Surface Fitting for Industrial Applications

Brian G. Schunck, Seth O. Rogers, Sarvajit S. Sinha

Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109-2110
srogers@engin.umich.edu

Abstract

In the use of laser scanning devices for inspection, metrology, and reverse engineering tasks, the accuracy of the data is of primary importance. To compensate for the error characteristics of such devices, we employ a two-stage algorithm fully presented in [7] for reducing the error and fitting a densemathematical surface representation to the data. We performed a number of tests with the algorithm on both synthetic and actual data, and we show that the error is significantly reduced. We also varied the parameters for the algorithm to minimize the error given the error characteristics of the laser scanner.

1 Introduction

This report presents and tests algorithms to create a robust surface model of an object utilizing data input from a ranging device. There is a large variety of laser scanning devices, with differing error characteristics and output formats [1]. Each scanner has a different ratio of Gaussian to non-Gaussian distributed noise (outliers, or a broad-tailed distribution), and a different standard deviation for the Gaussian distribution. Some common output formats are scattered, line, and gridded. Scattered data are randomly sampled points on the object surface, line data are points sampled unevenly along a number of curves on the surface of the object, and gridded data are points sampled on a regular grid. See Figure 1 for illustrations of the scattered, line, and gridded formats. All of these formats have the effect of creating a cloud of points located on the object surface. To maximize the utility of this technology, there must be a way to eliminate both the normal errors and the outliers and arrive at a dense canonical mathematical representation of the surface of the scanned object.

Our claim is that *the accuracy of laser scanning devices can be greatly improved by combining it with a surface approximation algorithm.* In fact, the sensor should not be viewed in isolation, but only as a part of the combination of sensor hardware and surface fitting software. In this report, we experiment with parameters of our model, and show that the accuracy does indeed increase when the raw data is filtered through our algorithms. The major advantages of our approach are the increase in sensor accuracy and a reduction in the data volume due to the approximation. We also specifically analyze the error characteristics of the Perceptron laser scanner and compare it to our theoretical experiments. In our tests with the Perceptron scanner, we were able to reduce a 154K data file containing 11,060 points to a 17K spline surface file containing 1024 knots, with an error reduction of more than one third.

The software we have developed accepts raw data from a laser range scanner, with the type of errors and any of the output formats mentioned above, and fits it using a two stage algorithm to get a dense representation of the data. The file formats are described in Appendix A.

1.1 Algorithm overview

The algorithm uses a combination of local least median of squares regression as the first stage and an approximating weighted bicubic spline for the sec-

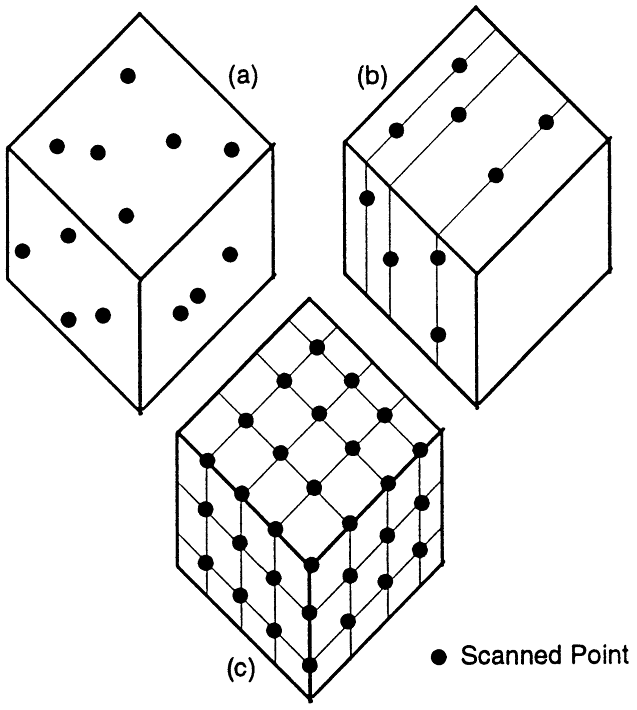


Figure 1: Descriptions of the three common laser range scanner formats: (a) Scattered, (b) Line, (c) Gridded.

ond stage. In our current implementation of the two-stage philosophy, we have dealt with the problems that arose from an early implementation [8]. We do not simply use least-squares for approximation. The assumption behind least-squares fitting is that the noise in the data has a Gaussian distribution. We found our input data contains outliers (due to sensor errors) and discontinuities due to the physical nature of the scene. These violate the assumptions of normally distributed noise. An approximating spline in the second stage also assumes that the errors are normally distributed, whereas again this is not the case because of the outliers in the data. This leads us to the use of a *robust* Least Median Squares based algorithm for fitting in the first stage, and an *approximating* spline in the second stage. The first stage eliminates outliers and grids the data, while the second stage reduces noise from a Gaussian distribution. The approximating spline presented in this report has little relationship to the interpolating spline in [3, 8], except for the fact that a weighting term is used in both. We have been able to use our experience with the adaptive weights in [8] to create a weighting function in our new formulation.

2 Background

In the past, inspection, metrology and reverse engineering projects have utilized coordinate measuring machines (CMM) in order to digitize a physical model. This device is an accurate robotic arm with a touch-probe sensor attached to its wrist. The device is either manually probed over the surface, or programmed to follow a particular path. In either case the process is laborious and time-consuming. Another disadvantage of the CMM is its price. However, the one great advantage of the CMM is its accuracy— a typical CMM has an accucracy of the order of 10 microns.

A different approach involves using an "optical" CMM (laser plane-of-light triangulation or laser radar) to obtain surface points. This is a cheaper sensor, and measurements over the whole object can be obtained in a few frames. As the laser scanner operates by line-of-sight, an important consideration is that the points generated will all belong to a surface which will not bend behind itself with respect to the direction of the scanner. Therefore, by definition, the points belong to a *graph surface*. This allows us to assume that all points in a certain neighborhood lie on the same surface, and can be processed together. The disadvantage of the laser scanner is that each individual point has less accuracy than a mechanical CMM. Figure 2

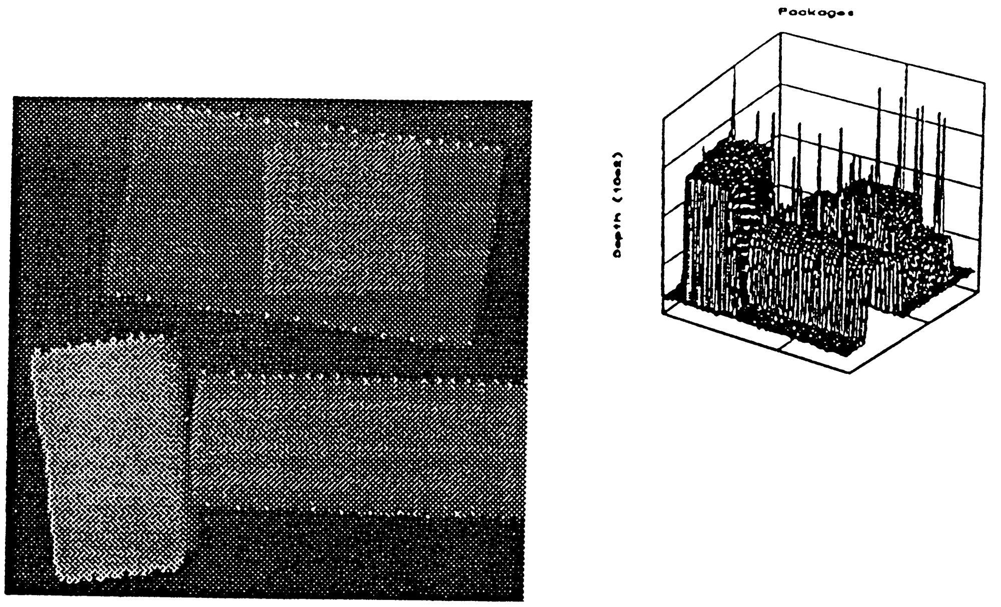


Figure 2: A real range image of postal packages on a tabletop, with noise (outliers) at the discontinuities caused by sensor errors.

shows the data obtained from such a sensor. As is apparent, there are three problems that arise— (a) Outliers, or data points not obeying a Gaussian distribution, occur due to sensor errors, (b) Data dropouts occur due to light scattering from the metallic object, especially near the edges, where the slope becomes large, (c) Gaussian noise caused within the sensor, due to its electronics and software. The accuracy of the system is of the order of 150 microns. Another source of error is due to the polynomial interpolation used when transforming coordinate systems between the pixel space to real sensor space, and finally to a world coordinate system.

Our solution to these problems is to use a two stage algorithm to first grid the data and remove outliers, and second fit the data to a discontinuity preserving spline surface to eliminate the Gaussian errors and create a dense representation of the data. This report describes and tests both stages of the surface reconstruction algorithm.

2.1 The first stage

The first stage is categorized as a moving least median squares operation, which grids and cleans the range data. This means the raw data can be transformed from a scattered or line data format to a regular grid of any density, and any outliers in the raw data can be removed. The first stage works by imposing a rectangular set of grid points on the input point cloud. The range value for each grid point is calculated from the neighborhood or window of surrounding points in the following manner: Every possible combination of three points in the neighborhood is examined (For a more efficient approximation, the algorithm can operate with only a certain maximum number of combinations.). For each combination, a plane is fit through all three points, and the error for all points in the neighborhood is calculated. The errors should fall into three categories: (a) Zero errors for the points used to calculate the plane, (b) High errors for the outliers with respect to the points in category (a), and (c) Normally distributed errors for the points on the plane. We then associate an error with the plane fit, and continue until all combinations of points have been examined. Instead of using a mean of the point errors for the fit error, we chose to use the median error. This median error measures the accuracy of the planar fit to a "typical" data point in the neighborhood of the grid point, ignoring both the well-fit points such as the ones used to calculate the plane, and the poorly-fit points such as outliers, as long as no more than 50% of the neighboring points are in one of these categories. The plane for which the median error is minimized is used to calculate the range value for the grid point.

The Least Median of Squares algorithm allows us to choose a plane for each grid point which best fits the majority of neighboring points, ignoring the points which are not well fit. Note that it is also possible to fit any surface through the sample points, such as polynomial patches [2]. Up to 50% of the points are ignored in this fashion, allowing the algorithm to perform with up to 50% outliers.

The first stage also preserves discontinuities as an attribute of its median filtering nature [4]. If a neighborhood contains a discontinuity (i.e. two regions with different surface equations), the points in the smaller region will be considered outliers and the plane will be fit to points in the larger region.

2.2 The second stage

The second stage of the algorithm uses the gridded and cleaned result of the first stage to compute a smooth spline fit to the data. The second stage is based on the regularized solution to the surface reconstruction problem [10, 11], minimizing

$$\mathcal{J}(f) = \mathcal{E}(f) + \nu^2 \mathcal{S}(f) \tag{1}$$

where $\mathcal{E}$ is an error term and $\mathcal{S}$ is a smoothness norm (The function is smooth.). In 1-D, $\mathcal{S}$ is chosen to be $\mathcal{S}(f) = \int f''^2(x)dx$. The square term in the integrand can be thought of as the "energy" of the curve, and is an approximation to the curvature

$$\kappa = \frac{f''}{(1 + f'^2)^{\frac{3}{2}}} \tag{2}$$

of the curve for small f'. Minimizing the global measure $f''(x)$ gives equal weight to all data. The modification we have introduced is to weight the quadratic smoothness norm with an adaptive measure $w(x, y)$ of the variation in the surface. For rapidly varying data, such as an image with discontinuities, the smoothness norm should be small over flat regions and large over steep ones. The weighting function $w(x, y)$ is chosen such that it is large when the data is flat and small when the data is steep. This allows $\mathcal{S}(f)$ to be smaller in the flat regions, which implies the surface $f(x)$ can be smoother. In the steep regions, $\mathcal{S}(f)$ can be large, allowing the surface to vary or "bend" rapidly. The general form of these norms (or stabilizers) is derived by Tikhonov and Arsenin [12]. Based on their general form, we obtain in 2-D the approximating surface $f(x, y)$ by minimizing the functional

$$\mathcal{J}(f) = \sum_{i=1}^{N} (z_i - f(x_i, y_i))^2 + \\ \int\int_{\Omega} w(x, y) \left(f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2\right) dxdy \tag{3}$$

where N is the number of data points. Here, the smoothness functional measures the energy of a thin-plate under small perturbations. The stabilizer is necessary because our experiments with regression splines alone (without the integral term) gave poorly-conditioned results.

The surface model used for the fitting procedure in the second stage algorithm is the adaptive thin plate spline [6]. The approximating spline is obtained by minimizing expression 3. The 2-D adaptive spline is given as a linear combination of the tensor products of 1-D B-splines.

$$f(x, y) = \sum_{i=1}^{g+k} \sum_{j=1}^{h+k} c_{ij} b_{ki}(x) b_{kj}(y) \tag{4}$$

$b_{ki}(x)$ and $b_{kj}(y)$ are 1-dimensional B-splines of degree $k-1$, $b_{kj}(y)$ is described in the same manner as $b_{ki}(x)$, and g and h are the number of knots in x and y. We assume that the basis functions are cubic (order $k=4$).

The weighting functions are chosen to be

$$w_x(x_i, y_j) = \left[\rho(1 + v_x^2 + v_y^2)\right]^{-2} \tag{5}$$

$$w_y(x_i, y_j) = \left[\rho(1 + v_x^2 + v_y^2)\right]^{-2} \tag{6}$$

where

$$v_x(x_i, y_j) = \frac{|z_{ij} - z_{i+1,j+1} + z_{i,j+1} - z_{i+1,j}|}{\frac{1}{2}(\Delta x + \Delta y)}$$

$$v_y(x_i, y_j) = \frac{|z_{ij} - z_{i+1,j+1} - z_{i,j+1} + z_{i+1,j}|}{\frac{1}{2}(\Delta x + \Delta y)}$$

We use a modified version of Inoue's technique [5] for solving the problem. B-Splines are used as the finite element in the solution instead of the quadratic element. Inoue formulates the problem as a set of N linear equations in N unknowns. Solving this using standard elimination techniques is computationally intractable, so instead Inoue constructs an iterative solution. After the initial solution is computed by elimination, the next step doubles the number of knots in each direction, then interpolates to find a new approximation. The new coefficent matrix and value vector are calculated, and a new solution is obtained via a successive overrelaxation method. This solution then can be used in the next step of the iteration.

2.3 Parameters

Besides the availability of three data formats, the two-stage algorithm has a number of user-adjustable parameters that gives it a range of behaviors. Besides parameters controlling the number of grid rows and columns, there are a number of parameters for the first stage involving the window used to find the plane for each grid point. The size of the window can be adjusted, and the maximum and minimum number of points in the window can be specified. If the minimum number of points is not found in the window due to the sparseness of the data in that area, there are a number of search options that can be employed. The window search can be immediately abandoned, the window can be grown to a maximum size, or until a certain number of points is found in the window. If the chosen search fails, a user-defined background value is associated with the grid point. It is also possible to choose how many samples will be chosen from the points in the window to find the plane fit.

The first stage also performs some initial data filtering tasks. The input data can be scaled in the x, y, and z dimensions, and for line data a "graphize" filter can be employed to remove points that seem to conflict with the graph surface assumption. This filter works by assuming that coordinates in the x and y axes are monotonically increasing. So, if any value is found that is not a new maximum for that axis, it is removed. The filter only works with line data because gridded data is implicitly a graph surface and more assumptions must be made for scattered data.

The second stage has four parameters influencing the shape of the surface fit. The most important parameter in our experiments has been ρ, the roughness parameter. For a low ρ, the spline fit is very smooth. As ρ increases, the spline fit is allowed to fit the data more and more closely, causing a rougher appearance due to the noise in the data. The tension parameter τ represents the amount of energy pulling the sides of the thin plate. Finally, the third and fourth parameters define the number of knots in the x and y directions, which affects the level of detail in the fit.

The second stage also has two parameters named unit length l_u and unit observable d_u, which normally stay constant at 0.1 to allow the other parameters to maintain their intended effects.

See Appendix B for the parameters to the subroutines used for the algorithm.

3 Testing methodology

The experimental testing of the two stage algorithm was designed to measure the error reduction achieved by the bicubic spline compared to the original point cloud. Because the input data may have different levels of noise, one of the objectives of the algorithm was to find the best possible solution given the Gaussian standard deviation. This was accomplished by varying the smoothness of the spline fit. If the data is noisy, the spline must smooth out the surface to eliminate the Gaussian peaks and valleys. If the data has little noise, the spline can fit the data more closely. The parameter which determines where the spline is located on the loose to tight fit continuum is denoted ρ.

There were two experiments performed: (a) Synthetic data, and (b)

Actual data. In each experiment, several spline fits were performed with different ρ values to determine empirically what ρ value is optimal for a sensor with known error characteristics. Error measurements were taken in two regions: the continuous planar region where the fit was expected to be most accurate, and the transition region between the two surfaces, which was expected to have a larger error since the spline surface smooths across step discontinuities. Because of Gibbs phenomena (a common artifact causing "ripples" near the boundary of a surface) on the edges, the continuous region for the actual data set is taken from the interior.

The synthetic test model was a cube on a plane viewed from directly above. Several test images were created of this model simulating errors of 0, 2, 5, 10, and 15 units standard deviation (σ). The images were created by first generating a background of constant height, then adding the cube into the middle of the image 50 units above the background. Finally, the Gaussian-distributed noise was added to the entire image by adding $\sigma \cdot n$, where n comes from a Gaussian distribution with a unit standard deviation, to each value. The model is shown in Figure 3 (a) with no noise and (b) with a σ of 5. The continuous and transition regions are marked. Since the images were already gridded and contained no outliers, the first stage was unnecessary and instead the second stage was directly applied with a 32 by 32 knot grid. For each test image, the spline fit used values of 10, 50, 100, 500, and 1000 for ρ. Figure 4 presents the resulting spline fits for (ρ, σ) values of (10,0), (10,10), and (100,0).

The actual data was taken from a Perceptron laser range finder with an accuracy of 20 microns in the z dimension. The accuracy in the x and y dimensions is unimportant because the first stage uses a neighborhood of points to calculate each grid point value. The object was a rectilinear block on a flat support. The scanner was directly above one edge of the block. The geometry of the scan is depicted in Figure 5. The scan created a data set consisting of two horizontal planes separated by a discontinuity. The data is organized into lines, with a number of points scattered along each line. The original data is shown in Figure 6, rotated to make it viewable. The equation for the top plane is $0.002293x + -0.089056y + z + -224.711655 = 0$ and the equation for the bottom plane is $0.00190x + -0.206307y + z + -1.779542 = 0$. The data was first "graphized" and scaled to match the synthetic data, then the first stage was applied to grid and clean the data. A 64 by 64 grid was chosen. For each grid point, the least median squares operator used a window two pixels in diameter to search for neighbors. If not enough points were in the window, the window was expanded, continuing until the

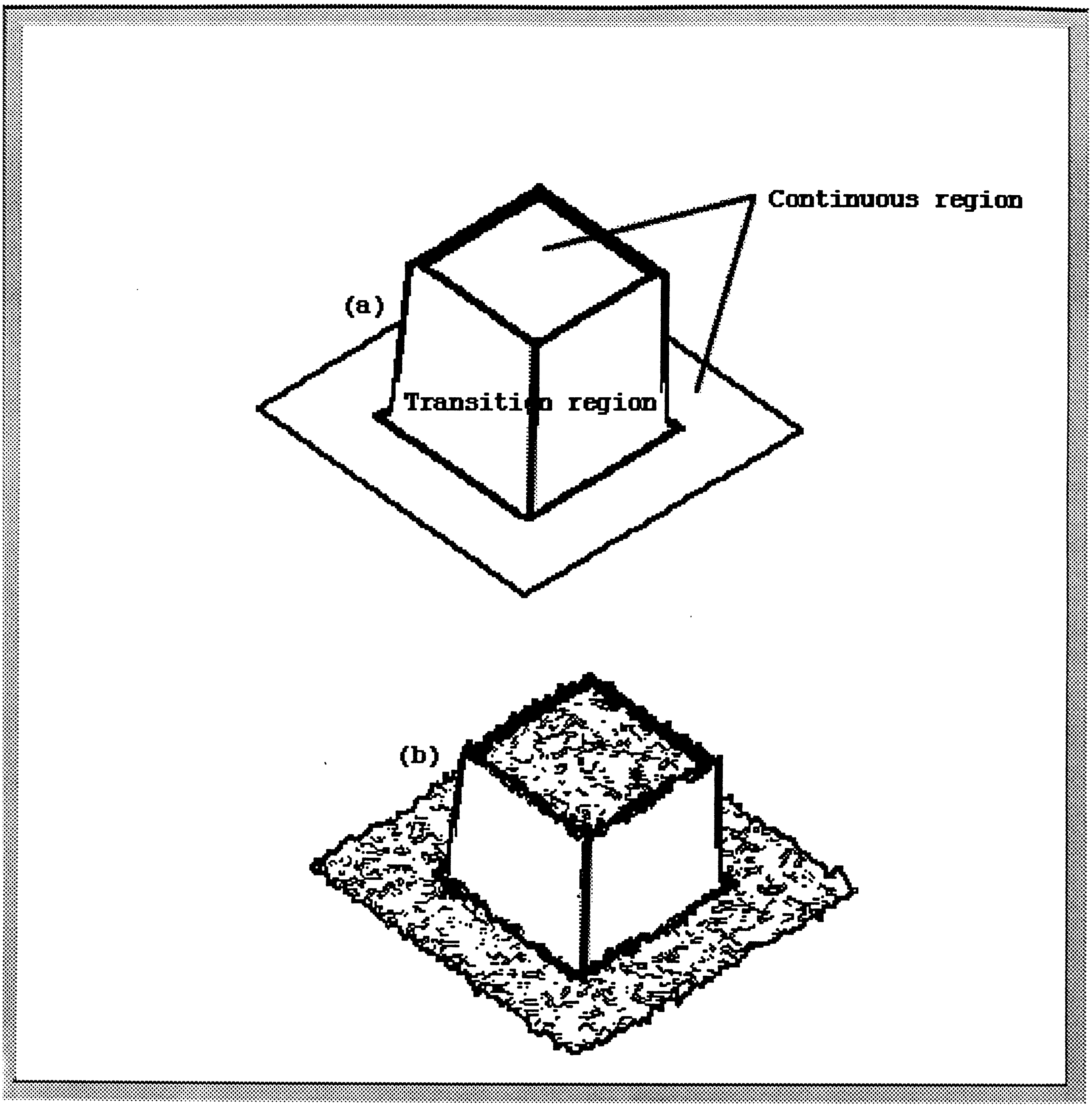


Figure 3: Original synthetic data, (a) $\sigma = 0$, (b) $\sigma = 5$.

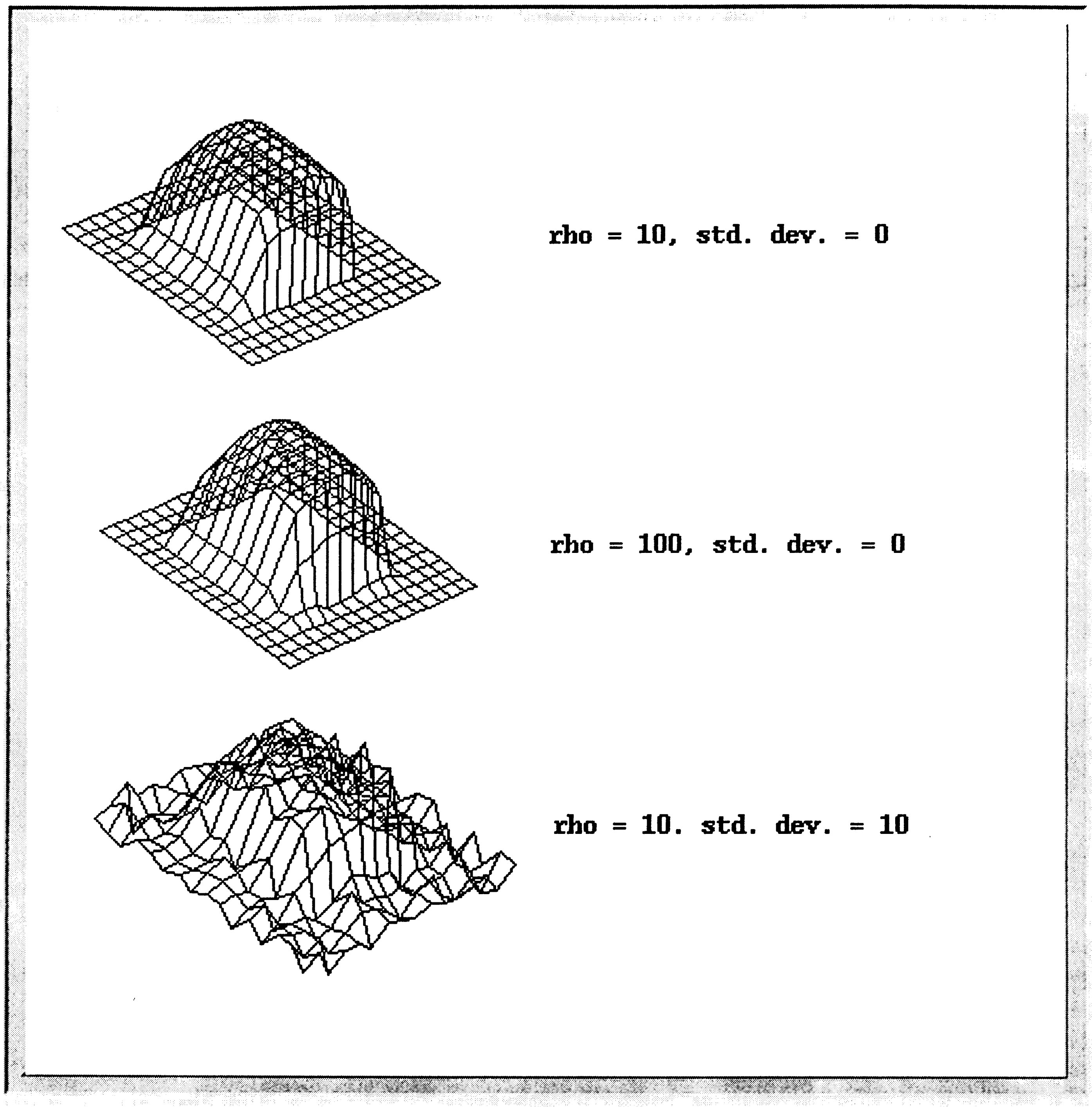


Figure 4: Spline fits of synthetic data using (ρ, σ) values of (10,0), (10,10), and (100,0) and a 32 $\times$ 32 knot grid

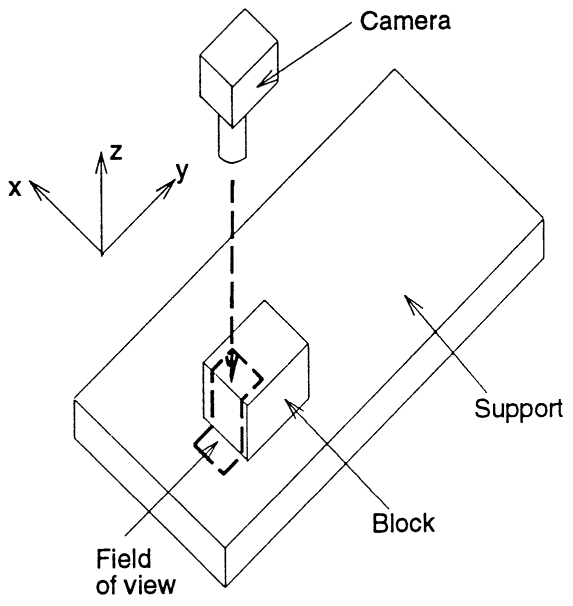


Figure 5: Geometry of the scanning setup

window was six pixels wide. If there were still too few neighboring points, the search was abandoned and a value of 0 was associated with that grid point. For computational tractability, at most 20 neighbors were considered by the median filtering operation. The first stage produced a regular mesh without outliers shown in Figure 7. Figure 7 also shows the transition and continuous regions chosen for this data set. The top planar region covers pixels (10,44) to (54,54), the bottom planar region covers pixels (10,10) to (54,20), the top transition region covers pixels (10,33) to (54,44) and the bottom transition region covers pixels (10,20) to (54,28). The second stage was run with the same five values for the smoothing coefficent ρ as the previous experiment. Figure 8 presents the resulting spline fits.

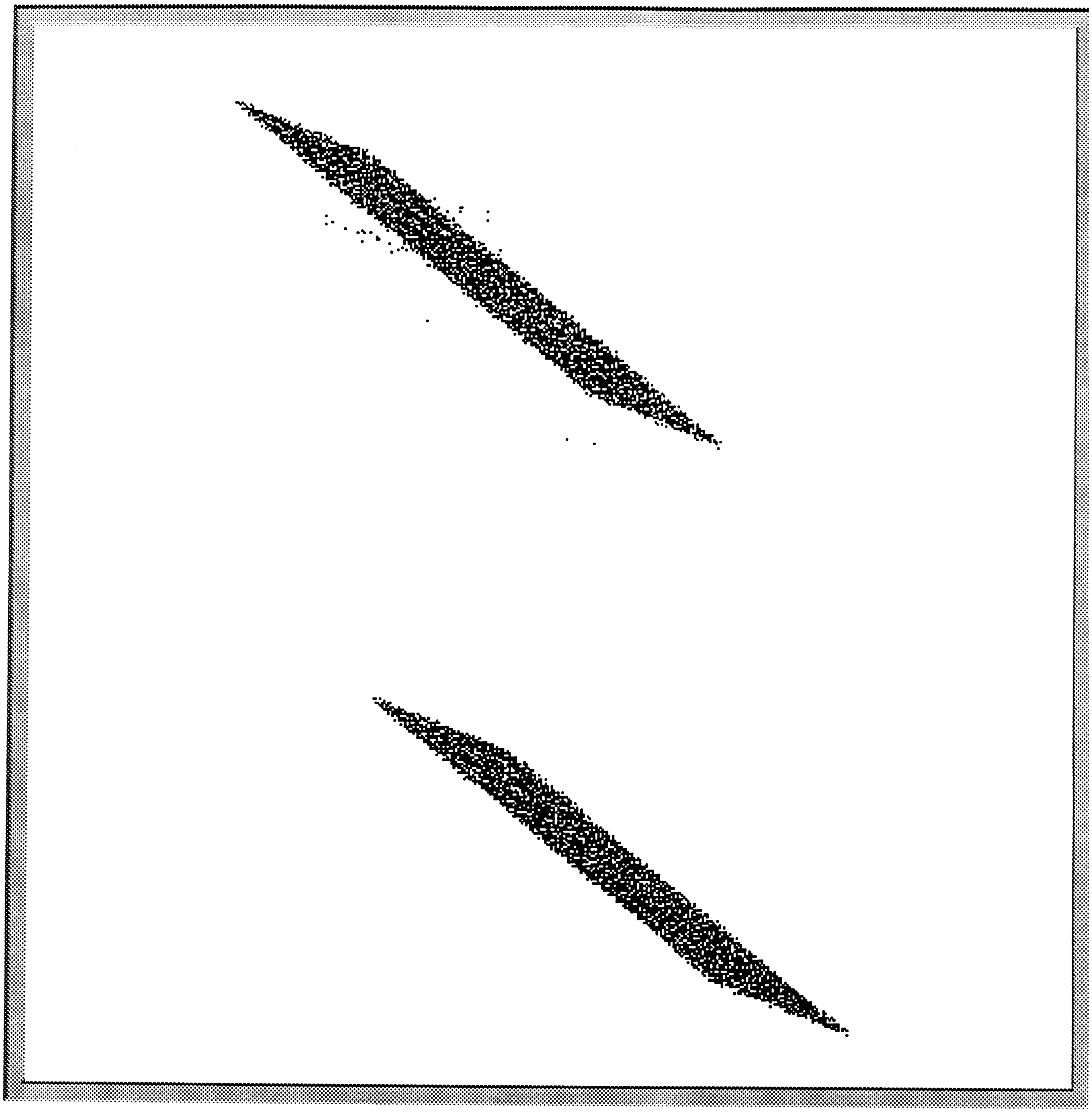


Figure 6: Original actual data, with typical features of actual laser scans: (a) outliers, (b) Gaussian noise, (c) sparse data. The equation for the top plane is $0.002293x + -0.089056y + z + -224.711655 = 0$ and the equation for the bottom plane is $0.00190x + -0.206307y + z + -1.779542 = 0$.

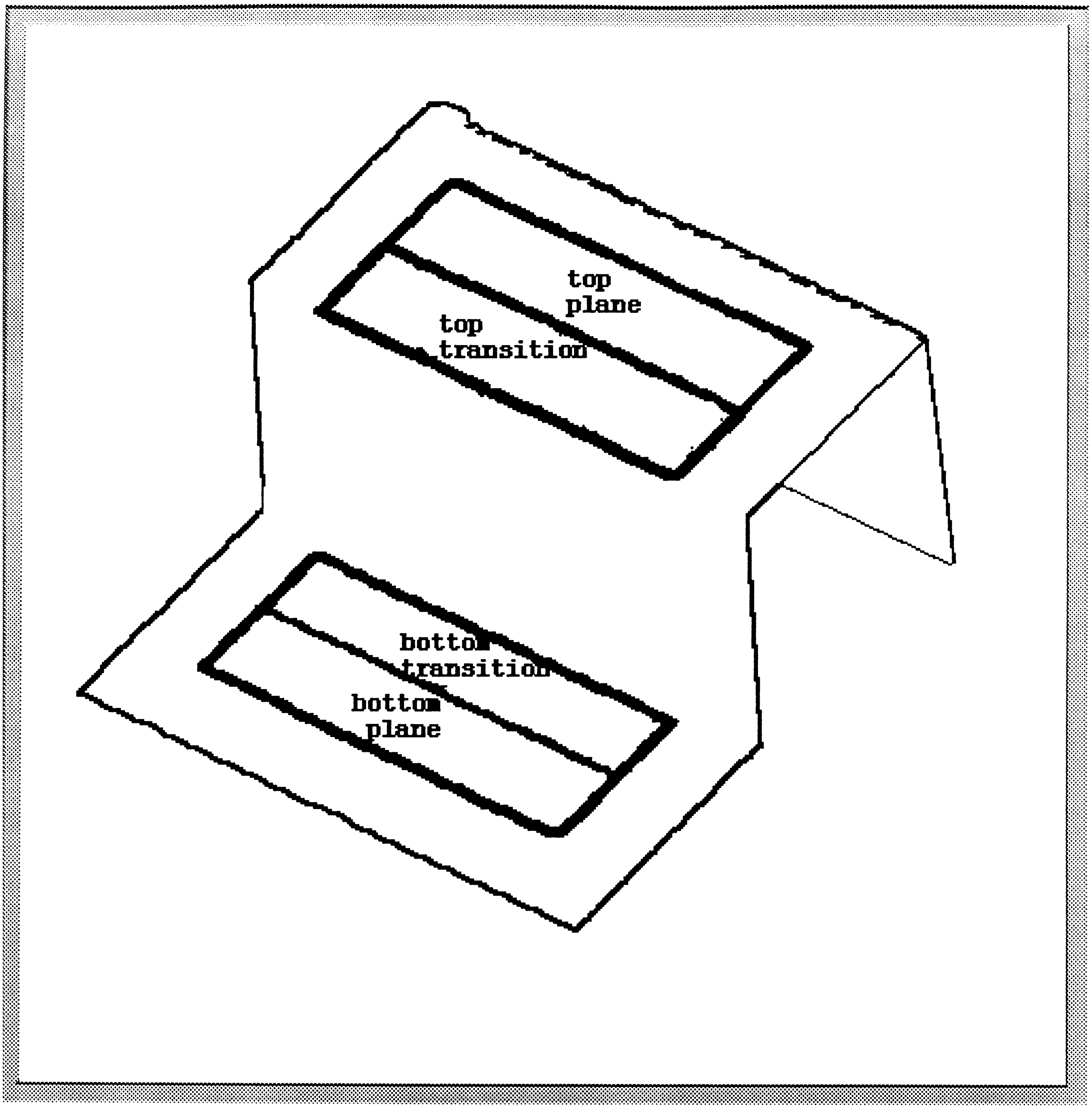


Figure 7: Moving Least Median Squares filtering. The outliers have been removed and the surface is now on a regular 64 × 64 grid. Maximum of twenty neighbors, window size starting at two pixels and growing to six. The planar and transition regions are shown. The top planar region covers pixels (10,44) to (54,54). The bottom planar region covers pixels (10,10) to (54,20). The top transition region covers pixels (10,33) to (54,44). The bottom transition region covers pixels (10,20) to (54,28).

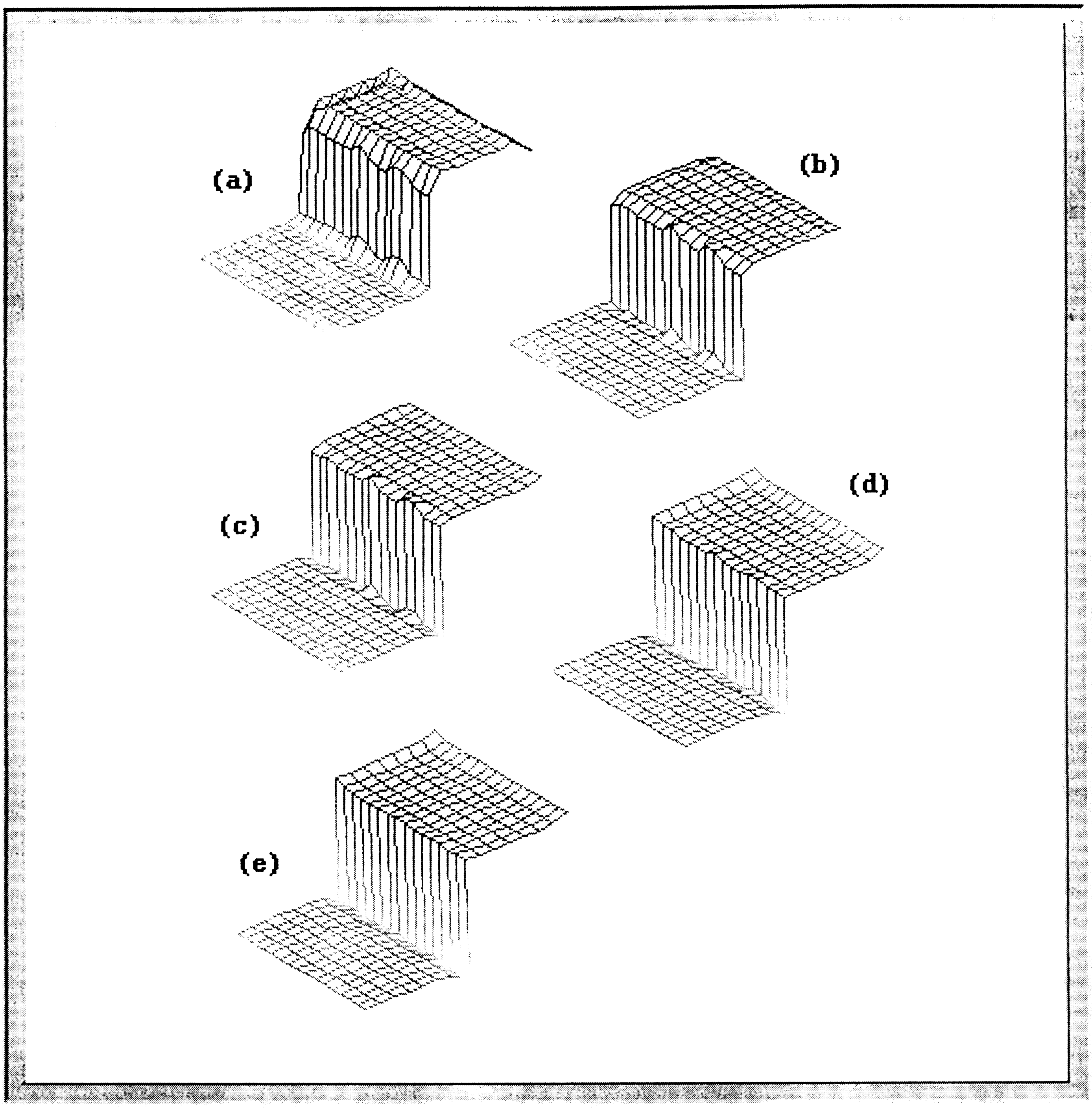


Figure 8: Spline fits of actual data using ρ values of 10, 50, 100, 500, and 1000 and a 32×32 knot grid

RMS errors: Continuous region	$\sigma = 0$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$
$\rho = 10$	2.482037	2.739954	3.515457	5.877151	8.724344
$\rho = 50$	2.296183	2.673086	3.585446	5.914074	8.789596
$\rho = 100$	2.325248	2.698088	3.604199	5.923125	8.787552
$\rho = 500$	2.474280	2.733994	3.624499	5.936285	8.799943
$\rho = 1000$	2.526491	2.740286	3.628135	5.939984	8.804909

Table 1: RMS errors for the continuous segment of the synthetic data case.

RMS errors: Transition region	$\sigma = 0$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$	$\sigma = 15$
$\rho = 10$	13.02646	11.06147	10.90778	11.75066	13.16949
$\rho = 50$	11.65798	10.64287	10.84079	11.75580	13.17786
$\rho = 100$	11.22843	10.58591	10.83498	11.75650	13.17991
$\rho = 500$	10.66605	10.54125	10.83144	11.75787	13.18161
$\rho = 1000$	10.57072	10.53557	10.83123	11.75793	13.18173

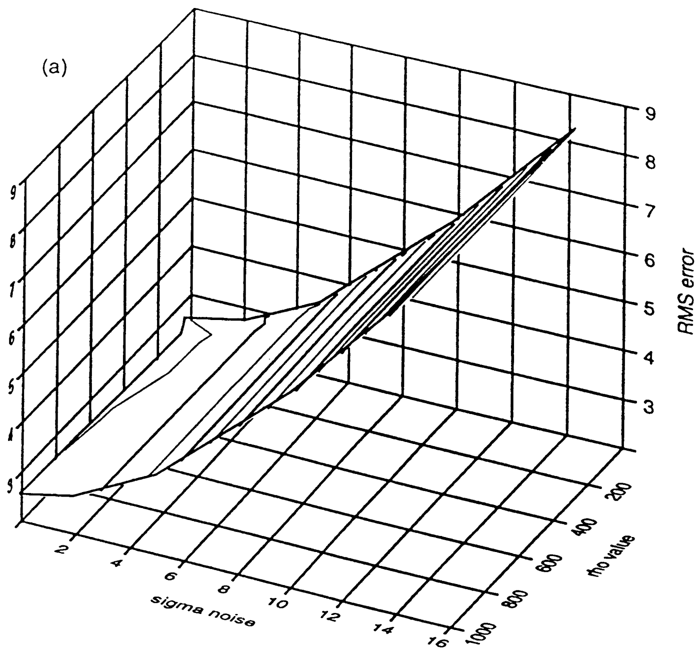
Table 2: RMS errors for the transition segment of the synthetic data case.

4 Experimental results and comparison of test cases

The RMS errors for the spline fits in the synthetic case are presented in Tables 1 and 2 and graphed in Figure 9. As expected, the error increased as the noise level increased, but much less so in the transition region. As the ρ value increased, the error in the transition region dropped but the error in the continuous region actually increased slightly. This occurred because the extremely high ρ values introduced artifacts into the transition region that extended into the continuous region.

The statistics for the analysis of actual data are presented in Table 3, and graphed in Figure 10. See Figure 11 for the distribution of errors. In the original data, the RMS error for the top plane was high because of outliers. The first stage eliminated these errors, but had little effect on Gaussian errors, as evidenced by the small decline in the error for the bottom plane. The second stage substantially reduced the error for the continuous regions, but increased the error for the transition regions. This occurred because of the inevitable smoothing effect of spline surfaces.

The results for the synthetic experiment measured the RMS error of the



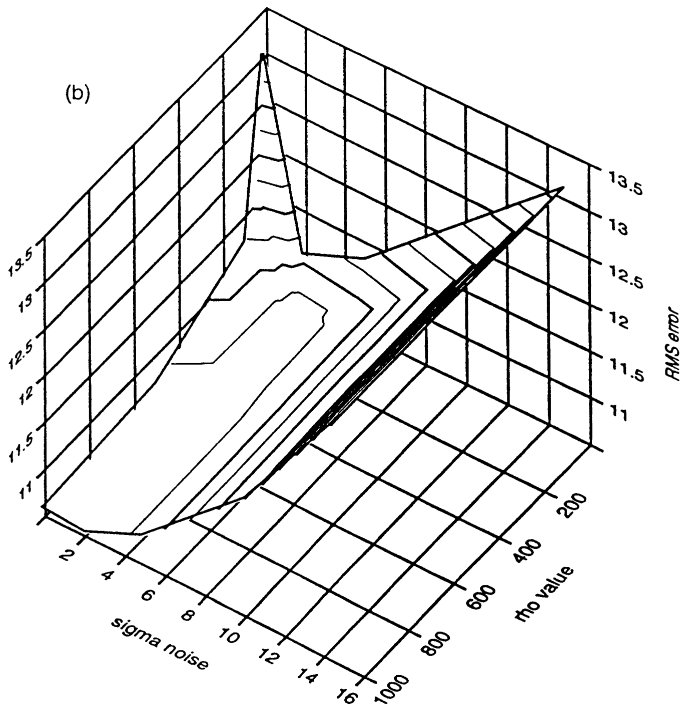


Figure 9: ρ vs. σ vs RMS error graph of the synthetic case, (a) continuous region, (b) transition region

RMS errors	Bottom plane	Top plane	Bottom Transition	Top Transition
Original data	0.810616	2.9244	0.926927	1.77072
First stage	0.706392	0.766733	0.779363	0.716332
Second stage, $\rho = 10$	0.450508	0.469865	4.041837	3.786735
$\rho = 50$	0.499335	0.551630	2.971268	2.288447
$\rho = 100$	0.534586	0.574002	2.537365	3.619183
$\rho = 500$	0.661485	0.722999	3.621776	4.637547
$\rho = 1000$	0.765062	0.731556	4.545904	5.09075

Table 3: RMS errors for the actual data case.

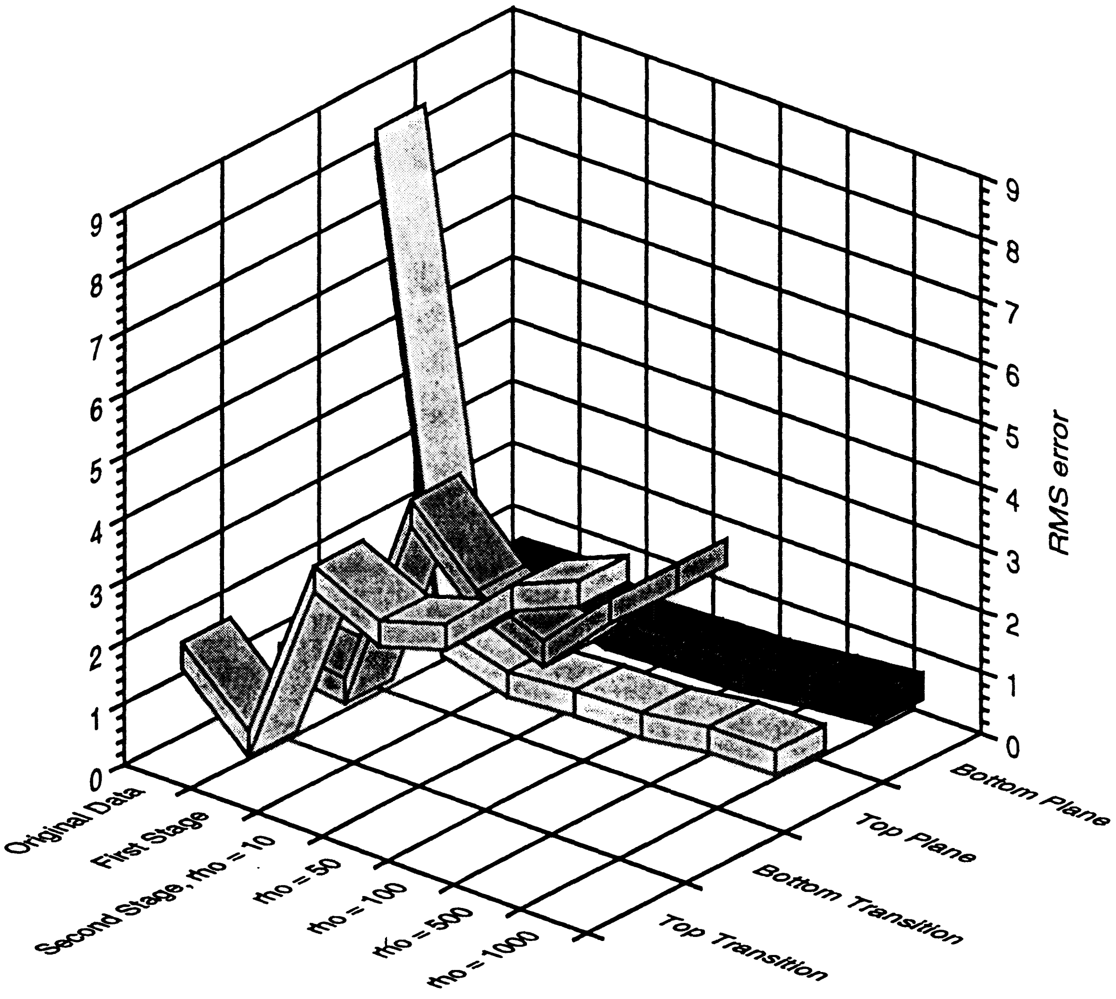


Figure 10: RMS error graph for the actual data case

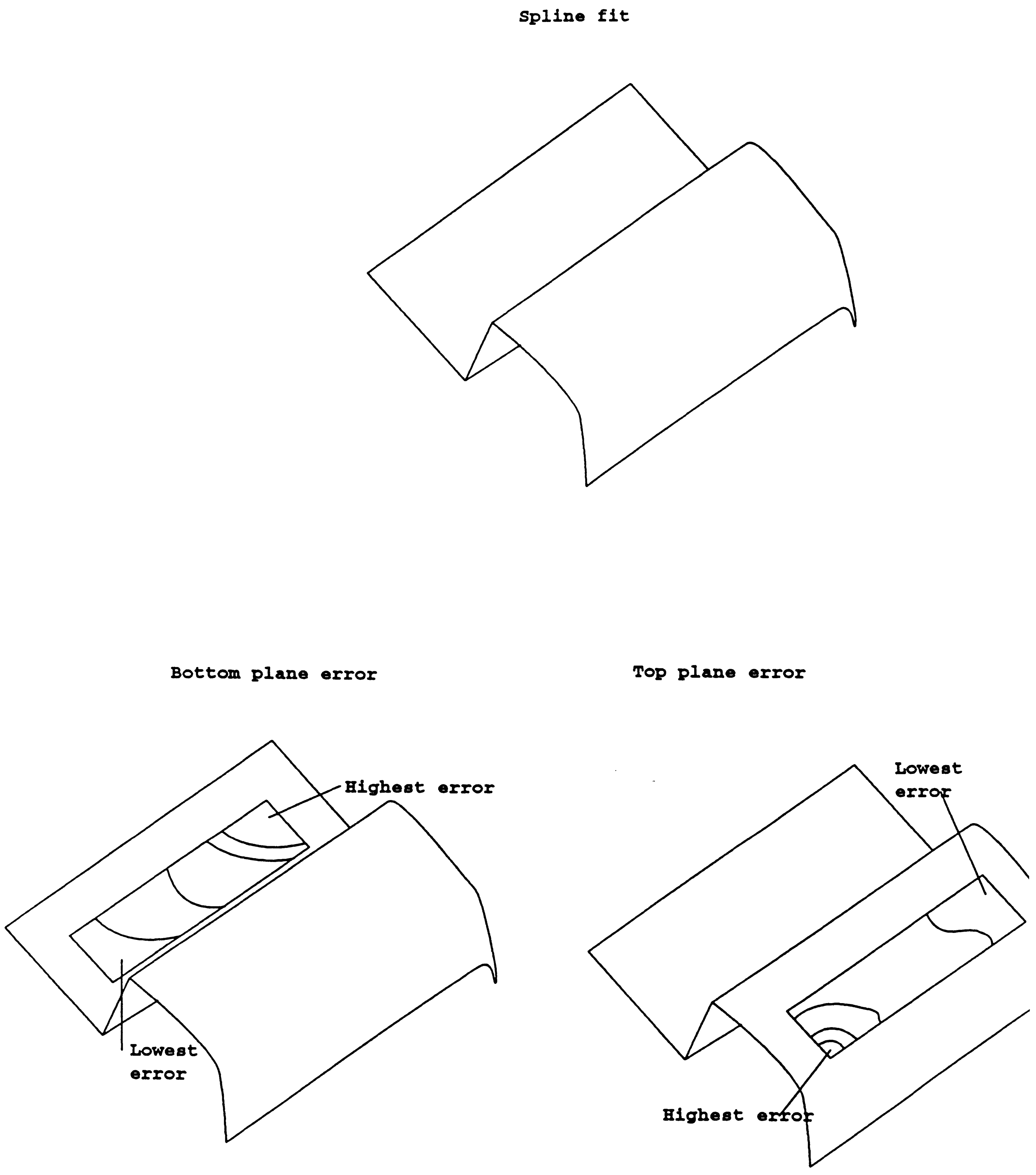


Figure 11: distribution of errors for the actual data case

second stage as a function of ρ and the sensor error. The hypothesis was that more inaccurate sensors would require smaller ρ values to minimize error, which was observed.

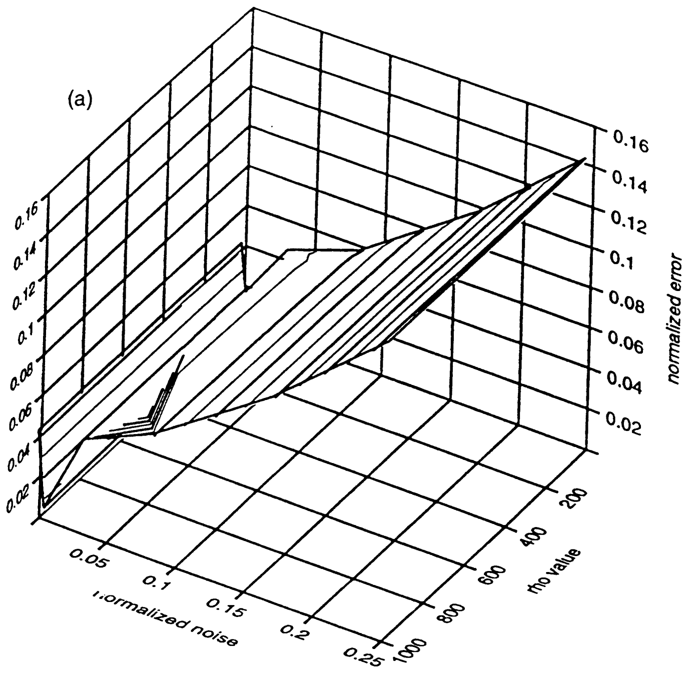
Since the synthetic results were obtained directly from gridded data without outliers, they cannot be directly compared to the input data for the actual case. Instead, since the first stage removes outliers and grids the data, it can be compared with the sensor error from the synthetic case. The graph of the synthetic data with the new results superimposed is in Figure 12. The error curves for the bottom and top planes match the synthetic results well, so we conclude that the two-stage algorithm is effective in reducing error found in the raw data generated by laser range scanners. In fact, for the Perceptron scanner, using a ρ value of 10, the root mean squared Gaussian error in the continuous region was reduced by more than one third, and all outliers were eliminated. These results were obtained using a grid with 64 rows and columns, and a spline surface with 32 knots in each direction. We have experimentally determined that increasing the grid density and number of knots does not significantly reduce the error. We expect that similar results can be achieved with any range scanner, and accuracy can be maximized by choosing the appropriate ρ value for the error characteristics of the sensor, which can be done using the graph in Figure 13, where the error units have been normalized by the mean depth value so any sensor's expected performance can be measured.

5 Conclusion

This report has described the error-reduction properties of the two-stage algorithm for surface reconstruction. The first stage uses a moving least median squares filter to clean and grid range data. The second stage uses a weighted bicubic B-Spline representation of the visible surface. The algorithm fits a surface to scattered depth data while rejecting outliers and without distortion such as Gibbs phenomena at discontinuities. The moving least median squares regression algorithm in the first stage removes outliers and preserves discontinuities. Weighted bicubic splines are able to fit scattered data (containing discontinuities and normal noise) with fidelity and no distortion such as Gibbs phenomena.

The major advantages of our approach are

- The increase in sensor accuracy.



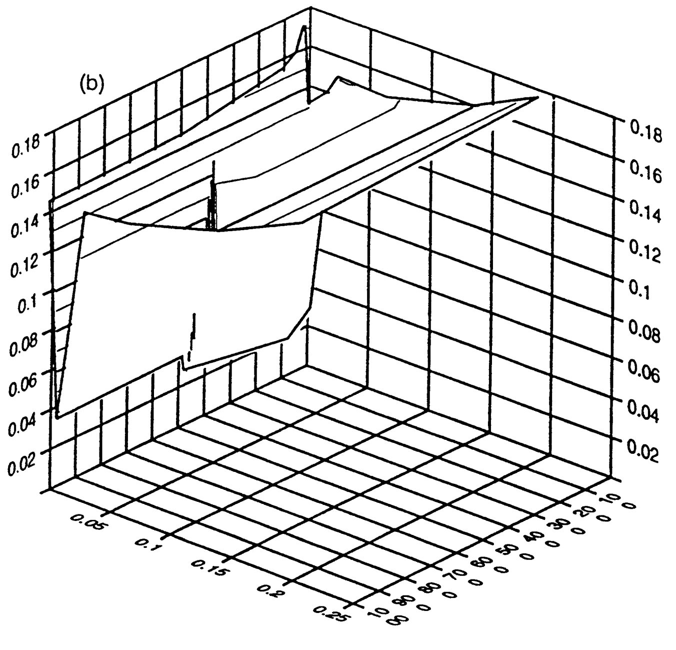
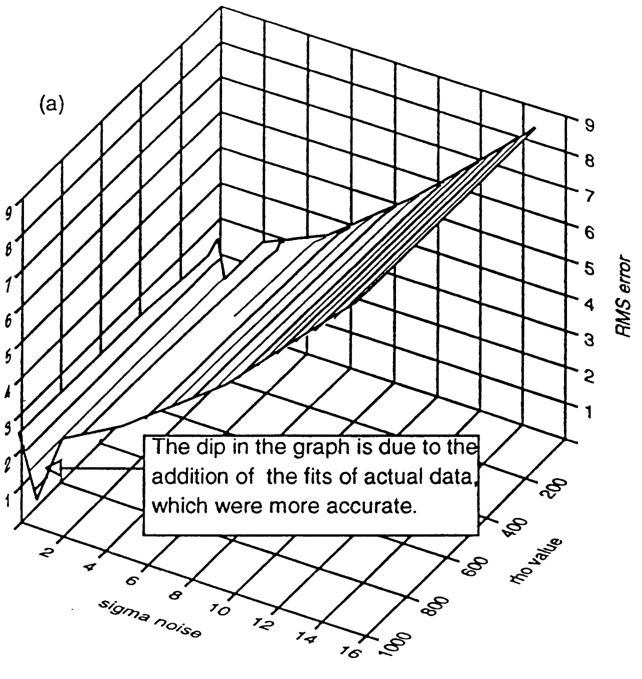


Figure 12: Error comparison between the actual and synthetic cases, (a) continuous region, (b) transition region



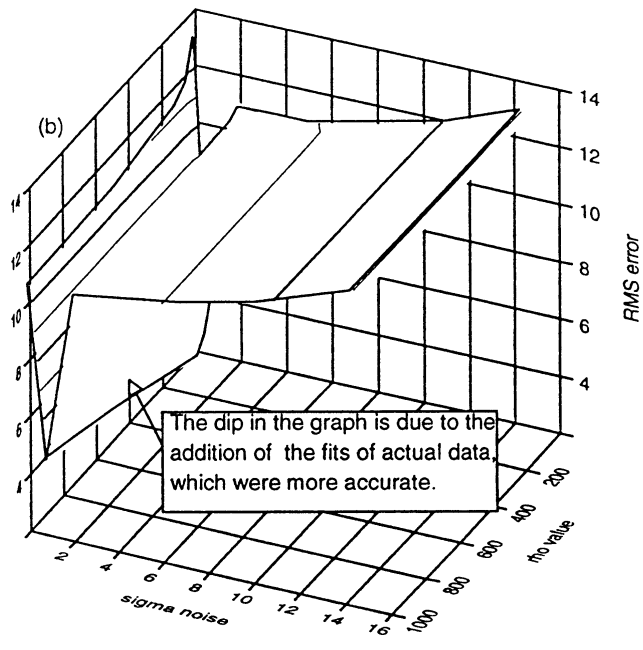


Figure 13: General graph of the expected error of the second stage output given sensor noise (σ) and tightness of spline fit (ρ), (a) continuous region, (b) transition region

- A reduction in the data volume due to approximation.
- Reduced time data collection.

Acknowledgments

We would like to thank Yudong Chen and Perceptron for their support and the use of their equipment. We would also like to thank Prof. Ramesh Jain at the University of California San Diego, Dwight Carlson and Gary Johnson at Perceptron, and Charles Wu and Sam Ebenstein at Ford Scientific Research Laboratories for their advice. This work was supported by grants from Perceptron and Ford Motor Company.

References

[1] P. J. Besl. *Surfaces in Range Image Understanding.* Springer-Verlag, 1988.

[2] Paul J. Besl and Ramesh C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(2):167–192, March 1988.

[3] Thomas A. Foley. Weighted bicubic spline interpolation to rapidly varying data. *ACM Trans. Graphics*, 6:1–18, January 1987.

[4] N. C. Gallagher and G. L. Wise. A theoretical analysis of the properties of median filters. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 29:1136–1141, 1981.

[5] Hiroshi Inoue. A least-squares smooth fitting for irregularly spaced data: Finite-element approach using the cubic B-spline basis. *Geophysics*, 51(11):2051–2066, November 1986.

[6] S. S. Sinha. Differential properties from adaptive thin plate splines. In *Proc. S. P. I. E. Conf. Geometric Methods in Computer Vision*, San Diego, California, July 1991.

[7] S. S. Sinha and B. G. Schunck. A two-stage algorithm for discontinuity-preserving surface reconstruction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(1):36–55, January 1992.

[8] Sarvajit S. Sinha and Brian G. Schunck. Discontinuity-preserving surface reconstruction in vision processing. In *Proc. Conf. Computer Vision and Pattern Recognition*, 1989.

[9] Sarvajit S. Sinha, Brian G. Schunck, and Seth Rogers. Towards reverse engineering: Accuracy of adaptive thin-plate surface fitting. In *SPIE: Sensor Fusion IV*, pages 532–541, November 1991.

[10] Demetri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(4):413–424, July 1986.

[11] Demetri Terzopoulos. The computation of visible surface representations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10:417–438, July 1988.

[12] Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of Ill-Posed Problems.* V. H. Winston & Sons, 1977.

A File formats

The file formats used by the two-stage algorithm correspond to the three types of scanning patterns employed by laser scanners: scattered, line, and gridded. Any of these formats can be used as either input to or output from the algorithm.

The input and output formats are represented by an ASCII text file. Scattered data has the extension `.xyz` and the data points are presented one per line, with the x, y, and z coordinates followed by a semi-colon. Here is an example of file `scat_example.xyz`:

```
113.4181;-0.8952;205.9230;
113.3281;-1.2301;205.9526;
113.3121;-1.3980;205.8819;
113.3360;-1.6866;205.8326;
113.3089;-2.0491;205.8505;
```

The line data file format has the extension `.DT`. The data points are organized into a set of strings, where each string of points has the same x coordinate. A new string is begun with an `X␣` and the constant x coordinate on its own line. Each point in the string is defined by a `P␣` followed by the y and z coordinates, separated by a space. Again, each point is on its own line. Here is an example of file `line_example.DT`:

```
X 0.000000
P 62.997548 232.980167
P 62.076995 233.736144
P 61.153380 231.671953
X 1.258941
P 62.690697 233.216075
P 61.768307 232.170575
P 60.847142 232.497629
X 2.519143
P 61.462068 233.130290
P 62.382009 232.020453
```

Finally, the grid file format is a standard image format known as **pgm**, which is a widely used format for image conversion and manipulation. Following is a part of the **man** page describing the **pgm** format:

```
NAME
     pgm - portable graymap file format
```

```
DESCRIPTION
     The portable graymap format is a lowest common denominator
     grayscale file format.  The definition is as follows:

     - A "magic number" for identifying the file type.  A pgm
       file's magic number is the two characters "P2".

     - Whitespace (blanks, TABs, CRs, LFs).

     - A width, formatted as ASCII characters in decimal.

     - Whitespace.

     - A height, again in ASCII decimal.

     - Whitespace.

     - The maximum gray value, again in ASCII decimal.

     - Whitespace.

     - Width * height gray values, each in ASCII decimal, between
       0 and the specified maximum value, separated by whi-
       tespace, starting at the top-left corner of the graymap,
       proceeding in normal English reading order.  A value of 0
       means black, and the maximum value means white.

     - Characters from a "#" to the next end-of-line are ignored
       (comments).

     - No line should be longer than 70 characters.

     Here is an example of a small graymap in this format:
     P2
     # feep.pgm
     24 7
     15
     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
     0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
     0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
     0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
     0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
     0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Since the gridded file format implicitly assumes the location of the grid points, the x coordindate of each grid point is defined to be the column number of the point, and the y coordinate to be the row number of the point.

All of these file formats contain as their basic element a point cloud as described above, but the line and grid formats contain more structure, so the algorithm can take advantage of this to operate more efficiently.

B Subroutine declarations

The first stage has one major subroutine which calculates the least median of squares value for a given grid location. The definition of `lms` is

```
double lms(data_type type, any_type data, float row, float col,
   float delta_x, float delta_y, float cur_window);
```

where `data` is the input data of type `type`, `row` and `col` is the grid location, `delta_x` and `delta_y` is the area covered by only one grid point (a "pixel"), and `cur_window` is the size of the window in pixels.

The second stage has one subroutine to calculate the spline fit and one to evaluate the spline at a particular location. The subroutine `lsf2c` finds the spline coefficients. Its definition is

```
      subroutine lsf2c(xmin,ymin,xmax,ymax,xd,yd,fd,sd,weightx,
     1weighty,nd,unl,und,rou,tau,mx0,my0,n2,nit,cij,kc,a,b,indp,aint)
```

where:

- `xmin`, `xmax`, `ymin`, and `ymax` define the bounding box of the data,
- `xd`, `yd`, and `fd` are arrays of the x, y, and z locations of each point,
- `sd` is an estimate of the standard deviation at each point,
- `weightx` and `weighty` contain the weights of each point,
- `nd` is the number of data points,
- `unl`, `und`, `rou`, `tau`, `mx0`, and `my0` are the fitting paramters described earlier,
- `n2` and `nit` are the number of binary divisions and number of interations in the relaxation step,

- `cij` is the two-dimensional array of the spline coefficients whose row dimension is `kc`,
- Finally, `a`, `b`, `indp`, and `aint` are temporary work arrays.

The subroutine to evaluate a point on the spline surface is named `lsf2f` and is defined

```
      subroutine lsf2f(xmin,ymin,xmax,ymax,mx,my,cij,kc
     1                                ,x,y,idefx,idefy,f)
```

where:

- The first eight parameters are identical to `lsf2c`,
- `x` and `y` is the location of the point,
- `idefx` and `idefy` is the order of x, y differentiation of the fitted surface (0,0 is the value of the surface itself), and
- `f` is the desired z value.