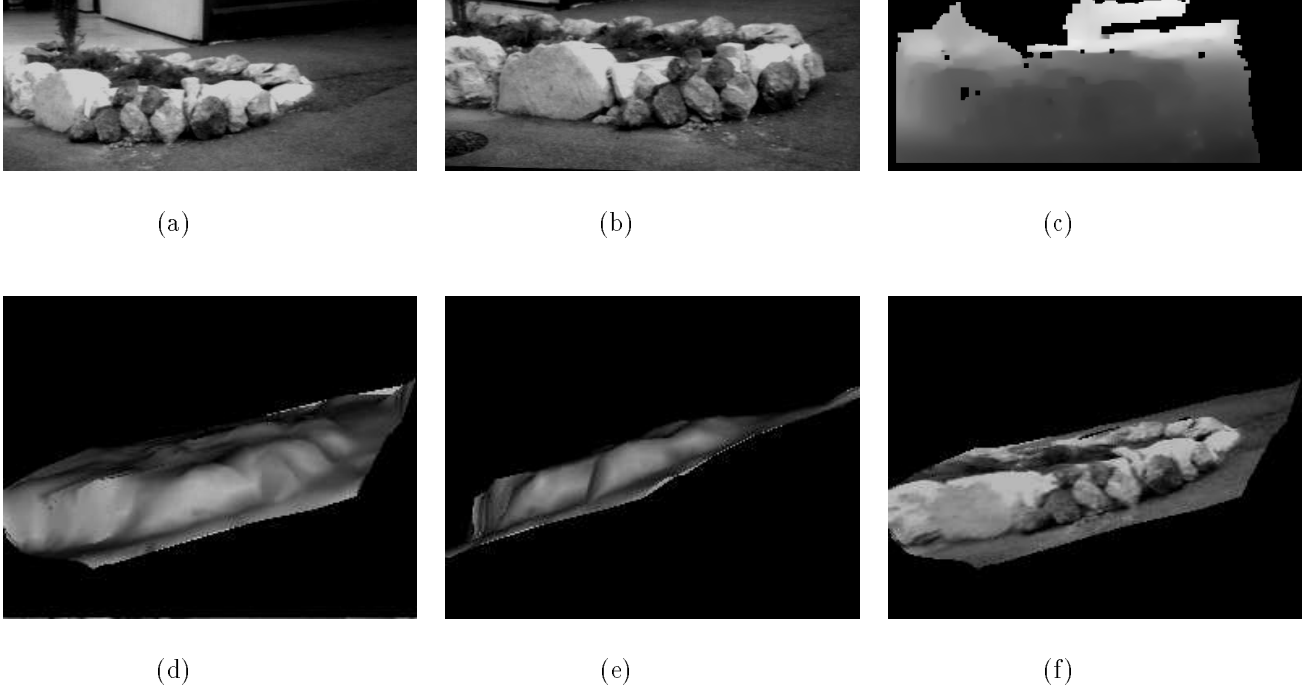
Most correlation based algorithms attempt to find points of interest on which to perform the correlation. This approach is justified when only limited computing resources are available, but with modern hardware architectures and massively parallel computers it becomes possible to perform the correlation over all image points and retain only matches that appear to be "valid."

To generate a dense and accurate depth map, one must then interpolate these measures in such a way as to preserve depth discontinuities. To do so, we model the world as made of smooth surfaces separated by depth discontinuities that generate changes in grey level intensities due to changes in orientation and surface material.

Given a pair of images and corresponding camera models, the computation of the depth map consists of four steps.

1. **Rectification.** The images are reprojected onto the same image plane so that all epipolar lines become parallel. This makes the parallel implementation of the correlation algorithm much simpler because the exact same operations are performed at every pixel.
2. **Matching.** Correlation scores are computed by comparing a fixed window in the first image to a shifting window in the second. The second window is moved in the second image by integer increments and an array of correlation scores is generated for integer disparity values. To compute the disparity with subpixel accuracy, we fit a second degree curve to the correlation scores in the neighborhood of the maximum and compute the optimal disparity by interpolation.

   As shown by Nishihara [Nishihara and T.Poggio, 1983], the probability of a mismatch goes down as the size of the window and the amount of texture increase. However, using large windows leads to a loss of accuracy and the possible loss of important image features. We choose to consider as acceptable only those results for which we get the same result by reversing the roles of the two images, thereby greatly reducing the probability of error even when using very small windows (down to 3x3 for textured outdoor scenes). In fact the density of such consistent matches in a given area of the image appears to be an excellent indicator of the quality of the stereo matching. An occasional "false positive" (a pixel for which the same erroneous disparity is measured when matching both from left to right and right to left) may occur, but we have never encountered a



(a) (b) (c)

(d) (e) (f)

Figure 8: (a) (b) The two left images of a pair of stereo views. (c) The depth map computed using the first one. (d) (e) Two shaded views. (f) A texture mapped view.

situation that gave rise to a large clump of such errors.

3. **Merging across Hierarchy.** We perform the matching at several levels of resolution with identical window sizes, which is conceptually equivalent to matching at one level with windows of different sizes [Kanade and Okutomi, 1990] but computationally more efficient. We then pick the disparity computed at the highest level of resolution for which an acceptable disparity with respect to our consistency test can be found.

   This is a departure from traditional hierarchical implementations that make use of the results generated at low resolution to guide the search at higher resolutions. While these are good methods for reducing computation time, they assume that the result generated at low resolution are more reliable, even if less precise, than those generated at high resolution. This is a questionable assumption especially in the presence of occlusions.

4. **Interpolation.** Finally, the dense depth map w is computed by fitting a piecewise smooth surface to the correlation-based depths $w0$. Assuming that depth discontinuities are more likely to occur where the image intensity gradient is large, we use a conjugate gradient method [Szeliski, 1990; Terzopoulos, 1986] to minimize the following criterion:

$$\mathcal{C} = c(w - w0)^2 + \lambda_x \frac{\partial w}{\partial x}^2 + \lambda_y \frac{\partial w}{\partial y}^2 ,$$

   where c is zero if the correlation has failed and proportional to the normalized correlation score otherwise, and λ_x, λ_y are inversely proportional to the image gradients in the x and y directions. This interpolation scheme produces dense depth maps that preserve depth discontinuities without having to explicitly specify their location.

This algorithm uses simultaneously the grey level information present in a single image and the stereo information present in a pair. In fact, as suggested by Moravec [Moravec, 1981] and many others, more than two images can and should be used whenever practical.

However, the smoothing scheme described above operates in the image plane and makes the 2 1/2D assumption of all such interpolation techniques. In particular, it cannot deal with occlusions and hidden faces. To recover the full 3D structure of objects and merge range data computed from several viewpoints we need to invoke the 3D smoothing procedure of Section 2.

References

[Cohen *et al.*, 1991] Isaac Cohen, Laurent Cohen, and Nicholas Ayache. Introducing deformable surfaces to segment 3D images and infer differential structure. Technical report, INRIA, 1991.

[Ferrie *et al.*,] Frank P. Ferrie, Jean Lagarde, and Peter Whaite. Recovery of volumetric object descriptions from laser rangefinder images. These proceedings.

[Fua and Hanson, 1989] P. Fua and A.J. Hanson. Objective functions for feature discrimination. In *IJCAI-89 Conference*, Detroit, August 1989.

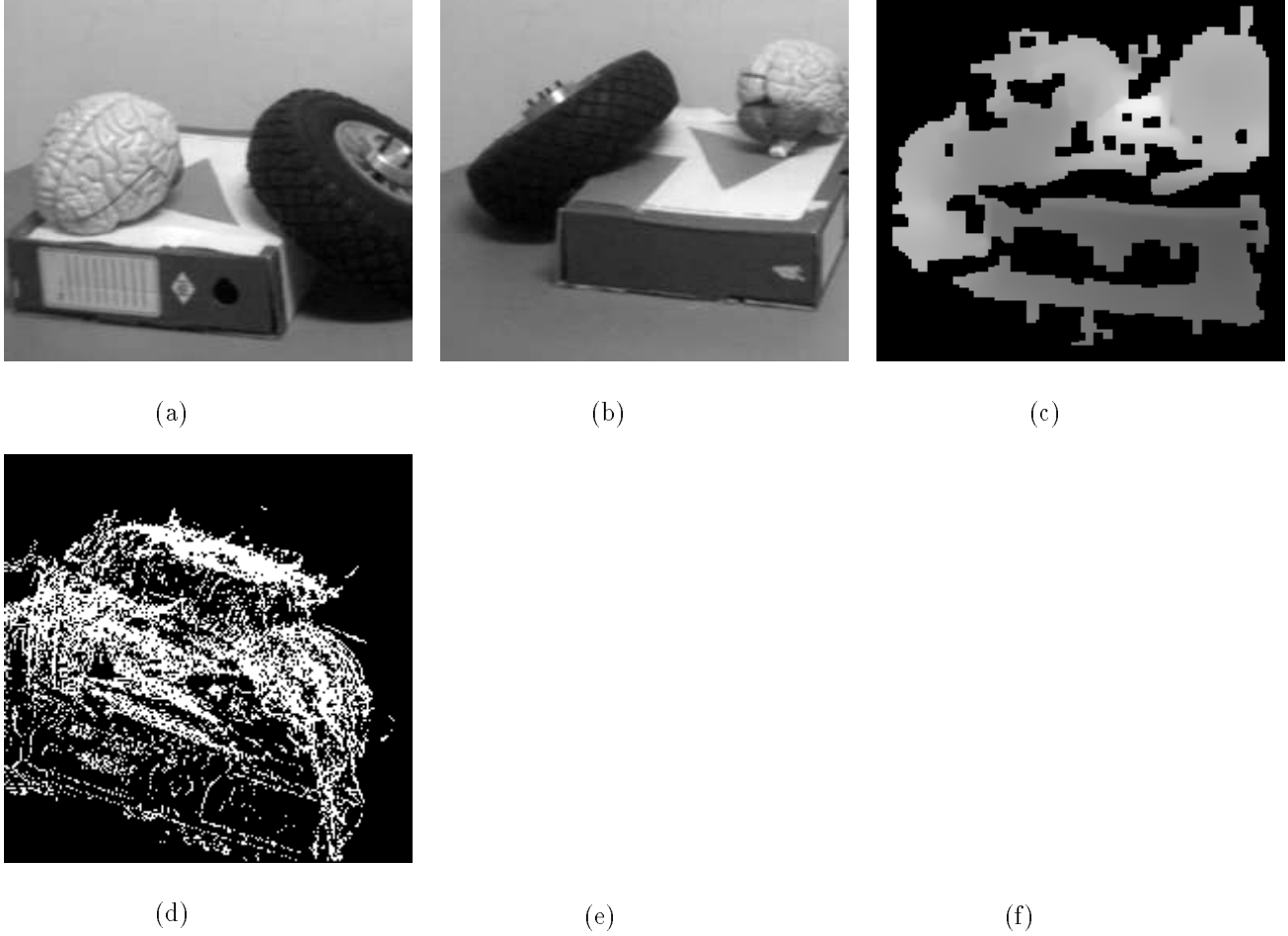


Figure 9: (a) (b) Two diametrically opposing views of three objects on a turntable: a box, the wheel of a robot, and the brain of an unsuccessful graduate student. (c) The depth map corresponding to viewpoint b. (d) The raw stereo data from all views shown as a noisy cloud of points. (e) (f) The smoothed and resampled points shown as a crossed-eye stereogram to be fused by the reader. Note that the vertical sides and the horizontal top of the box, the wheel, and the brain are correctly recovered.

[Fua, 1991a] P. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *IJCAI Conference*, Sydney, Australia, August 1991.

[Fua, 1991b] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 1991. Accepted for publication.

[Horowitz and Pentland, 1991] Bradley Horowitz and Alex Pentland. Recovery of non-rigid motion and structure. In *Proceedings of CVPR'91*, pages 325–330, Hawaii, June 1991.

[Kanade and Okutomi, 1990] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptative window: Theory and experiment. In *Image Understanding Workshop*, September 1990.

[Leclerc, 1989] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102, 1989.

[Moravec, 1981] H. Moravec. *Robot Rover Visual Navigation*. UMI Research Press, Ann Arbor, Michigan, 1981.

[Nishihara and T.Poggio, 1983] H.K. Nishihara and T.Poggio. Stereo vision for robotics. In *ISRR83 Conference*, Bretton Woods, New Hampshire, 1983.

[Rissanen, 1987] J. Rissanen. Minimum-description-length principle. *Encyclopedia of Statistical Sciences*, 5:523–527, 1987.

[Sander and Zucker, 1990] Peter T. Sander and Steven W. Zucker. Inferring surface trace and differential structure from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(9):833–854, September 1990.

[Szeliski, 1990] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transac-*

(a) (b) (c)

Figure 10: The triangular faces of 3D Delaunay triangles whose orientations are consistent with the local surface patches. (a) Texture mapped view. (b) (c) Shaded views. The texture maped view is obviously more realistic but also much less revealing of the actual performance of the algorithm. Note also, that the algorithm leaves holes where there was not enough stereo data to build reliable triangles.

tions on Pattern Analysis and Machine Intelligence, 12(6):513–528, June 1990.

[Terzopoulos and Metaxas, 1990] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. In *Proceedings of ICCV'90*, pages 606–615, Osaka, December 1990.

[Terzopoulos, 1986] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):129–139, March 1986.

[Zhang and Faugeras, 1990]
Z. Zhang and O.D. Faugeras. Tracking and motion estimation in a sequence of stereo frames. In L.C. Aiello, editor, *Proc. 9th European Conf. Artif. Intell.*, pages 747–752, Stockholm, Sweden, August 1990.