A Robust Variable Order Facet Model for Image Data

Y. Mainguy, J. B. Birch, and L. T. Watson

TR 91-33

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

December 12, 1991

A Robust Variable Order Facet Model for Image Data*

Y. Mainguy
Department of Computer Science

J. B. Birch
Department of Statistics

L. T. Watson
Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106

TR 91–33

Abstract. The underlying piecewise continuous surface of a digital image can be estimated through robust statistical procedures. This paper contains a systematic Monte Carlo study of M-estimation and LMS estimation for image surface approximation, an examination of the merits of postprocessing and tuning various parameters in the robust estimation procedures, and a new robust variable order facet model paradigm. Several new goodness of fit measures are introduced, and systematically compared. It is shown that the M-estimation tuning parameters are not crucial, postprocessing is cheap and well worth the cost, and the robust variable order facet model algorithm (using M-estimation, new statistical goodness of fit measures, and postprocessing) manages to retain most of the statistical efficiency of M-estimation yet displays good robustness properties, and preserves the main geometric features of an image surface: step edges, roof edges and corners.

* This work was supported in part by Department of Energy grant DE-FG05-88ER25068/A002, USDA grant 91-37103-6544, National Science Foundation grant CTS-8913198, and Air Force Office of Scientific Research grant 89-0497.

1. Introduction.

1.1. Surface estimation.

The sensors used in computer vision can measure intensity values (CCD camera), distances from the sensor to the scene (range sensors) or temperatures (thermal sensors), to cite a few. Due to obvious physical limitations, the measurements can only be performed at spaced intervals. The measured (or observed) data are thus discrete and only related to each other by the underlying signal. The relations between adjacent pixels are somewhat lost and distorted by the sensing process because of noise and quantization errors. Therefore the first task after sensing is to filter out the effect of noise and quantization, and construct an intermediate level model of the data as a basis for further processing. The intermediate level model assumes that the sensed image is a piecewise continuous surface.

Traditionally filtering in computer vision consists of scanning the image with a window, replacing the center pixel value in the window by the result of a linear or nonlinear operator on the values of its neighbors in the window. For instance, the value of each pixel x can be estimated as the median value of all the pixels in the window centered on the pixel x. Similarly, we scan the image with windows performing robust estimations of polynomial models of the pixel values in each window. Since a majority of pixels in a window must support the polynomial model, *the support of any feature in the image must be at least of 5 pixels when processing with a* 3×3 *window and 13 pixels when using a* 5×5 *window.*

The work described in this paper is heavily based on robust statistics. A robust estimator yields good estimates when the sample data is distributed according to a given distribution and reasonable estimates if the data is contaminated with points from another distribution, which is the case when a surface discontinuity (edge or corner) is present in the window. An estimator is characterized by its efficiency and its robustness. The term efficiency refers to the relative ability of a procedure to yield optimal estimates under ideal conditions; e.g., least squares estimators are optimally efficient for identically distributed normal variables. The term robustness refers to the relative ability of a procedure to yield reasonable estimates under less than ideal conditions. There is a fundamental tradeoff between efficiency and robustness: fully efficient procedures cannot be the most robust and the most robust procedures cannot be fully efficient.

The main contributions of this paper are a systematic Monte Carlo study of M-estimation and LMS estimation for image surface approximation, examination of the merits of postprocessing and tuning various parameters in the robust estimation procedures, and a new robust variable order facet model paradigm. Several new goodness of fit measures are introduced, and systematically compared. Briefly, the main conclusions are that the M-estimation tuning parameters are not crucial, postprocessing is cheap but well worth the cost, and the robust variable order facet model algorithm (using M-estimation, new statistical goodness of fit measures, and postprocessing) manages to retain most of the statistical efficiency of M-estimation yet displays good robustness properties.

1.2. Related work.

Constant coefficient window operators have been used since the beginning of computer vision [34], [33], [23], [1], [15, 16], [7], [2]. Local surface fitting based on orthogonal polynomials provides the theoretical basis for such operators. Robust estimation probably started in computer vision with the use of median filtering [13, 21] and the Hough transform [11], albeit informally. The first vision paper to actually use formal robust statistical procedures appears to be Forstner [12]. Davis and Rosenfeld [10], Hurt and Rosenfeld [24], Hoffman and Jain [20], and Harwood et al. [19] have used

a robust technique known as nearest-neighbor smoothing. Bolles and Fischler's Ransac method [8] and Besl and Jain's region growing method [3] address related problems of outlier rejection in global surface fitting. Chen [9] accomplished outlier rejection in a form closely related to the robust statistical influence function. Medioni [28] developed a robust derivative estimation scheme based on diffusion concepts. Kashyap and Eom [26, 27] have developed robust methods based on M-estimation for image smoothing which are extensions of Sharma and Chellappa's [37] method for two-dimensional spectral estimation and Hansen and Chellappa's [14] algorithms. Adaptive window operators for smoothing, derivative estimation, and edge detection are known [41], and can be made more robust using the two-dimensional autoregression model of Hansen and Chellappa [14]. Haralick and Joo [15, 16] discussed the use of M-estimation for line fitting and pose estimation. Meer, Mintz, and Rosenfeld [29] have a least median of squares (LMS) based paradigm to filter noise and segment images. Continuing this work, Mintz developed a consensus by decomposition paradigm [31, 32]. Schunck [38, 39] used LMS for surface reconstruction without postprocessing. Roth and Levine [35] also make use of LMS to segment geometric objects whereas Waks and Tretiak [40] use M-estimation to detect the boundaries of regions. Zhuang and Haralick [42, 43] have developed a highly robust estimation procedure, which has a very high breakdown value, by using heuristic reasoning combined with Bayesian rules. In their robust clustering algorithm, Jolion, Meer and Bataouche [25] analyze the Hough space through M-estimation. Meer et al. published a review of the different robust estimators used in computer vision [30].

This paper presents a method of estimating the underlying piecewise continuous surface through robust statistical procedures. We first define M-estimation in Section 2, and then study the effect of tuning the M-estimation parameters in Section 3. In Sections 4 and 5, we introduce and compare several different new variable order and postprocessing paradigms. Section 6 introduces a LMS based paradigm which is compared to the equivalent M-estimation paradigm of Section 4. In Section 7, a postprocessed M-estimation paradigm is compared to a postprocessed LMS paradigm. Section 8 summarizes the results of the Monte Carlo studies in the earlier sections, and draws some conclusions.

2. M-estimation.

2.1. Generalities.

A good understanding of M-estimation can be achieved through a comparison with least squares (LS) estimation since M-estimation is a generalization of LS estimation. In the typical signal processing problem, the N responses $d(i)$ are observed at the integers $i = 1, \ldots, N$. One goal is to then model the responses $d(i)$ in terms of some "model" function $f_{\mathbf{a}}(i)$, which depends on the vector $\mathbf{a}$, the regression coefficients, and i, the regressor variable. Since the vector $\mathbf{a}$ is unknown and one desires to use $f_{\mathbf{a}}(i)$ to model the $d(i)$, $\mathbf{a}$ must be estimated using the data $(i, d(i))$. One technique is LS estimation, where one tries to minimize the sum of squared residuals

$$r(\mathbf{a}) = \sum_{i=1}^{N} e_{\mathbf{a}}(i)^2, \qquad \text{where } e_{\mathbf{a}}(i) = d(i) - f_{\mathbf{a}}(i), \tag{2.1}$$

over all parameter vectors $\mathbf{a}$ defining a class of "model" functions $f_{\mathbf{a}}$. The "best fit" is that model function $f_{\mathbf{a}}$ which minimizes $r(\mathbf{a})$. The error $e_{\mathbf{a}}(i)$ corresponds to the difference between the observed response $d(i)$ and the value of the fitting function $f_{\mathbf{a}}(i)$, the model's estimated response evaluated at the corresponding i. The often stated weakness of least squares is that the function

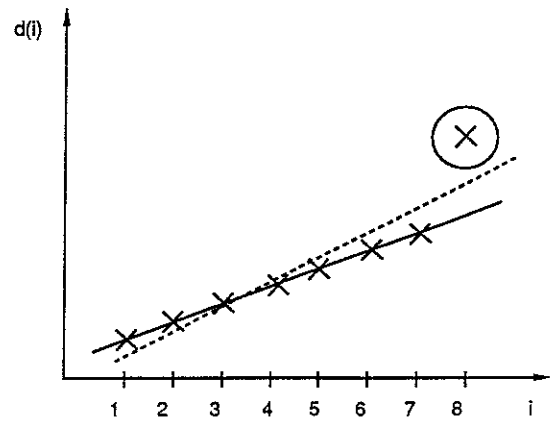

Figure 2.1: ideal versus LS fit.

$f_a(i)$ is computed by equally weighting all the data even if some of these data are not consistent with the pattern expressed by the majority of the data. To illustrate this, consider Figure 2.1, where the data obviously follow a line, except one point (circled cross).

The solid line represents the line we would like to get, and the dashed line shows the actual least squares line. The circled point is called an *outlier*, while the other data are called *inliers*. An inlier is a datum which is consistent with the model whereas an outlier is inconsistent. One should not infer from this example that the dichotomy between inliers and outliers is always so clear. We will give more details on this later.

Another concept of robust statistics is *the breakdown point* of an estimator. It is the smallest fraction of contamination that can cause the estimator to take on values arbitrarily far from the value of the estimator computed without contamination [36]. The breakdown value of least squares estimation is $1/N$, which means that its asymptotic breakdown value is zero (one bad datum can corrupt the whole estimator).

To reduce the impact of outliers on resulting estimators, Huber [22] suggested that one minimize $\sum_{i=1}^{N} \rho(e_a(i)/s)$ instead of $\sum_{i=1}^{N} (e_a(i))^2$, where ρ must be even (only the magnitude of the error is significant, not its sign) and differentiable. A scaling factor s is required to rescale the residuals. In LS estimation, $\rho(x) = x^2/2$.

2.2. Algorithm.

The general algorithm for M-estimation is described below for the two dimensional case, which is just a straightforward extension of the one dimensional algorithm. The purpose of this algorithm is to determine a surface that approximates a majority of pixel values in a window. The input data is a $n \times m$ matrix, extracted with a window from an image, where n and m are odd so that the window can be centered on a pixel, and usually n and m are equal (square window). A function d is used to index the data in the window according to the local coordinate system: $d(r, c)$ is the data value at the pixel (r, c) located at the intersection of the rth row and the cth column. The origin of the local coordinate system is set at the center of the window, the "row-axis" is oriented from top to bottom, and the "column-axis" from left to right. The fitting functions f_a will be also defined in the local coordinate system. For example with a 3×3 window (Figure 2.2), $f_a(-1, -1)$ will be

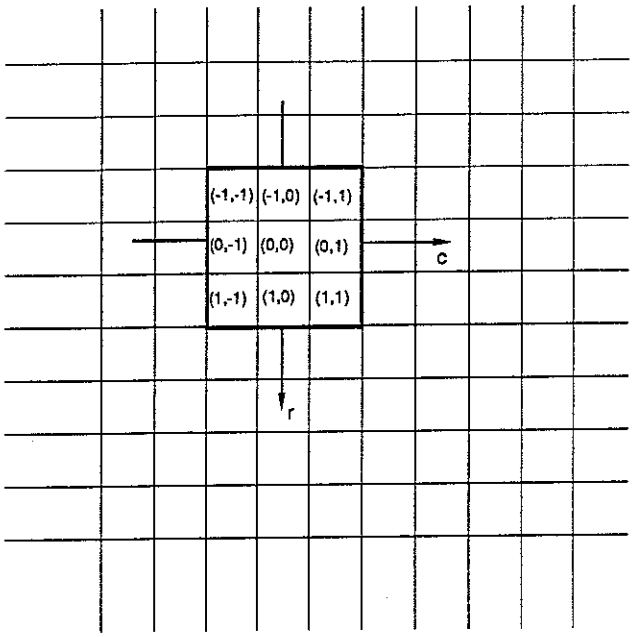

Figure 2.2: a 3×3 window to extract data from an image.

the pixel value estimate of the left upper pixel in the window, $f_{\mathbf{a}}(0,0)$ the pixel value estimate of the center pixel in the window and so on.
For later use, we define

$$\psi(x) = \frac{d}{dx}\rho(x) \tag{2.2}$$

and

$$\hat{n} = \frac{n-1}{2}, \qquad \hat{m} = \frac{m-1}{2}. \tag{2.3}$$

To determine the coefficient vector $\mathbf{a}$ of the fitting function, we minimize the function:

$$r(\mathbf{a}) = \sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} \rho\left(\frac{e_{\mathbf{a}}(r,c)}{s}\right) \tag{2.4}$$

where the residual

$$e_{\mathbf{a}}(r,c) = d(r,c) - f_{\mathbf{a}}(r,c) \tag{2.5}$$

is the error between the observed data and the model function values computed with the model function defined by

$$f_{\mathbf{a}}(r,c) = \sum_{i=1}^{p} a_i \phi_i(r,c), \tag{2.6}$$

where the coefficients a_i are the p components of the vector $\mathbf{a}$. The ϕ_i are a family of basis functions, which are polynomials in our application. For example, in a planar fit ($p = 3$), the functions are $\phi_1(r,c) = 1$, $\phi_2(r,c) = r$ and $\phi_3(r,c) = c$. The scaling factor s is evaluated using the MAD (Median Absolute Deviation)

$$s = 1.4286 \,\text{median}|e_{\mathbf{a}}(r,c)|. \tag{2.7}$$

The MAD is both a robust estimator of the standard deviation of the noise and computationally inexpensive. The constant 1.4286 is used to make the MAD a consistent estimator of the standard deviation if the noise are random observations from a normal distribution [22, p. 108]. In the algorithm that follows, s is computed from (2.7) initially and then kept constant during the remainder of the algorithm [5].

The necessary conditions for a minimum of (2.4) are

$$\frac{\partial r(\mathbf{a})}{\partial a_i} = 0 \qquad \forall\, i = 1, \ldots, p. \tag{2.8}$$

With the additional constraint of ρ being convex, i.e., ψ monotonically increasing, the solution a is unique [5]. Combining (2.4) and (2.8) we get

$$\sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} \psi\left(\frac{e_{\mathbf{a}}(r,c)}{s}\right) \phi_i(r,c) = 0 \qquad \forall\, i = 1, \ldots, p. \tag{2.9}$$

Substituting the weights

$$w(r,c) = \frac{\psi\left(\dfrac{e_{\mathbf{a}}(r,c)}{s}\right)}{\dfrac{e_{\mathbf{a}}(r,c)}{s}} \tag{2.10}$$

into (2.9) gives

$$\sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=\hat{m}}^{\hat{m}} w(r,c) \left(\frac{e_{\mathbf{a}}(r,c)}{s}\right) \phi_i(r,c) = 0 \qquad \forall\, i = 1, \ldots, p \tag{2.11}$$

or, using (2.6), we obtain

$$\sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} \sum_{k=1}^{p} w(r,c)\, \phi_i(r,c)\, a_k \phi_k(r,c)$$

$$= \sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} d(r,c)\, w(r,c)\, \phi_i(r,c) \qquad \forall i = 1, \ldots, p. \tag{2.12}$$

This can be written in matrix form as

$$\Phi^t \mathbf{W}_{\mathbf{a}} \Phi\, \mathbf{a} = \Phi^t \mathbf{W}_{\mathbf{a}}\, \mathbf{z}, \tag{2.13}$$

which is a nonlinear equation. Φ is a $nm \times p$ matrix whose rows are $\phi_1(r,c), \ldots, \phi_p(r,c)$, $\mathbf{W}_{\mathbf{a}}$ is a $nm \times nm$ diagonal matrix whose diagonal elements are $w(r,c)$, $\mathbf{a}$ is a p-vector whose entries are a_i, and $\mathbf{z}$ is a nm-vector whose entries are $d(r,c)$.

The matrix equation (2.13) is of the form

$$\mathbf{A}^t \mathbf{A}\, \mathbf{a} = \mathbf{A}^t \mathbf{b} \qquad \text{where} \quad \mathbf{A} = \mathbf{W}_{\mathbf{a}}{}^{1/2}\, \Phi \quad \text{and} \quad \mathbf{b} = \mathbf{W}_{\mathbf{a}}{}^{1/2}\, \mathbf{z}. \tag{2.14}$$

A solution to (2.13) can be found using a QR-decomposition of $\mathbf{A}$,

$$\mathbf{A} = \mathbf{QR}, \tag{2.15}$$

where $\mathbf{Q}$ is a $nm \times nm$ orthogonal matrix, and $\mathbf{R}$ is an $nm \times p$ upper triangular matrix. Since $\mathbf{W_a}$ depends on $\mathbf{a}$, $\mathbf{Q}$ and $\mathbf{R}$ also depend on $\mathbf{a}$. One gets the recurrence formula

$$\mathbf{a}^{(k+1)} = (\mathbf{R}^t_{\mathbf{a}^{(k)}}\mathbf{R}_{\mathbf{a}^{(k)}})^{-1}\mathbf{R}^t_{\mathbf{a}^{(k)}}\,\mathbf{Q}^t_{\mathbf{a}^{(k)}}\,\mathbf{b} \tag{2.16}$$

($\mathbf{a}^{(k)}$ denotes the value of $\mathbf{a}$ at the kth iteration). This method of solving (2.13) is known as iteratively reweighted least squares (IRLS). To initialize the recurrence, an initial fit coefficient vector $\mathbf{a}^{(0)}$ is needed; $\mathbf{a}^{(0)}$ is set with the previous order fit coefficient vector, e.g., the initial quadratic fit coefficient vector will be set with the planar fit coefficient vector that has just been computed. There are no particular difficulties with the implementation of this algorithm (IRLS M-estimation) except that the rank of the matrix $\mathbf{A}$ has to be checked; rank(A) must equal p, otherwise the QR-decomposition of $\mathbf{A}$ cannot be computed.

2.3. ψ-functions and weights.

If the ψ-function is chosen as the identity function i.e., $\psi(x) = x$, then $\mathbf{W_a} = \mathbf{I}$ in (2.13), and the solution to (2.13) is just the least squares estimator. To overcome the limitations of least squares estimation described in Section 2.1, Huber introduced the Huber minimax function or Huber function (Figure 2.3)

$$\psi_b(x) = \min\{b, \max\{x, -b\}\}, \tag{2.17}$$

where b, called the cutoff value, is a constant chosen to limit the influence of the outliers on the resulting estimates. In fact, since $\lim_{b\to\infty} \psi_b(x) = x$, resulting in the LS estimate, b is typically chosen between 1 and 2.

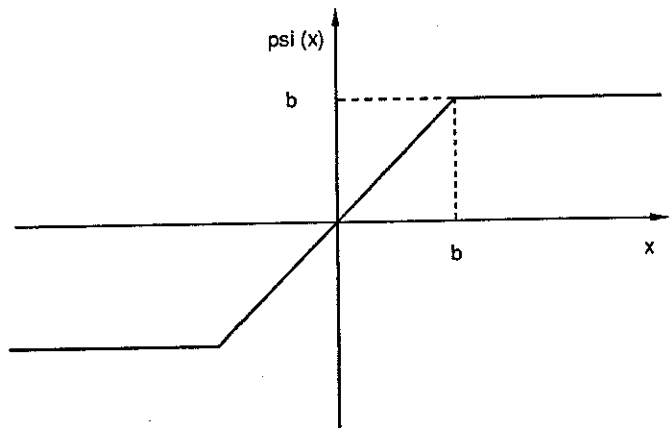

Figure 2.3: Huber ψ-function.

When ψ is bounded, the influence of large errors (potential outliers) is limited by the weights $w(i,j)$ as defined in (2.10). This idea has been extended with the use of redescending ψ-functions like the Hampel ψ-function (Figure 2.4)

$$\psi(x) = \begin{cases} x, & \text{if } -a \le x \le a, \\ a, & \text{if } a \le |x| \le b, \\ \dfrac{a(x-c)}{b-c}, & \text{if } b \le |x| \le c, \\ 0, & \text{otherwise,} \end{cases} \tag{2.18}$$

or the bisquare (also called Tukey's biweight) ψ-function

$$\psi_c(x) = \begin{cases} x\left(1 - \frac{x^2}{c^2}\right)^2, & \text{if } -c \le x \le c, \\ 0, & \text{otherwise} \end{cases}, \tag{2.19}$$

which will discard some data completely, if $|x| > c$.

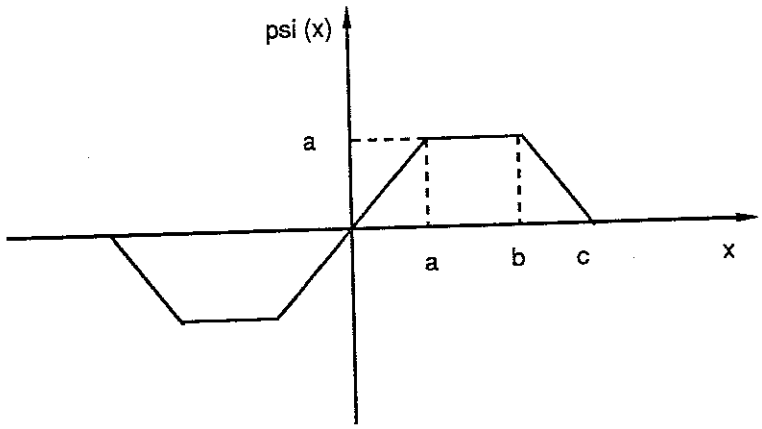

Figure 2.4: Hampel redescending ψ-function.

Since redescending ψ-functions cannot assure convergence of the IRLS algorithm to a solution, a few iterations are performed with a Huber function to approach a solution, and then further iterations are completed with a redescending ψ-function. Figure 2.5 displays three weight functions, used in (2.13): a Huber (solid line) ψ-function with $b = 1.5$, a Hampel (dashed line) ψ-function with $a = 1.5$, $b = 3$, $c = 4.5$ and a bisquare ψ-function (dot-dashed line) with $c = 4.1$.

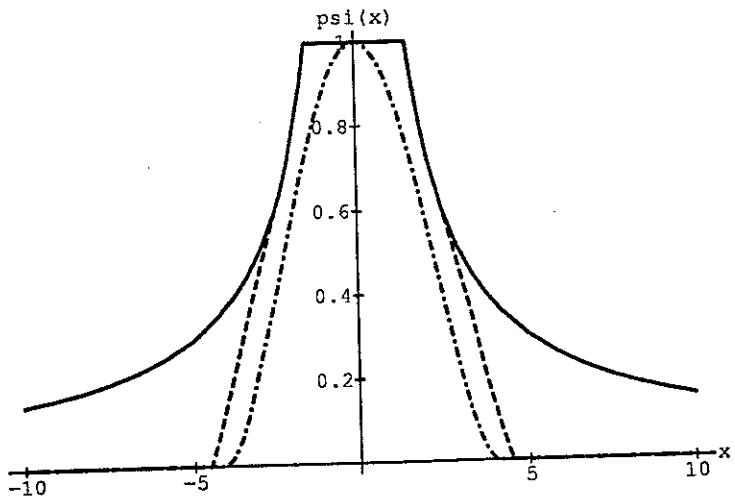

Figure 2.5: different weight functions.

2.4. M-estimation in computer vision applications.

Digital images are usually arrays of integers in a small range (0 to 255 is common), and differ qualitatively from the typical data for which M-estimation techniques were developed. A zero MAD for data sets from other scientific disciplines is very rare, but can easily happen for image data if the fit is perfect for at least half the data in the window, which is likely for low noise images. A zero MAD makes the weights undefined, so the standard M-estimation algorithm must be logically extended to deal with zero MAD. It can even happen that distinct polynomial fits of different orders all have zero MAD for a given window, and deciding which of these "perfect" fits to use is nontrivial. The complete details for dealing with zero MADs are in Section 4.

Another difference is that in statistics the quality of the result is fundamental, whereas in computer vision there is a trade off between the quality of the result and the speed at which the result is obtained. Vision applications cannot afford to iterate till convergence during IRLS M-estimation, hence only about 4 iterations per estimation are performed. In any event, the precision of the coefficients a_i does not need to be high as the predicted pixel values, computed through (2.6), are rounded to an integer between 0 and 255.

3. Tuning M-estimation.

3.1. Choice of the ψ-function.

In this section and the next two (Sections 4 and 5), we discuss methods to improve our surface estimation algorithm [4]. This algorithm works as follow: for every pixel x in the image to process, one considers the pixels enclosed in a window centered on this pixel x, three different surface models are fit to the data in this window using IRLS M-estimation, one chooses the surface model $f_{\mathbf{a}}(r,c)$, which "best" fits a majority of pixels, then the new value of the pixel x is estimated as $f_{\mathbf{a}}(0,0)$. The three fitted models are the constant, planar and quadratic surfaces, that is, polynomials in two variables of order 0,1 and 2. In section 4 we elaborate more on these three models and on the choice of the "best" model. In this section, we focus on adjusting the IRLS M-estimation part of the algorithm.

It is recommended in IRLS M-estimation, when using a redescending ψ-function, to begin the iterations with a Huber ψ-function (a nonredescending ψ-function) to localize a solution and then use a redescending ψ-function to converge faster to the solution [5]. However, the number of iterations with the Huber ψ-function, the type of redescending ψ-function and the number of iterations with this ψ-function are parameters that can be adjusted for a particular application. A Monte Carlo study is performed to determine these "optimum" parameters for our surface estimation algorithm.

In this study two families of redescending ψ-functions have been tested: the Hampel and the bisquare ψ-functions. For each family, different cutoff values are used. These different functions are illustrated in Figure 3.1. The results, presented in the Tables 3.1 to 3.4, have been computed to determine the best combination of iteration numbers and ψ-functions.

3.2. Experiment protocol.

The Monte Carlo study is conducted as follows: we begin with a noiseless image, displayed in Figure 3.2, and on each Monte Carlo trial, alter its appearance by adding noise, random values generated from the ϵ-contaminated normal distribution of the form

$$(1-\epsilon)N(0,\sigma_1^2)+\epsilon N(0,\sigma_2^2). \tag{3.1}$$

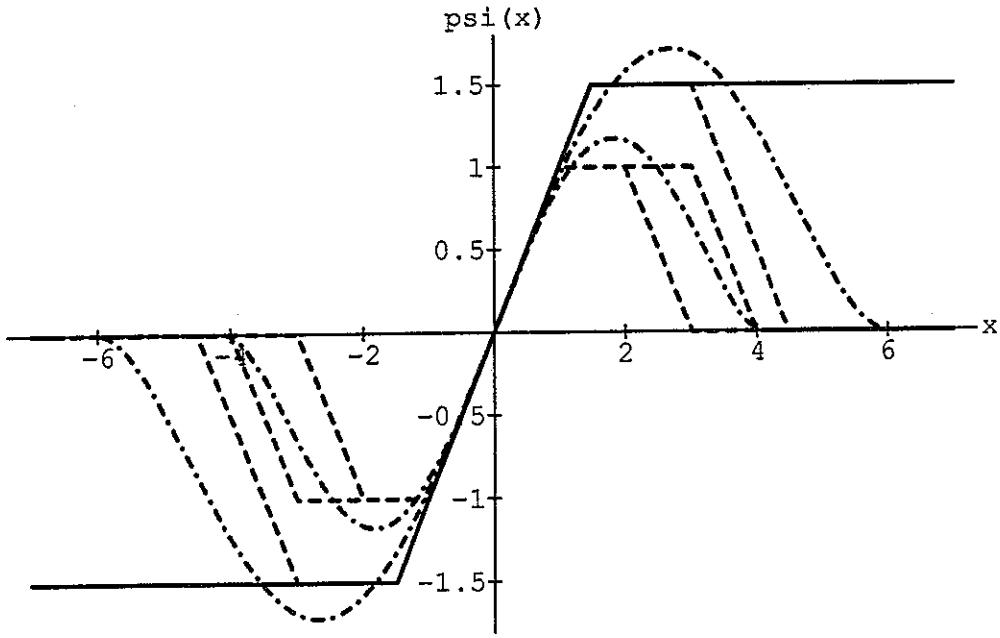

Figure 3.1: ψ-functions used in the Monte Carlo study.

Legend	
Solid line curve: Huber function ($Huber(x, 1.5)$)	
Dashed line curves: Hampel functions	
$Hampel(x, 1, 2, 3)$	**type 0**
$Hampel(x, 1.5, 3, 4.5)$	**type 1**
$Hampel(x, 1, 3, 4)$	**type 2**
Dot-dashed line curves: Bisquare functions	
$Bisquare(x, 4.1)$	**type 3**
$Bisquare(x, 6.0)$	**type 4**

Here, $N(0, \sigma^2)$ represents a normal distribution with mean of 0 and a variance of σ^2. Thus, with probability $1 - \epsilon$, the noise is an observation from $N(0, \sigma_1^2)$, and with probability ϵ, the noise is an observation from $N(0, \sigma_2^2)$. The ϵ-contaminated normal distribution is widely used and recognized as adequate for modeling noise in images [43]. In the following experiments, $\epsilon = 5\%$, σ_1 is set successively to: 3, 5 and 10, representing low, medium, and high noise levels, and $\sigma_2 = 20$. The noiseless image, first introduced by Besl et al. [4], was chosen because of its interesting features including step and roof edges and adequate representation of industrial range images.

The results of each Monte Carlo trial for each estimation procedure are evaluated by using two goodness of fit measures: the MAE (Mean Absolute Error) computed as

$$MAE = \frac{\sum_{i,j} |x_{i,j} - \hat{x}_{i,j}|}{N} \tag{3.2}$$

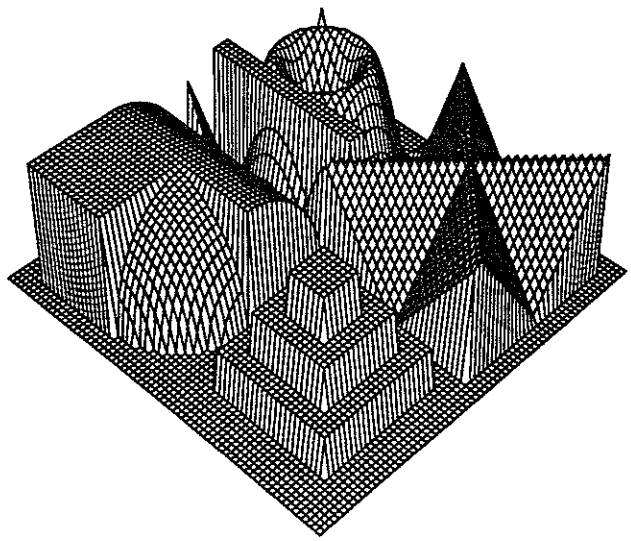

Figure 3.2: noiseless image (72×72).

and the RMS (Root Mean Square) computed as

$$RMS = \frac{\sqrt{\sum_{i,j}(x_{i,j} - \hat{x}_{i,j})^2}}{N} \quad , \tag{3.3}$$

where the sums are all over all processed pixels in the images, N is the number of processed pixels, $x_{i,j}$ is the original pixel value and $\hat{x}_{i,j}$ is the pixel value after filtering (processing). The pixels on the border cannot be processed as a window cannot be centered on these pixels and postprocessing is needed to take care of them (Section 5 describes how postprocessing can be done). The motivation behind the use of two different goodness of fit measures is that MAE is less sensitive to an occasional poor fit at a pixel than RMS, which squares the errors, thereby increasing the effect of large errors. On the other hand, RMS is the measure traditionally used in the vision research community. Average values for MAE and RMS, $\overline{MAE}$ and $\overline{RMS}$, respectively, are obtained by averaging these quantities over 50 Monte Carlo trials. The standard errors, $SE(\overline{MAE})$ and $SE(\overline{RMS})$, defined by

$$SE(\bar{x}) = \sqrt{\frac{\sum_{i=1}^{n_e}(x_i - \bar{x})^2}{(n_e - 1)n_e}}, \qquad \text{where } \bar{x} \text{ is the mean of the } x_i, \tag{3.4}$$

are also computed. They indicate the dispersion of $\overline{MAE}$ and $\overline{RMS}$. Here, n_e is the number of trials in the Monte Carlo study. Generally to decrease the standard error, the number of trials n_e is increased. We had to limit n_e to 50 to keep the computational time within reasonable bounds.

3.3. Conclusion.

In each table, the Original MAE or Original RMS is indicated, it corresponds to the MAE or RMS, respectively, between the noiseless image and the image corrupted by noise before processing. The MAE results for a 3×3 window of this experiment are shown in Table 3.1 which displays the $\overline{MAE}$, $SE(\overline{MAE})$ for each type of ψ-function (types given in Figure 3.1) for varying values of iteration number. A quick conservative method to compare one ψ-function by iteration combination with another can be made by checking whether 95% confidence intervals ($\bar{X} \pm 2\,SE(\bar{X})$) between combinations overlap. As shown in Table 3.1, the best parameter combination is 2 iterations with Huber ψ-function and 2 iterations with ψ-function of type 4. Its confidence interval is overlapping with the interval of the second best (2 iterations with Huber ψ-function and 2 iterations of ψ-function type 1) only for the case $\sigma_1 = 3$. Table 3.2 is similar to Table 3.1 except that average RMS across ψ-function by iteration are displayed. The two "bests" are the same in Table 3.2, but there is an overlap of the confidence intervals in all cases ($\sigma_1 = 3$, $\sigma_1 = 5$ and $\sigma_1 = 10$). Tables 3.3 and 3.4 repeat the information contained in Tables 3.1 and 3.2, respectively, using 5×5 windows. The two "bests" in Table 3.3 are 2 iterations with Huber and 2 iterations with respectively ψ-functions type 2 and type 3. In Table 3.4, the "winners" are 2 iterations with Huber ψ-function, 2 iterations with ψ-function type 4, and 3 iterations with Huber ψ-function, 1 iteration with ψ-function type 4. The results across the four tables also point out that the values obtained with a 5×5 window are always worse than the one computed with a 3×3 window. This can be explained by the size of some image features, which are only 3 pixels large, and that a 5×5 window processing tends to smooth regions more than a 3×3 window processing. The fact that the confidence intervals of RMS values are 2 to 5 times larger than those for MAE values show the higher sensitivity of the RMS measure to bad pixel estimates.

Overall, the best combination is 2 iterations of Huber followed by 2 iterations of ψ-function type 4 i.e., a bisquare function. It is important to notice that the results are quite close, as shown by the overlapping of confidence intervals, hence the influence of the choice of the number of iterations and ψ-function is negligible on the performance of our algorithm. As a consequence, trying other ψ-functions or increasing the number of trials n_e to narrow the confidence intervals seems useless. Other factors have to be studied in order to improve our surface estimation algorithm.

Table 3.1: averaged MAEs and standard errors over 50 trials. The window size is 3x3.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original MAE	2.422	3.600	6.475
Huber iterations: 2 Redescending function type: 0 Redescending function iterations: 2	2.295 ±0.007	3.169 ±0.009	5.209 ± 0.014
Huber iterations: 2 Redescending function type: 1 Redescending function iterations: 2	2.186 ±0.006	2.987 ±0.007	4.916 ±0.012
Huber iterations: 2 Redescending function type: 2 Redescending function iterations: 2	2.243 ± 0.007	3.086 ±0.008	5.095 ±0.013
Huber iterations: 2 Redescending function type: 3 Redescending function iterations: 2	2.231 ±0.007	3.070 ±0.007	5.062 ±0.012
Huber iterations: 2 Redescending function type: 4 Redescending function iterations: 2	2.168* ±0.006	2.954* ±0.008	4.856* ±0.011
Huber iterations: 3 Redescending function type: 1 Redescending function iterations: 1	2.230 ±0.007	2.995 ±0.008	4.875 ±0.013
Huber iterations: 3 Redescending function type: 3 Redescending function iterations: 1	2.243 ±0.007	3.028 ±0.009	4.900 ±0.012
Huber iterations: 3 Redescending function type: 4 Redescending function iterations: 1	2.222 ±0.007	2.979 ±0.008	4.860 ±0.012
Huber iterations: 4 Redescending function type: Redescending function iterations: 0	2.257 ±0.007	3.047 ±0.009	4.917 ±0.013

* denotes the minimum value of MAE.

Table 3.2: averaged RMSs and standard errors over 50 trials. The window size is 3x3.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original RMS	4.605	5.733	9.281
Huber iterations: 2 Redescending function type: 0 Redescending function iterations: 2	9.275 ±0.062	9.515 ± 0.064	10.696 ±0.058
Huber iterations: 2 Redescending function type: 1 Redescending function iterations: 2	9.009 ±0.043	9.032 ±0.046	10.138 ±0.047
Huber iterations: 2 Redescending function type: 2 Redescending function iterations: 2	9.193 ±0.058	9.290 ±0.056	10.474 ±0.053
Huber iterations: 2 Redescending function type: 3 Redescending function iterations: 2	9.163 ±0.060	9.246 ±0.055	10.410 ±0.054
Huber iterations: 2 Redescending function type: 4 Redescending function iterations: 2	8.921* ±0.043	8.901* ±0.047	10.031* ±0.055
Huber iterations: 3 Redescending function type: 1 Redescending function iterations: 1	9.254 ±0.061	9.151 ±0.056	10.169 ±0.055
Huber iterations: 3 Redescending function type: 3 Redescending function iterations: 1	9.293 ±0.061	9.267 ±0.059	10.257 ±0.055
Huber iterations: 3 Redescending function type: 4 Redescending function iterations: 1	9.196 ±0.060	9.036 ±0.057	10.090 ±0.057
Huber iterations: 4 Redescending function type: Redescending function iterations: 0	9.344 ±0.063	9.368 ±0.058	10.392 ±0.051

* denotes the minimum value of RMS.

Table 3.3: averaged MAEs and standard errors over 50 trials. The window size is 5x5.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original MAE	2.422	3.600	6.475
Huber iterations: 2 Redescending function type: 0 Redescending function iterations: 2	4.453 ± 0.010	5.126* ±0.012	7.217 ±0.021
Huber iterations: 2 Redescending function type: 1 Redescending function iterations: 2	4.768 ±0.011	5.458 ±0.013	7.546 ±0.019
Huber iterations: 2 Redescending function type: 2 Redescending function iterations: 2	4.525* ±0.010	5.173 ± 0.010	7.160 ±0.019
Huber iterations: 2 Redescending function type: 3 Redescending function iterations: 2	4.550 ±0.010	5.165 ±0.011	7.142* ±0.019
Huber iterations: 2 Redescending function type: 4 Redescending function iterations: 2	4.847 ±0.011	5.577 ±0.014	7.491 ±0.020
Huber iterations: 3 Redescending function type: 1 Redescending function iterations: 1	4.879 ±0.009	5.585 ±0.013	7.618 ±0.019
Huber iterations: 3 Redescending function type: 3 Redescending function iterations: 1	4.851 ±0.008	5.454 ±0.010	7.347 ±0.018
Huber iterations: 3 Redescending function type: 4 Redescending function iterations: 1	4.807 ±0.009	5.574 ±0.014	7.460 ±0.020
Huber iterations: 4 Redescending function type: Redescending function iterations: 0	4.965 ±0.009	5.586 ±0.010	7.604 ±0.019

* indicates the minimum value for MAE.

Table 3.4: averaged RMSs and standard errors over 50 trials. The window size is 5x5.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original RMS	4.605	5.733	9.282
Huber iterations: 2 Redescending function type: 0 Redescending function iterations: 2	16.789 ±0.050	16.621 ± 0.038	17.262 ± 0.041
Huber iterations: 2 Redescending function type: 1 Redescending function iterations: 2	17.119 ± 0.062	16.692 ± 0.042	17.262 ± 0.038
Huber iterations: 2 Redescending function type: 2 Redescending function iterations: 2	17.040 ± 0.056	16.681 ± 0.036	17.050 ± 0.038
Huber iterations: 2 Redescending function type: 3 Redescending function iterations: 2	17.099 ± 0.061	16.699 ± 0.039	17.050 ± 0.038
Huber iterations: 2 Redescending function type: 4 Redescending function iterations: 2	17.026 ± 0.070	16.515 ± 0.048	16.895 ± 0.054
Huber iterations: 3 Redescending function type: 1 Redescending function iterations: 1	16.896 ± 0.040	16.736 ± 0.038	17.315 ± 0.039
Huber iterations: 3 Redescending function type: 3 Redescending function iterations: 1	16.895 ± 0.038	16.678 ± 0.037	17.050 ± 0.036
Huber iterations: 3 Redescending function type: 4 Redescending function iterations: 1	16.550* ± 0.046	16.390* ± 0.045	16.782* ± 0.052
Huber iterations: 4 Redescending function type: Redescending function iterations: 0	17.318 ± 0.034	17.196 ± 0.033	17.315 ± 0.034

* indicates the minimum value for RMS.

4. Variable order and M-estimation based paradigms.

4.1. Variable order estimator.

When estimating data in a window either an unique model is used or several models are considered; in the latter case the estimates are computed using the model that "best" fits the data in the window. The time complexity of a surface estimation approach is roughly the number of fitted models times the complexity of the estimation algorithm (IRLS M-estimation or least median of squares, described in Section 6). The time complexity of IRLS M-estimation is $O(np^2)$, whereas the complexity of least median of squares is $O(n^{p+1}\log n)$, where n is the number of data and p the number of parameters [30]. The high complexity of least median of squares limits its use in surface estimation, so generally only one model (planar) is fit [29]. With IRLS M-estimation several models can be used, which gives more degrees of freedom to model the data, hence increasing the quality of the estimates. In our approach, based on IRLS M-estimation, three models are used: constant, planar and quadratic. A constant fit could be seen as just a particular case of planar fit, but it is not. A constant fit can be computed by either a zero order M-estimator or a median estimator, both of which have a 0.5 breakdown value, whereas the breakdown value of a planar M-estimator is only 0.25. This difference is fundamental when detecting (estimating) step edges, since with a constant fit a step edge is recovered if it occupies at least 50% of the window, and with a planar fit if it occupies at least 75% of the window. The same kind of argument prevails between a planar and a quadratic model, since the breakdown value of a quadratic M-estimator is only $1/p = 1/6$. There are limitations on the number of models one can fit to the window data. Obviously the computational time is one limitation, and as the order (or the number of parameters) of the model increases, the window size gets bigger, since the relation

$$nm - n_o \geq p, \tag{4.1}$$

where nm is the number of pixels in the window, n_o the number of outliers, and p the number of parameters, has to be satisfied for each model to be estimated (the number of equations must equal the number of unknowns).

The difficulty in a multiple model paradigm is determining which of the models will yield the best estimates for the data. For this purpose, we need to measure the difference between the window data and the data estimates; the outliers, data which don't follow the fitting model, should be downweighted. For example, if a majority of pixels in a window are on a plane, we don't want to consider the one or two pixels not on this plane when evaluating the quality of the estimates computed with a planar model. We have used the following measures to assess the goodness of fit:

- $MAD = 1.4286\,\text{median}|e_{\mathbf{a}}(r,c)|$, which is a good robust estimator of the standard deviation if the data sample is large enough and follows a normal distribution. The first assumption is not realized for window data but this operator is computationally inexpensive.
- $\gamma_p = \sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} w_{\mathbf{a}}(r,c)$, which measures the consistency of the fit with the data. If the fit is perfect all the weights $w_{\mathbf{a}}(r,c) = 1$ and $\gamma_p = nm$, the window size. This measure is used in conjunction with the MAD. A zero MAD, which is particular to computer vision, means that the fit is perfect for at least half the data in the window. If two different models yield a MAD of zero, γ_p is used to select the model which has the best fit for the largest number of pixels. It is also used to decide between two fits with the same η_p— the fit with the largest γ_p is chosen.

- $\eta_p = \dfrac{n^2 m^2 s^2 \sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} \psi^2(e_a(r,c)/s)}{(nm-p)\left(\sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} \psi'(e_a(r,c)/s)\right)^2}$, where s is a scale estimate, which is an excellent robust estimator of the variance.

- $WRMS = \sqrt{\dfrac{\sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} w_a(r,c) e_a^2(r,c)}{\left(\sum_{r=-\hat{n}}^{\hat{n}} \sum_{c=-\hat{m}}^{\hat{m}} w_a(r,c)\right) - p}}$, where $w_a(r,c) = \dfrac{\psi(e_a(r,c)/s)}{e_a(r,c)/s}$, which is also a good robust estimator of the standard deviation.

The problem with the η_p and the $WRMS$ (weighted root mean square) measures is that they require a scaling factor s, an estimator of the standard deviation, which is what they are estimating. The MAD is used as a quick and dirty estimator of s in the computations of η_p and $WRMS$. As in [4], the median is used for the constant fit because it is faster than the zeroth order M-estimator and produces comparable results. Weights $w_a(r,c)$ for the constant fit are computed from (2.10) with the bisquare.

Using these different estimators of the standard deviation, we have built five paradigms to determine which model is the best for the current window:

1. If the MAD resulting from the constant fit is less than a threshold, use the constant fit, otherwise select the best (η_p-wise) of the planar and quadratic fits (the model which yields the smallest η_p is selected). In this paradigm, if a constant fit is detected, no M-estimation is performed (the planar and quadratic models are not fit).
2. Paradigm 1 drops the constant fit if the threshold is exceeded, but it seems that even if this happens the constant fit can still be better than the two others. Hence Paradigm 2 is: if the MAD is less than a threshold, use the constant fit, otherwise select the best of the constant, planar and quadratic fits (η_p-wise, MAD^2 is used to approximate η_p for the constant fit). In case of a tie give the preference to the lowest order.
3. Since a MAD of zero or smaller than a threshold means that the fit is perfect or very good for at least half the data, in an attempt to decrease the complexity of our surface estimation paradigm, M-estimation of a model is performed only if the previous model doesn't have a small MAD. If any of the fitted MADs are smaller than a threshold in the order constant, planar and quadratic, use this fit. Otherwise choose the best of the three fits (η_p-wise, MAD^2 is used to approximate η_p for the constant fit). For example, if the MAD of the planar fit is below the threshold, the M-estimation of the quadratic model is not performed. This paradigm looks also for the lowest order model.
4. An alternative to the η_p estimator is the $WRMS$ estimator, hence a simple scheme: compute the $WRMS$ of each fit and select the fit with the smallest $WRMS$, in case of a tie give the preference to the lowest order.
5. This paradigm tries to override the preference for the lowest or the highest order. For each fit (constant, planar and quadratic) compute the MAD. If the MAD is zero, set $\eta_p = 0$ and compute γ_p as the number of points with a perfect fit. Otherwise, evaluate η_p and compute γ_p as the consistency of fit across all pixels in the window. The chosen fit is the one with the smallest η_p and the largest γ_p, in case of ties.

4.2. Experiments.

Monte Carlo studies, using the protocol described in Section 3.2, have been run to determine which paradigm works best. For all the runs, two iterations with the Huber function followed by two iterations with the bisquare function ($C_b = 6.0$) have been performed during each IRLS M-estimation. Tables 4.1 to 4.4 show the MAE and RMS measures, results of these Monte Carlo studies, using Paradigms 1 to 5. In the last row of each table, the results of Paradigm 5 with no estimation of the quadratic model are shown. From the four tables (Tables 4.1 to 4.4), Paradigm 5 without a quadratic model produces worse results, by a large margin, than all the other paradigms, proving the usefulness of a multiple order paradigm.

As displayed in Table 4.1, Paradigm 1 yields the best MAEs, followed by Paradigm 4; the results of Paradigms 2, 3 and 5 are close. The classification of the paradigms based on RMS, established from Table 4.2, is the same as that from Table 4.1. However Paradigms 1 and 4 yield results twice as good as those of Paradigms 2, 3 and 5, which are even worse than the original (before processing) values. This is certainly due to the sensitivity of the RMS measure to very poor estimates at a few pixels. Hence Paradigm 1 works the best for 3×3 window processing. With a window size of 5×5, the MAE measure indicates (Table 4.3) that Paradigms 2, 3 and 5 give relatively good results whereas Paradigms 1 and 4 work poorly. The RMS measure distinguishes less clearly the different paradigms, but Paradigms 1 and 5 still yield the best results. Combining the MAE results with the RMS results, the conclusion is that Paradigm 5 produces the best results with 5×5 window processing. Since we don't want a different paradigm type for each window size, we will use the postprocessing results (Section 5) to choose between Paradigm 1 and Paradigm 5.

The results of processing with Paradigm 5 are shown in Figures 4.1 to 4.10 in order to illustrate the effect of 3×3 window versus 5×5 window processing. In all the processed images (Figures 4.2, 4.3, 4.5, 4.6, 4.8, 4.9, 4.11 and 4.12) the corners are chopped; this effect is much more noticeable with a 5×5 window than with a 3×3 window. 3×3 window processing preserves the geometric features of the input image better than 5×5 window processing, however, 5×5 window processing is much more efficient for smoothing noisy peaks and holes. This is the most clearly illustrated with the high noise input image (Figures 4.11 and 4.12).

Table 4.1: averaged MAEs and standard errors over 50 trials. The window size is 3×3.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original MAE	2.422	3.600	6.475
Paradigm 1	1.865 * ±0.007	2.714 * ±0.008	4.737 * ±0.012
Paradigm 2	2.168 ±0.006	2.973 ±0.008	4.840 ±0.010
Paradigm 3	2.155 ±0.003	2.967 ±0.007	4.836 ±0.010
Paradigm 4	2.127 ±0.007	2.896 ±0.008	4.814 ±0.012
Paradigm 5	2.168 ±0.006	2.954 ±0.008	4.856 ±0.013
Paradigm 5 No quadratic model	3.919 ±0.007	4.566 ±0.011	6.522 ±0.019

* denotes the minimum value of MAE.

Table 4.2: averaged RMSs and standard errors over 50 trials. The window size is 3×3.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original RMS	4.605	5.733	9.281
Paradigm 1	3.926 * ±0.111	4.638 * ±0.073	7.233* ±0.072
Paradigm 2	9.027 ±0.046	9.158 ±0.046	10.212 ±0.052
Paradigm 3	8.890 ±0.036	9.090 ±0.040	10.164 ±0.047
Paradigm 4	4.075 ±0.066	4.800 ±0.052	7.108 ±0.038
Paradigm 5	8.921 ±0.043	8.901 ±0.047	10.031 ±0.055
Paradigm 5 No quadratic model	12.468 ±0.021	12.700 ±0.025	14.131 ±0.031

* denotes the minimum value RMS.

Table 4.3: averaged MAEs and standard errors over 50 trials. The window size is 5×5.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original MAE	2.422	3.600	6.475
Paradigm 1	7.296 ±0.009	7.891 ±0.009	9.273 ±0.014
Paradigm 2	4.909 ±0.010	5.500 * ±0.011	7.354 * ±0.018
Paradigm 3	4.928 ±0.011	5.503 ±0.011	7.354* ±0.018
Paradigm 4	8.378 ±0.007	8.748 ±0.008	9.689 ±0.013
Paradigm 5	4.847 * ±0.011	5.577 ±0.014	7.491 ±0.020
Paradigm 5 No quadratic model	8.689 ±0.008	9.405 ±0.017	11.747 ±0.025

* denotes the minimum value of MAE.

Table 4.4: averaged RMSs and standard errors over 50 trials. The window size is 5×5.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original RMS	4.605	5.733	9.282
Paradigm 1	16.915 * ±0.053	16.566 ±0.037	17.108 ±0.025
Paradigm 2	17.392 ±0.057	16.935 ±0.029	17.289 ±0.030
Paradigm 3	17.527 ±0.065	16.960 ±0.025	17.289 ±0.030
Paradigm 4	17.244 ±0.014	17.411 ±0.012	17.967 ±0.014
Paradigm 5	17.026 ±0.070	16.515* ±0.048	16.895* ±0.054
Paradigm 5 No quadratic model	21.709 ±0.025	21.867 ±0.023	22.980 ±0.032

* denotes the minimum value of RMS

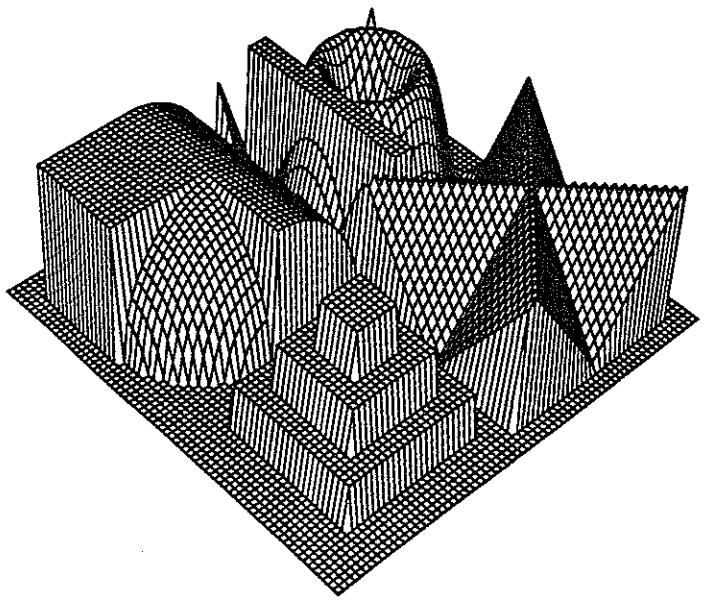

Figure 4.1: noiseless image.

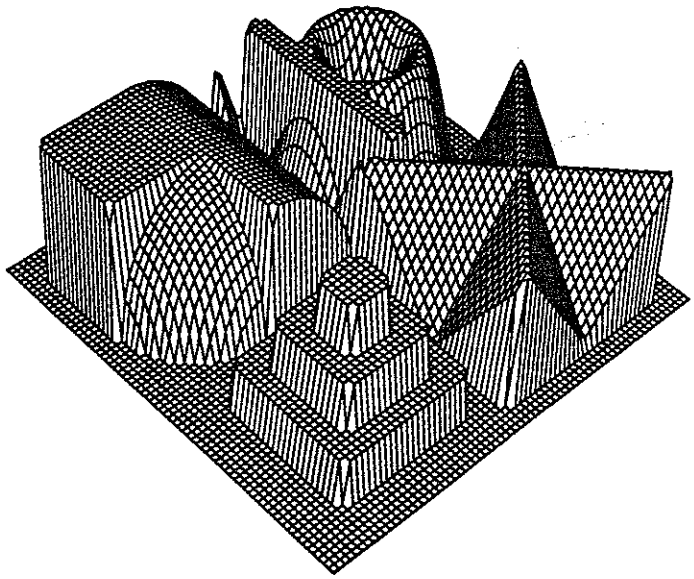

Figure 4.2: noiseless image processed (3×3 window).

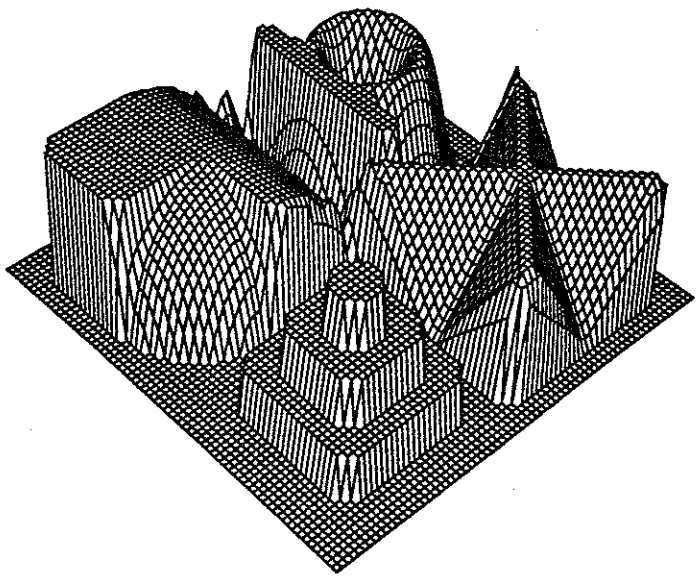

Figure 4.3: noiseless image processed (5×5 window).

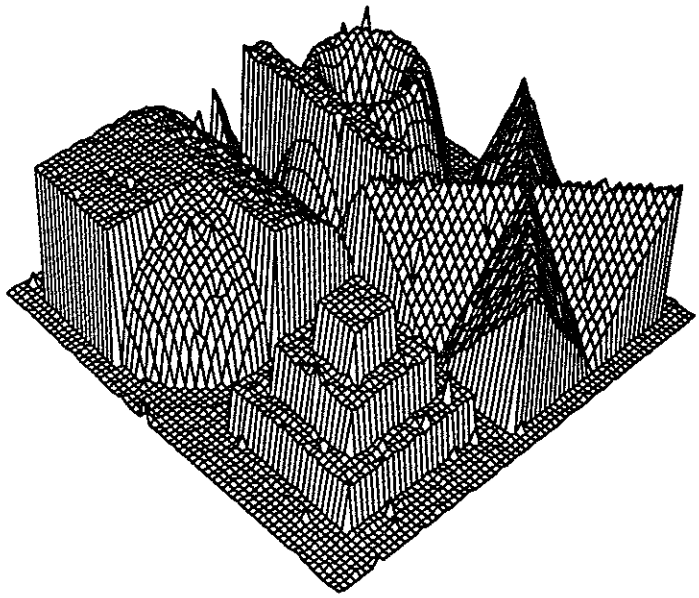

Figure 4.4: low noise image.

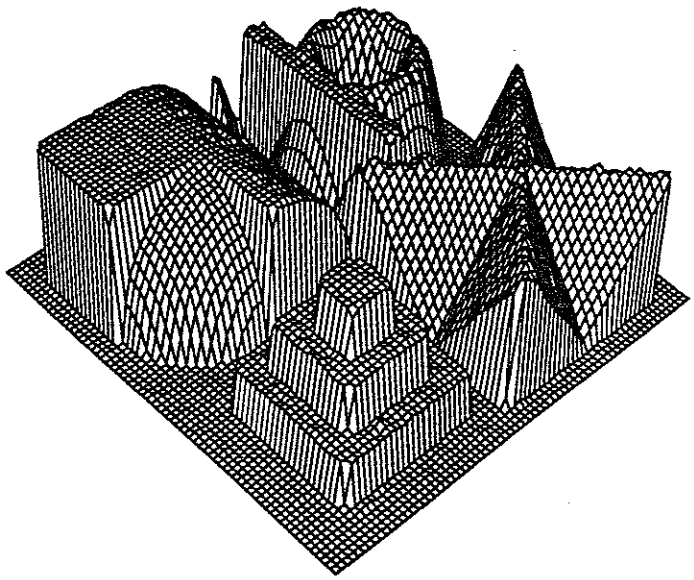

Figure 4.5: low noise image processed (3×3 window).

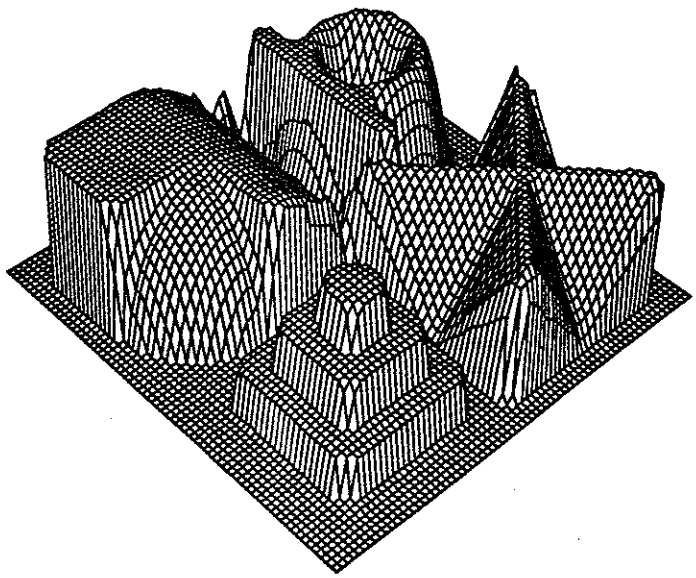

Figure 4.6: low noise image processed (5×5 window).

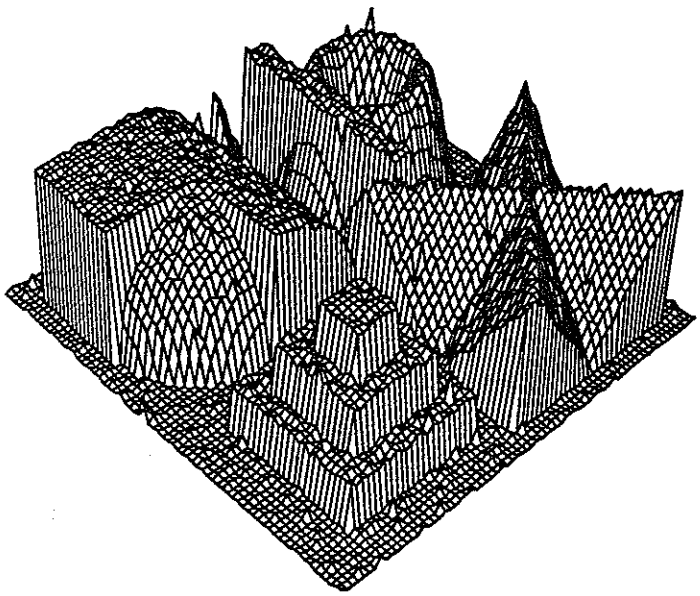

Figure 4.7: medium noise image.

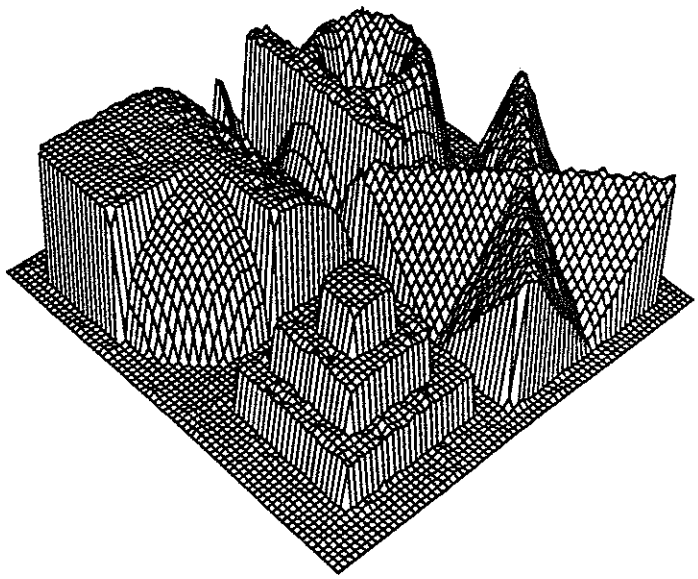

Figure 4.8: medium noise image processed (3×3 window).

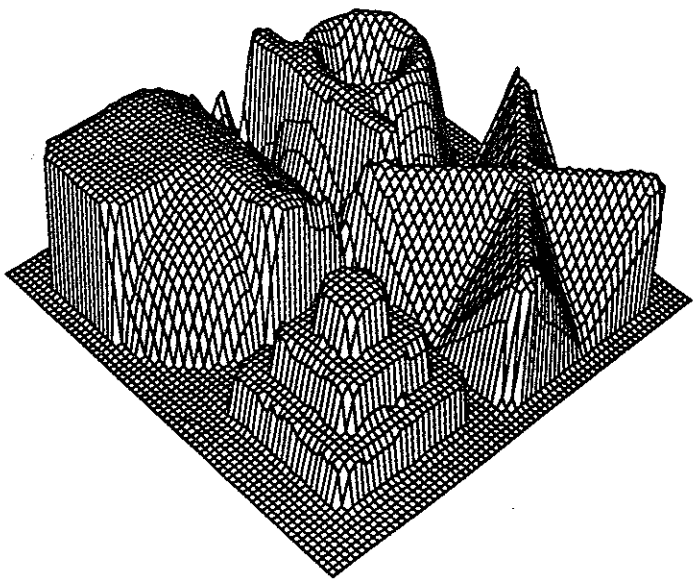

Figure 4.9: medium noise image processed (5×5 window).

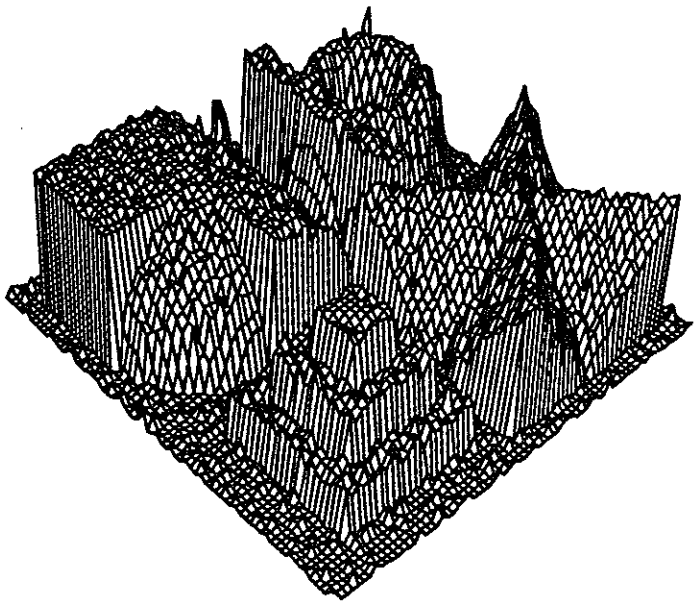

Figure 4.9: high noise image.

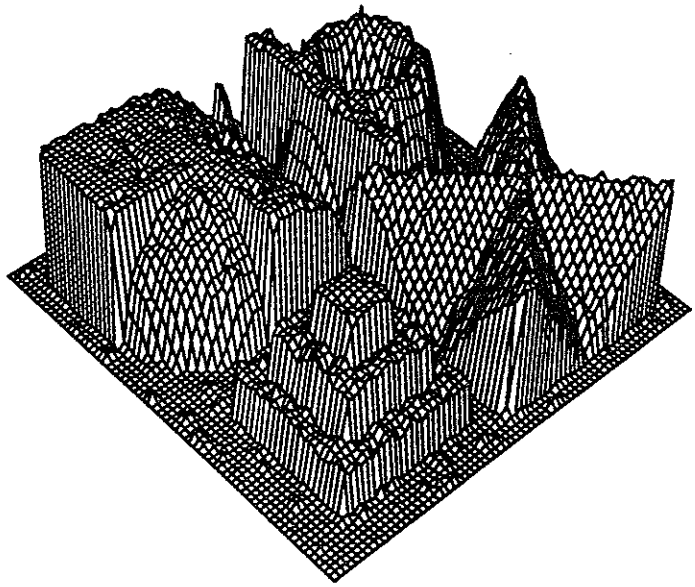

Figure 4.10: high noise image processed (3×3 window).

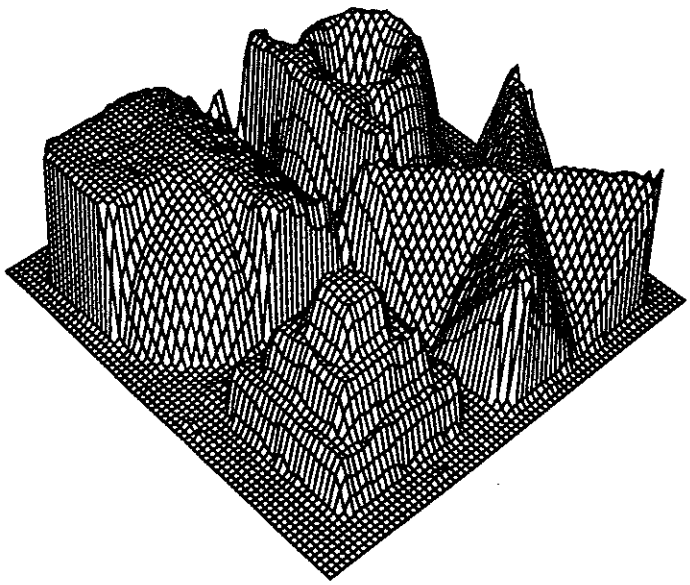

Figure 4.11: high noise image processed (5×5 window).

5. Postprocessing Variable Order Paradigms.

5.1. Postprocessing.

We have seen in the previous section how each pixel x is evaluated using a variable order model on the data enclosed in a window centered on that pixel x. As a consequence, the best fitting model is computed for every possible window in the image. In the spirit of the facet model of Haralick and Watson [17], one can use these results to find the "best" pixel estimate from among all nm windows containing that pixel instead of just considering one centered window. For each window, we have the order and the coefficients of the fit, the weights associated with each pixel in the window and measures of fit goodness, which depend on the paradigm used for processing. The goodness of fit measures yielded by Paradigms 1 and 5 are: MAE, η_p, and γ_p.

An intuitive scheme is to take the best fit among all windows containing a given pixel. Each corner should be estimated properly, since a window can be placed to overlap this corner such that the data enclosed in this window would yield a perfect fit f with no outlier (for ideal data). The fit f would be chosen since all the other windows would have at least one outlier (recall that to select a fit η_p is minimized, and in case of a tie γ_p is maximized). Figure 5.1 illustrates this point, showing the window (bold boundaries) used to obtain the fit f, which will accurately estimate the corner (pixel marked by a cross).

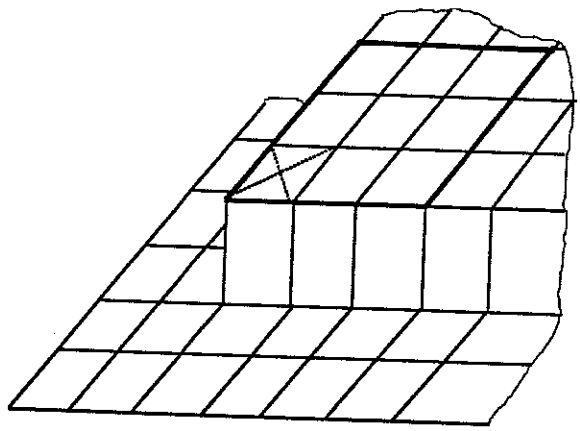

Figure 5.1: the window used to accurately estimate a corner.

Unfortunately, this straightforward approach doesn't work in the presence of noise or if a surface on one side of the discontinuity is not fit as well as the surface on the other side. Consider the profile of a step edge (Figure 5.2): on the step edge right-hand side the horizontal plane is going to be fit "perfectly", $\eta_p = 0$, by a constant model. However, on the left-hand side, if we assume that the surface is sinusoidal, a quadratic model will probably yield a reasonable fit, but certainly a nonzero η_p. Hence when estimating the pixels in the column $0 \leq r \leq 1$, the constant fit will be used (these pixels belong to the sinusoidal surface whose fit has a nonzero η_p, so the perfect fit of the horizontal surface will be preferred even though these pixels are outliers for this horizontal fit), moving the location of the edge to the left. The presence of noise can produce the same problem, altering the location of an edge for the same reasons.

The above example demonstrates the necessity of considering the inlier/outlier information for each pixel. Every pixel is assigned a weight w, whose value ranges from 0 (the pixel is an outlier) to 1 (the pixel is an inlier). However, if a w of 1 clearly indicates an inlier, a pixel with a weight of

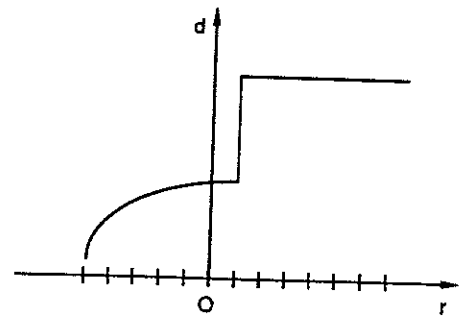

Figure 5.2: step edge.

0.6 could be either classified as an outlier if all the other weights in the window are close to 1, or classified as an inlier if most of the weights are in the range from 0.5 to 0.7. This suggests that an arbitrary threshold cannot be set to classify a pixel as an inlier or outlier according to its weight. It is preferable to avoid this binary decision, since optimizing the threshold value can be extremely difficult. To take into account these weights and yet avoid a binary classification, we introduce the function

$$g_{\mathrm{a}}(r,c) = \frac{\eta_p}{w_{\mathrm{a}}(r,c)^{\beta} + 0.001} \quad , \tag{5.1}$$

where β is a parameter to optimize. The constant 0.001 takes care of the case where the weight $w_{\mathrm{a}}(r,c)$ is zero. Instead of minimizing η_p across all windows containing the given pixel x, which suffers from the problem described above, we minimize $g_{\mathrm{a}}(r,c)$ across all windows containing the pixel x, where r and c are the coordinates of the pixel x in the local system of the window containing x. A low weight $w_{\mathrm{a}}(r,c)$ will increase the value of $g_{\mathrm{a}}(r,c)$. Hence if a pixel is an outlier for a fit (weight close to zero) that is good (small η_p) for all the other pixels, the value of $g_{\mathrm{a}}(r,c)$ will be huge and a fit with a worse η_p but a larger $w_{\mathrm{a}}(r,c)$ will be preferred. Since doubling the value of a weight does not correspond to doubling the η_p — a variation in a weight is much more significant than the same variation in η_p — β is set to an integer greater than one. In case of a tie between $g_{\mathrm{a}}(r,c)$ values, we maximize γ_p. The algorithm described above will be referred to as the *robust variable order facet model* paradigm.

5.2. Experiments.

Monte Carlo studies were run, using the protocol described in Section 3.2, to determine the optimum value of β and to decide between Paradigms 1 and 5. Tables 5.1 to 5.4 show the MAE and RMS measures yielded by postprocessing Paradigms 1 and 5, using two different window sizes for different values of β. Paradigm 1 yielded poor results for both window sizes and for the three noise configurations (Tables 5.1 to 5.4). This is to be expected, since the ability of a paradigm to accurately estimate surfaces is linked to the quality of the model. For the sake of speed, Paradigm 1 first considers the constant fit alone; if this constant model is not selected (the fit MAD exceeds a threshold), then the constant model is dropped and Paradigm 1 decides between the planar and quadratic models. The better results of Paradigm 5 compensate for its higher complexity, and justify its use over Paradigm 1. The remainder of the discussion is focused on Paradigm 5's postprocessing. The results in Tables 5.1 and 5.2 are consistent: $\beta = 2$ is clearly a bad choice. A correlation appears between β and the amount of noise, measured by σ_1: $\beta = 5$ does the best job for $\sigma_1 = 3$, $\beta = 4$ for $\sigma_1 = 5$ and $\beta = 3$ for $\sigma_1 = 10$. On the average, $\beta = 4$ seems to be a good choice for 3×3 window processing. The results in Tables 5.3 and 5.4 confirm that conclusion,

following the same pattern as the results in Tables 5.1 and 5.2. Hence $\beta = 4$ seems to be the best choice.

In order to visually assess the improvement yielded by postprocessing Paradigm 5, we have run our postprocessed surface estimation paradigm on the noiseless image (Figures 5.3 and 5.4), the low noise image (Figures 5.5 and 5.6), the medium noise image (Figures 5.7 and 5.8) and the high noise image (Figures 5.9 and 5.10). The results for the noiseless image shown for both window sizes look perfect: no geometric features of the original image are altered. Without postprocessing, all the corners of the "wedding cake" and the side edges of the cross in the noiseless image were chopped (Figures 4.2 and 4.3). The MAE and RMS measures (Table 5.5) show that 3×3 window processing is nearly perfect and that 5×5 window is still impressive. When noise is added to the noiseless image, the results of the postprocessed variable order paradigm of Section 5.1 (Figures 5.5 to 5.10) show that postprocessing effectively solves the corner preservation and edge location problems: no corners or edges are chopped or moved. 5×5 window processing does a better job than 3×3 window processing in smoothing out the noise in the interiors of large surfaces. This is substantiated by both the pictures and the MAE and RMS measures (Table 5.5).

Table 5.1: averaged MAEs and standard errors over 50 trials. The window size is 3×3.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original MAE	2.422	3.600	6.475
Postprocessed Par. 1 $\beta = 4$	7.448 ± 0.092	6.971 ± 0.079	8.362 ± 0.080
Postprocessed Par. 5 $\beta = 2$	1.960 ± 0.010	2.727 ± 0.010	4.933 ± 0.011
Postprocessed Par. 5 $\beta = 3$	1.832 ± 0.006	2.654 ± 0.007	4.761* ± 0.012
Postprocessed Par. 5 $\beta = 4$	1.816 ± 0.005	2.656* ± 0.007	4.809 ± 0.011
Postprocessed Par. 5 $\beta = 5$	1.815* ± 0.005	2.666 ± 0.007	4.844 ± 0.011

* denotes the minimum value of MAE.

Table 5.2: averaged RMSs and standard errors over 50 trials. The window size is 3×3.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original RMS	4.605	5.733	9.281
Postprocessed Par. 1 $\beta = 4$	34.393 ± 0.313	29.348 ± 0.430	26.137 ± 0.354
Postprocessed Par. 5 $\beta = 2$	5.053 ± 0.127	5.241 ± 0.108	8.751 ± 0.109
Postprocessed Par. 5 $\beta = 3$	3.532 ± 0.075	4.384 ± 0.063	7.243 * ± 0.057
Postprocessed Par. 5 $\beta = 4$	3.377 ± 0.073	4.297* ± 0.064	7.244 ± 0.058
Postprocessed Par. 5 $\beta = 5$	3.366 * ± 0.070	4.336 ± 0.064	7.312 ± 0.060

* denotes the minimum value of RMS.

Table 5.3: averaged MAEs and standard errors over 50 trials. The window size is 5×5.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original MAE	2.422	3.600	6.475
Postprocessed Par. 1 $\beta = 4$	6.151 ± 0.107	6.475 ± 0.119	8.182 ± 0.124
Postprocessed Par. 5 $\beta = 2$	3.304 ± 0.026	3.448 ± 0.022	5.107 ± 0.040
Postprocessed Par. 5 $\beta = 3$	2.578 ± 0.011	3.043 ± 0.012	4.488 ± 0.021
Postprocessed Par. 5 $\beta = 4$	2.521 ± 0.012	3.021* ± 0.013	4.550* ± 0.022
Postprocessed Par. 5 $\beta = 5$	2.517 * ± 0.013	3.032 ± 0.013	4.584 ± 0.022

* denotes the minimum value of MAE.

Table 5.4: averaged RMSs and standard errors over 50 trials. The window size is 5×5.

	$\sigma_1 = 3$	$\sigma_1 = 5$	$\sigma_1 = 10$
Original RMS	4.605	5.733	9.282
Postprocessed Par. 1 $\beta = 4$	29.694 ± 0.373	28.294 ± 0.439	27.676 ± 0.470
Postprocessed Par. 5 $\beta = 2$	13.264 ± 0.182	11.178 ± 0.181	13.408 ± 0.181
Postprocessed Par. 5 $\beta = 3$	8.238 ± 0.132	8.783 ± 0.112	10.438 * ± 0.104
Postprocessed Par. 5 $\beta = 4$	8.100 ± 0.140	8.691* ± 0.118	10.455 ± 0.106
Postprocessed Par. 5 $\beta = 5$	8.083* ± 0.144	8.725 ± 0.115	10.488 ± 0.107

* denotes the minimum value of RMS

Table 5.5: MAE and RMS measures for the postprocessed images (Figures 5.1 to 5.8).

	MAE	RMS
noiseless		
3×3 window	0.496	2.881
5×5 window	1.696	8.112
low noise		
3×3 window	2.063	3.720
5×5 window	1.437	2.974
medium noise		
3×3 window	3.175	4.983
5×5 window	2.130	3.511
high noise		
3×3 window	5.872	8.527
5×5 window	4.041	6.696

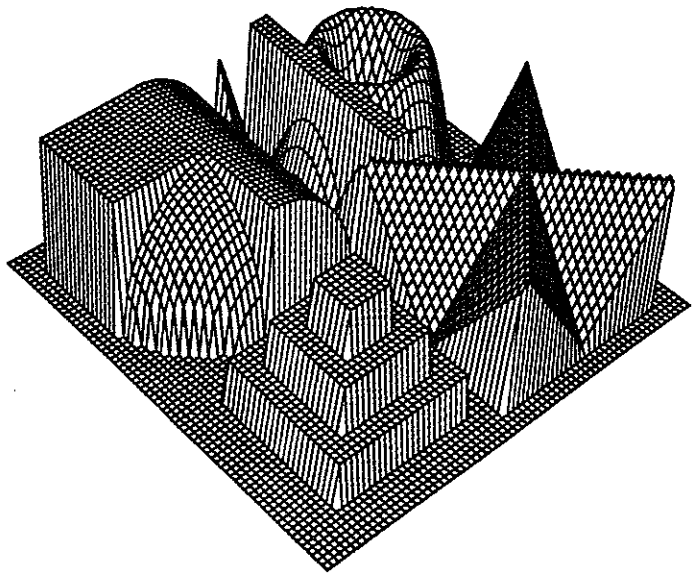

Figure 5.3: noiseless image postprocessed (3×3 window).

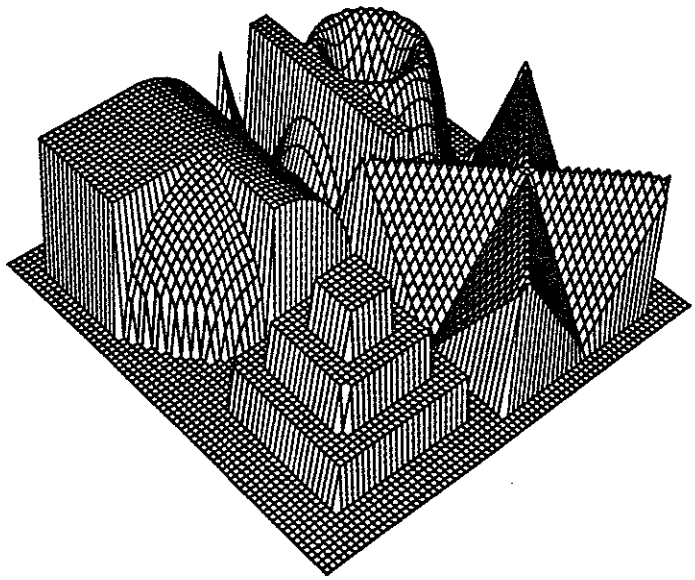

Figure 5.4: noiseless image postprocessed (5×5 window).

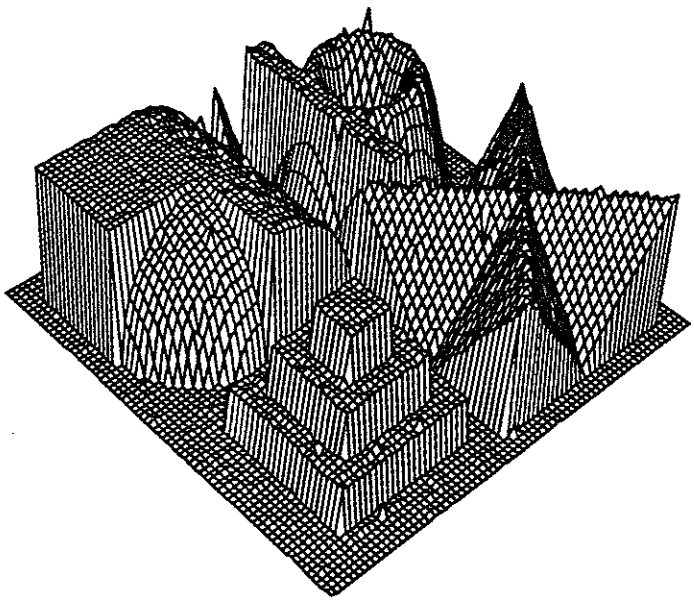

Figure 5.5: low noise image postprocessed (3×3 window).

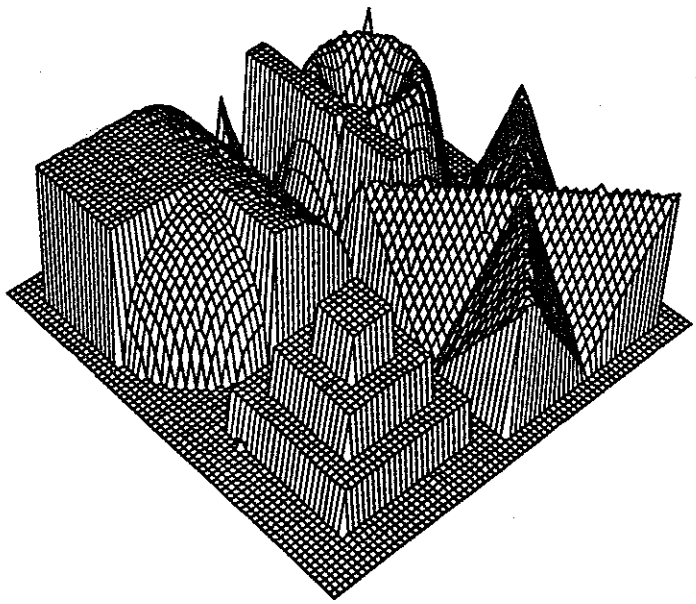

Figure 5.6: low noise image postprocessed (5×5 window).

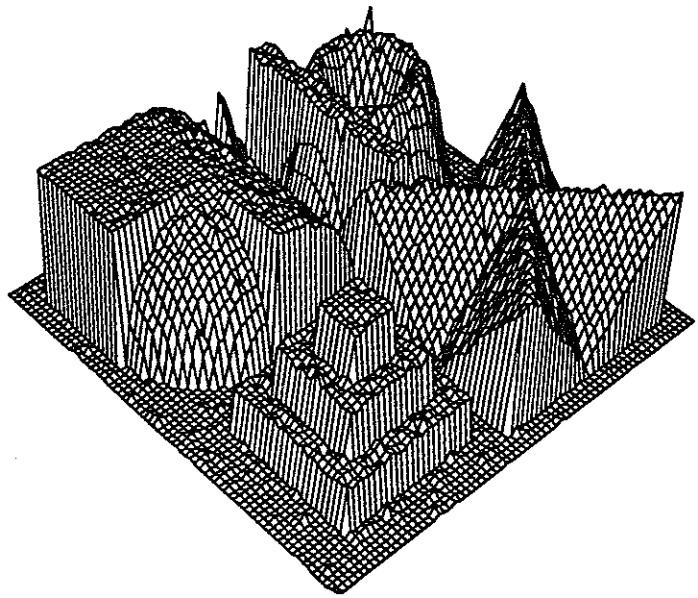

Figure 5.7: medium noise image postprocessed (3×3 window).

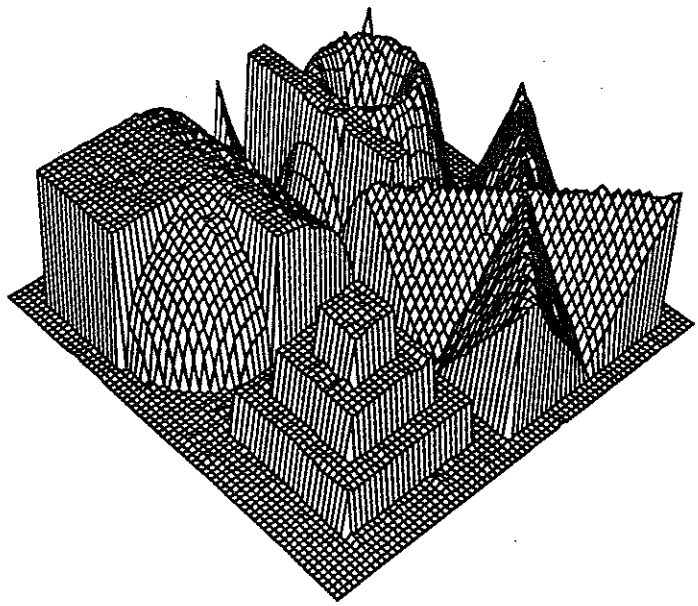

Figure 5.8: medium noise image postprocessed (5×5 window).

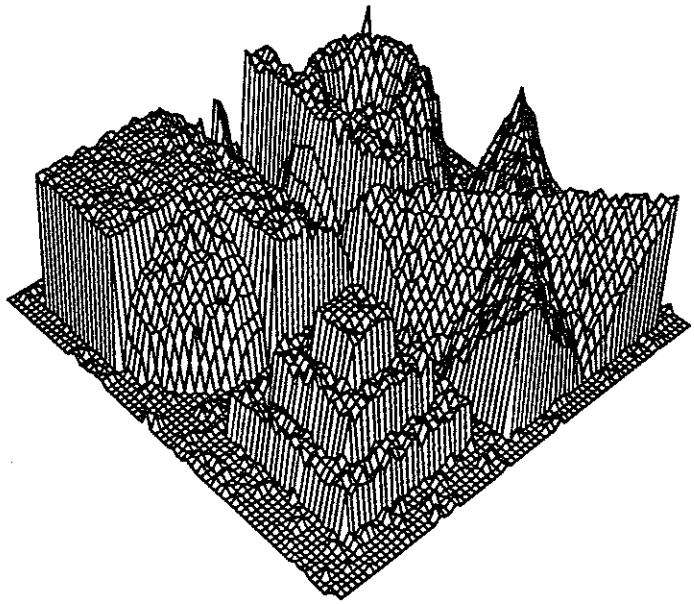

Figure 5.9: high noise image postprocessed (3×3 window).

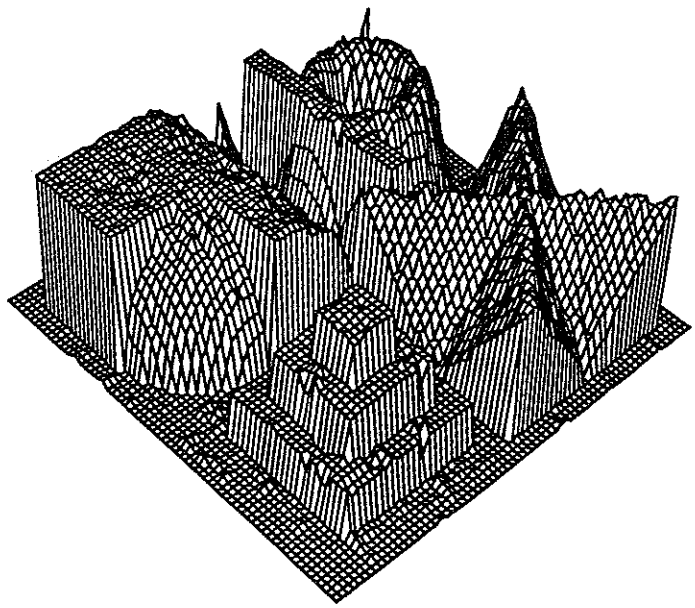

Figure 5.10: high noise image postprocessed (5×5 window).

6. Least median of squares based window operators.

6.1. Least median of squares.

In Sections 4 and 5 we have developed a surface estimation paradigm (Paradigm 5), which uses M-estimation to locally estimate different polynomial models (constant, planar and quadratic). The robust estimation of these models could also be performed by the least median of squares (LMS) method developed by Rousseeuw [36]. The objective of this section is to analyze the performance of LMS coupled with Paradigm 5, and compare it with the performance of Paradigm 5 using M-estimation. We will first describe the algorithm for LMS, and in Section 6.2 we will discuss the results of some experiments using LMS-based Paradigm 5, and then compare them to the results of Section 4.

The main difference between LMS and M-estimation (Section 2.1) is that in LMS the median of the squared errors,

$$r(\mathbf{a}) = \underset{r,c}{\text{median}}\, e_{\mathbf{a}}^2(r,c) \quad , \tag{6.1}$$

instead of the sum of a function of the errors, is minimized. The coefficient vector $\mathbf{a}$, which minimizes $\mathrm{r}(\mathbf{a})$, is the solution to the LMS problem. The error $e_{\mathbf{a}}(r,c)$ is the difference between the observed value at pixel (r,c), and the estimate computed with the polynomial model whose coefficients are the entries of $\mathbf{a}$. One of the main feature of LMS is its breakdown value

$$\epsilon^* = \frac{\text{int}(nm/2) - p + 2}{nm} \quad , \tag{6.2}$$

which converges toward 0.5 when nm increases ($\text{int}(x)$ returns the integer part of a number x). However ϵ^* is only 1/3 when fitting a planar model in a 3×3 window, and ϵ equals 0 when fitting a quadratic model in a 3×3 window.

Due to the use of a median operator in Equation 6.1, the solution vector $\mathbf{a}$ cannot be found analytically, instead the following algorithm has to be used. To begin, nm data points are extracted using a window as described in Section 2.2. We then consider all the p-tuples, which can be formed by choosing p points out of nm (p is the polynomial coefficient number). For every p-tuple $(d(r(1),c(1)), d(r(2),c(2)), \ldots, d(r(p),c(p)))$ of data points or pixels, we solve the linear system

$$d(r(i),c(i)) = \sum_{k=0}^{k+l \le d} \sum_{l=0} \alpha_{kl}\, r(i)^k\, c(i)^l \qquad i = 1,2,\ldots,p \quad , \tag{6.3}$$

where α_{kl} are the sought coefficients of the polynomial model, $r(i)$ and $c(i)$ are the coordinates of the ith p-tuple in the local coordinate system (Figure 2.2), $d = 1$ for a planar model and $d = 2$ for a quadratic model. Since there are C_p^{nm} possible p-tuples, one expects to solve C_p^{nm} linear systems. In reality, there are a few p-tuples leading to degenerate linear systems that are discarded. For example, in the case of a 3×3 window and a planar model, the three picked pixels cannot be aligned, so 8 p-tuples are dropped (3 p-tuples when the pixels are on the same row, 3 p-tuples when the pixels are on the same column and 2 p-tuples when the pixels are on the same diagonal), hence only $C_3^9 - 8 = 76$ p-tuples are used. With the same window size and a quadratic model, only 70 p-tuples are considered since more p-tuples have to be discarded (in each row, in each column, and in each diagonal a pixel must be picked). When processing with a 5×5 window, the number of p-tuples becomes too large to be handled realistically ($C_3^{25} = 2300$ and $C_6^{25} = 17700$) even if some of them are discarded; remember that this polynomial fitting is performed for every possible

window in the image. Rousseeuw [36] showed that a subsample of q p-tuples, where $q \ll C_p^{nm}$, could be picked randomly with only a small probability of error in estimating the coefficients and a slightly worse breakdown value. The number q of p-tuples can be determined by satisfying the inequality

$$1 - (1 - (1 - \epsilon)^p)^q \leq P \quad , \tag{6.4}$$

where ϵ is the "fraction of contaminated data" in the window, and P is the probability of finding the regression coefficients that are not influenced by "outliers" ($P = 1 - \xi$, with ξ a small positive number). For example, by setting $\epsilon = 0.45$ and $P = 0.95$, only 17 p-tuples are needed for estimating a planar model ($p = 3$) and 107 p-tuples with a quadratic model ($p = 6$). To make up for potential duplicate p-tuples, Rousseeuw advises to pick 400 p-tuples for a planar fit when the number of data points exceed 15, and 700 for a quadratic fit when the number of data points exceed 11 [36, p.199]. Thus q is set to these values when processing with a 5×5 window.

The fundamental assumption in LMS is that at least one p-tuple is not corrupted by outliers and will yield the robust coefficient vector α. For each solution α of Equation 6.3, we compute

$$\beta(r, c) = d(r, c) - \sum_{k=1}^{k+l \leq d} \sum_{l=1} \alpha_{kl}\, r^k\, c^l \quad \text{where } r = -\hat{n}, \ldots, \hat{n} \text{ and } c = -\hat{m}, \ldots, \hat{m}. \tag{6.5}$$

The constant coefficient α_{00} of the fitting polynomial is computed as the midpoint of the shortest half of the sample B that is composed of the $\beta(r, c)$ values. To determine the shortest half of the array B of nm values: first sort the array B to get B$'$, then the shortest half is the minimum of $\mathrm{B}'_{(nm-1)/2+l} - \mathrm{B}'_l$ with $l = 1, 2, \ldots, (nm-1)/2 + 1$, δ equals half the length of the shortest half, and $\alpha_{00} = \dfrac{\mathrm{B}'_{(nm-1)/2+l} + \mathrm{B}'_l}{2}$. Hence each p-tuple yields the fitting polynomial coefficients (α_{00}, α_{01}, ..., α_{d0}, α_{0d}). The next step is to find the best fitting polynomial for the window. Since $\delta^2 = \min r(\mathbf{a})$ for a given p-tuple, the p-tuple which yields the minimum δ^2 will minimize $r(\mathbf{a})$ over all p-tuples and hence the coefficients (α_{00}, α_{01}, ..., α_{d0}, α_{0d}) generated with this p-tuple are our solution vector $\mathbf{a}$.

This minimum δ can be used to estimate the standard deviation as

$$s = 1.4286 \left(1 + \frac{5}{(nm - p)}\right) \delta. \tag{6.6}$$

This formula is very similar to the MAD, used in Sections 4 and 5. The only difference is the $5/(nm-p)$ term, which is a finite sample size correction [36, p.202]. To be able to apply Paradigm 5, we need the weights associated with each pixel and the goodness of fit measure η_p. The weights can be computed using the bisquare ψ-function and s:

$$w(r, c) = \frac{\psi(e_{\mathbf{a}}(r, c)/s)}{e_{\mathbf{a}}(r, c)/s}. \tag{6.7}$$

Note that we could have used any ψ-function. To make valid comparisons we keep the same parameters as the ones used in Paradigm 5 (Section 4). Since the breakdown value of LMS is high, there is no need to distinguish the constant and planar model. Hence the best model is chosen as the one with the minimum η_p or the maximum γ_p in a case of a tie in the η_p values.

6.2. Experiments with LMS-based Paradigm 5.

A Monte Carlo study, following the experiment protocol described in Section 3.2, was conducted to analyze the performances of LMS as the estimation method in Paradigm 5. Table 6.1 displays the averaged MAE and RMS over 10 trials, and the corresponding standard errors for different noise configurations. The high computational cost of computing the LMS estimate limits the number of runs we could compute. We will give some figures on computation times in Section 7.

The results in Table 6.1 show that LMS based Paradigm 5 does a worse job that M-estimation based Paradigm 5 (Tables 4.1 to 4.4) by a large extent. The only case where the performance of LMS based Paradigm 5 is impressive is the noiseless image processed with a 3×3 window. For example, with a 3×3 window the M-estimation based Paradigm 5 MAEs values were about 40% less than the values obtained with LMS based Paradigm 5. Figures 6.1 to 6.8 are examples of the processed output with 3×3 and 5×5 window of noiseless, low noise, medium noise, and high noise images. They confirm the unsatisfactory performance of LMS based Paradigm 5. Note that 5×5 window processing does a better job in noise removal than 3×3 window processing but it alters edges. This last point is particularly clear in the high noise image case (Figure 6.8). This also shows that the higher breakdown value of LMS gives no advantages in surface estimation with a variable order paradigm. Hence the higher complexity of LMS is not compensated by better performance than M-estimation. In the next section (Section 7) we will postprocess LMS-based Paradigm 5 to check if this will improve its performance and make it competitive with M-estimation based Paradigm 5.

Table 6.1: MAE and RMS measures for the LMS processed images.

	MAE	RMS
noiseless		
3×3 window	0.470	5.196
5×5 window	2.598	17.677
low noise		
Original	2.440 $\pm$0.011	4.679 $\pm$0.038
3×3 window	3.511 $\pm$0.026	8.933 $\pm$0.287
5×5 window	5.619 $\pm$0.075	22.291 $\pm$0.273
medium noise		
Original	3.617 $\pm$0.013	5.787 $\pm$0.031
3×3 window	4.834 $\pm$0.043	10.069 $\pm$0.279
5×5 window	6.524 $\pm$0.053	22.611 $\pm$0.191
high noise		
Original	6.493 $\pm$0.020	9.306 $\pm$0.028
3×3 window	8.255 $\pm$0.033	3.483 $\pm$0.188
5×5 window	8.725 $\pm$0.070	23.178 $\pm$0.268

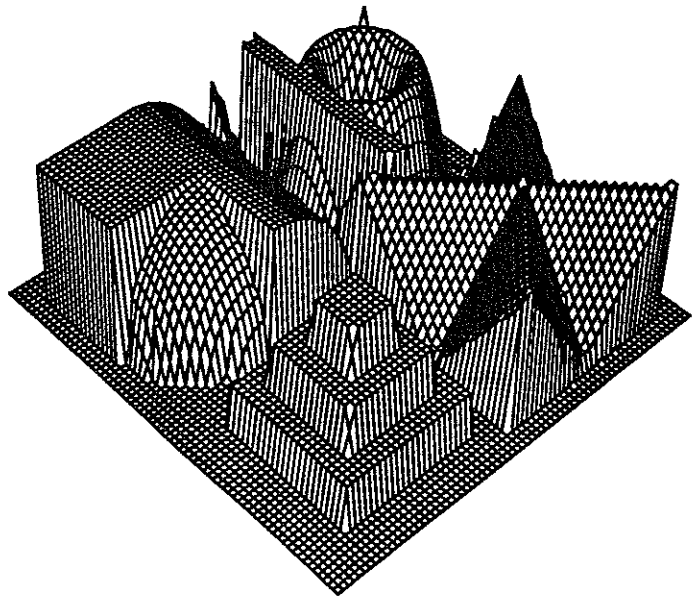

Figure 6.1: noiseless image processed (3×3 window).

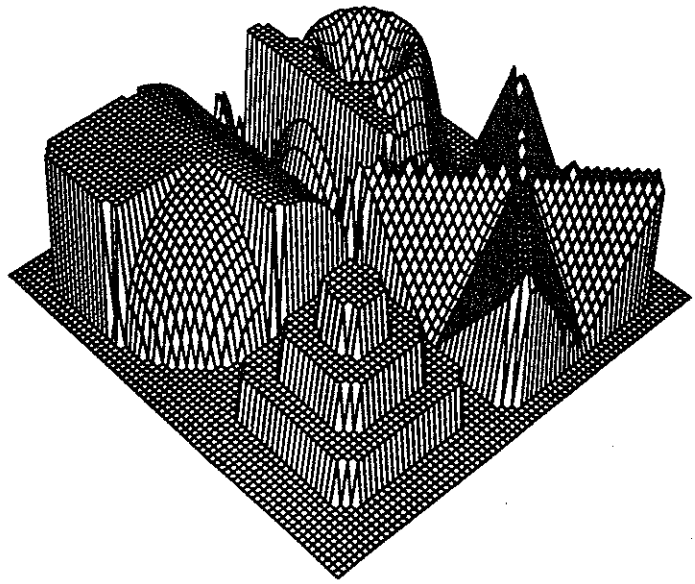

Figure 6.2: noiseless image processed (5×5 window).

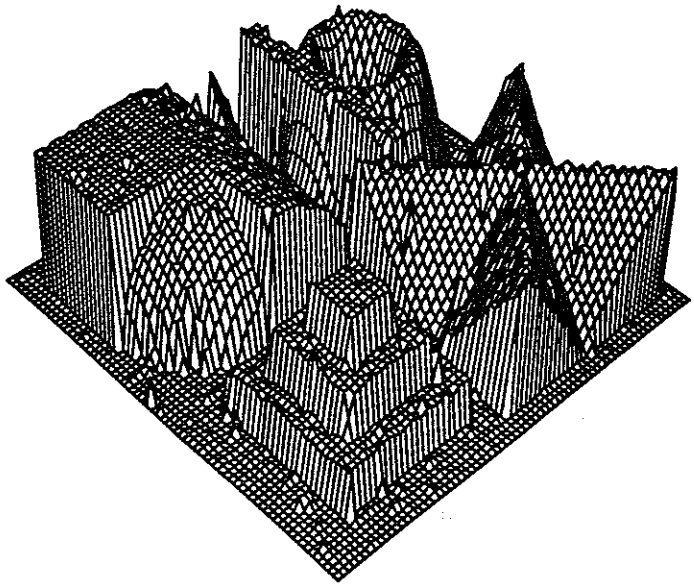

Figure 6.3: low noise image processed (3×3 window).

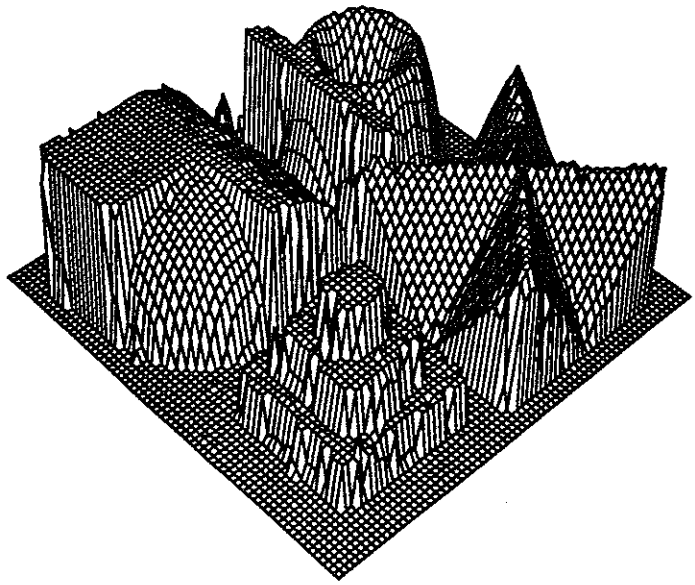

Figure 6.4: low noise image processed (5×5 window).

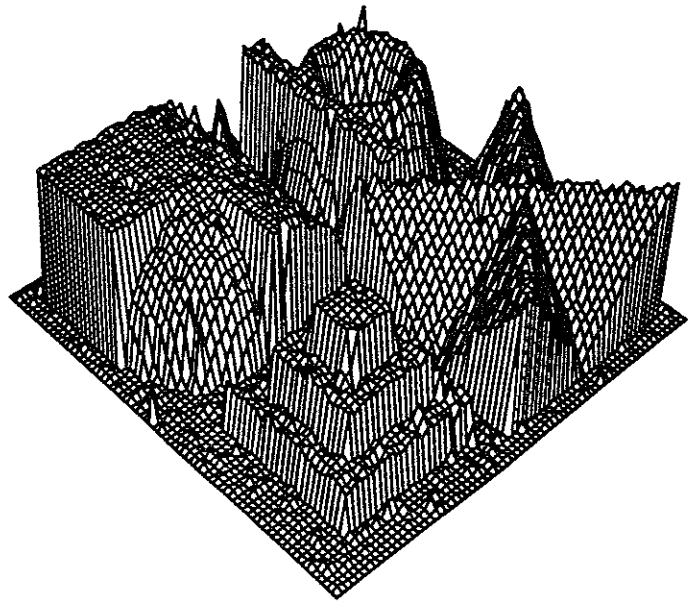

Figure 6.5: medium noise image processed (3×3 window).

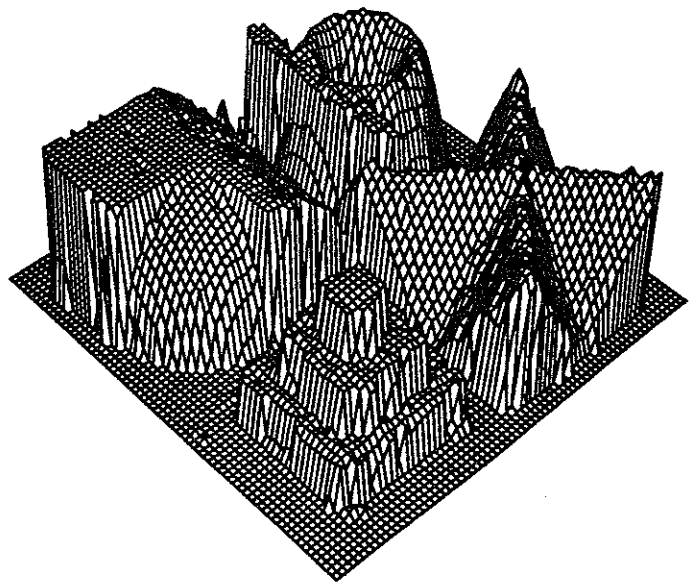

Figure 6.6: medium noise image processed (5×5 window).

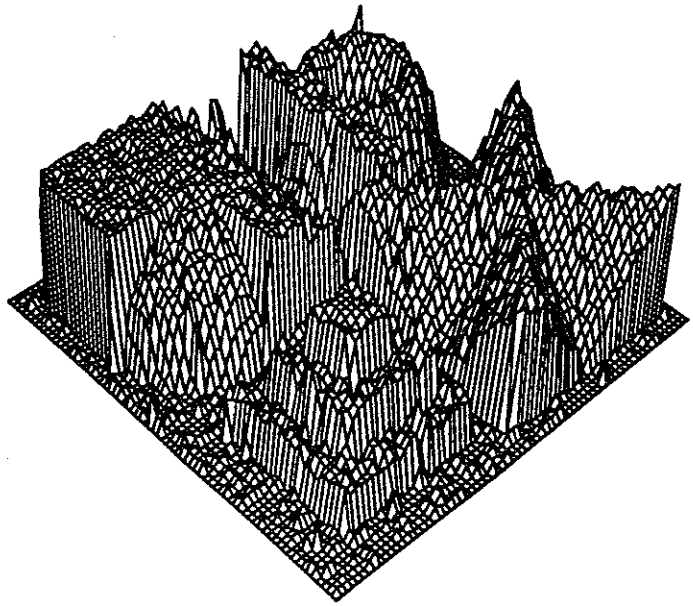

Figure 6.7: high noise image processed (3×3 window).

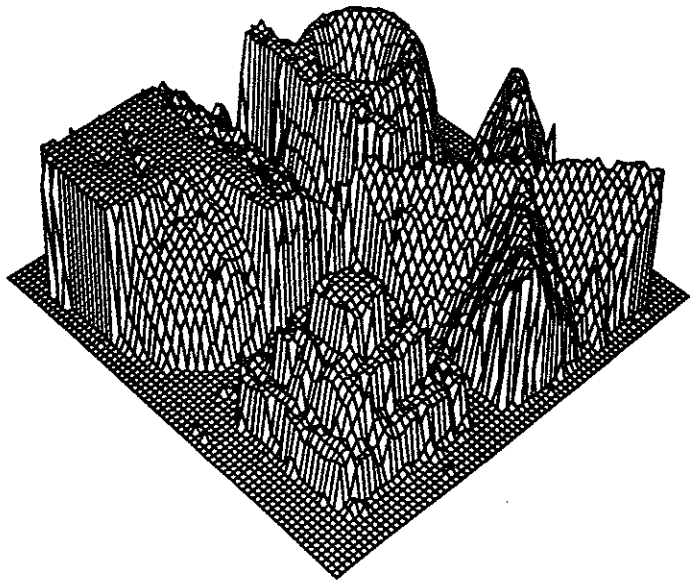

Figure 6.8: high noise image processed (5×5 window).

7. Postprocessing LMS-based Paradigm 5.

7.1. Postprocessing.

The results obtained in the previous section are somewhat disappointing, as the LMS based paradigm doesn't handle the noise along edges particularly well. This behavior of LMS-based operators was also observed by Meer, Mintz and Rosenfeld [29], who dealt with it by smoothing the data *before* the LMS processing. We will try to improve the LMS output by postprocessing it. As noted in Section 5, since the fit for every window in the image has already been computed, it is natural to exploit all these computations by estimating a pixel using all the fits from all the windows containing that pixel. The output from the LMS based Paradigm 5 is postprocessed exactly the same way as that from the M-estimation based Paradigm 5.

7.2. Experiments.

The same Monte Carlo study as the one described in Section 6.2 was performed to assess the contribution of postprocessing to a LMS based variable order facet model paradigm. Table 7.1 displays the averaged MAE and RMS over 10 runs with the associated standard errors. The results in Table 7.1 and Figures 7.1 to 7.4 show that postprocessing greatly improves the performances of the paradigm. A comparison between Table 7.1 and Table 5.5 indicates that LMS based postprocessed Paradigm 5 does better on the noiseless image than the M-estimation based postprocessed Paradigm 5, but as some noise is introduced, the M-estimation based postprocessed Paradigm 5 gains the advantage and the difference increases with the amount of noise added. This suggests that *LMS estimation is sensitive to noise and cannot handle noise as well as M-estimation*, which is consistent with the fact that the statistical efficiency of LMS estimation is lower than the statistical efficiency of M-estimation.

7.3. Computation times.

The practical utility of an image processing algorithm depends both on the quality of its output and its execution time; this is rather different from the typical application of statistics, where the quality of the results is paramount and computing time is generally irrelevant. In fairness, it must be remembered that LMS estimation was conceived in this latter context. In Table 7.2 execution times for different paradigms are given in seconds on a DECstation 5000/125, using a 72×72 medium noise image as the input image. These times illustrate two points: (1) the cost of postprocessing is relatively low, hence the benefit of postprocessing is important, and (2) the LMS-based paradigm is several orders of magnitude slower than the M-estimation based one. The random sampling of p-tuples used with 5×5 windows is insufficient to keep processing times within reasonable bounds. The data in the tables and figures of this section argue against the use of LMS as the estimation procedure in this type of application, and clearly prove the usefulness of postprocessing.

Table 7.2: execution times (seconds) for a typical 72×72 image.

		3×3 window	5×5 window
M-estimation based Paradigm 5	"plain"	41	89
	postprocessed	42	91
LMS based Paradigm 5	"plain"	469	6451
	postprocessed	470	6465

Table 7.1: MAE and RMS measures for the LMS postprocessed images.

	MAE	RMS
noiseless		
3 × 3 window	0.141	0.376
5 × 5 window	0.226	0.563
low noise		
Original	2.440 ±0.011	4.679 ±0.038
3 × 3 window	2.476 ±0.013	4.717 ±0.041
5 × 5 window	2.308 ±0.027	5.237 ±0.401
medium noise		
Original	3.617 ±0.013	5.787 ±0.031
3 × 3 window	3.676 ±0.016	5.837 ±0.033
5 × 5 window	3.548 ±0.028	6.246 ±0.396
high noise		
Original	6.493 ±0.020	9.306 ±0.028
3 × 3 window	6.606 ±0.023	9.388 ±0.027
5 × 5 window	6.505 ±0.043	9.697 ±0.151

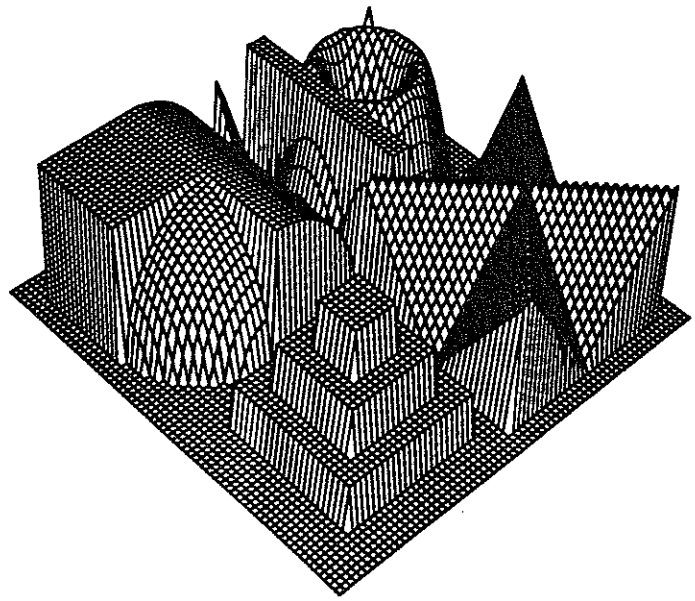

Figure 7.1: noiseless image postprocessed (3×3 window).

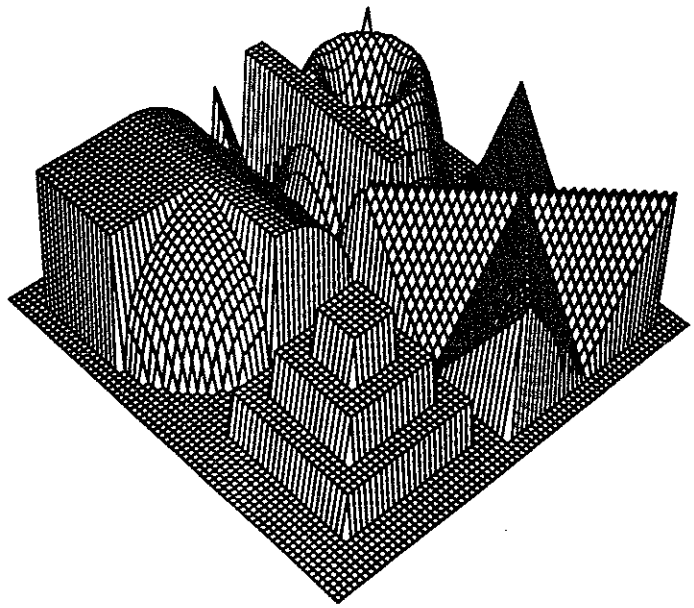

Figure 7.2: noiseless image postprocessed (5×5 window).

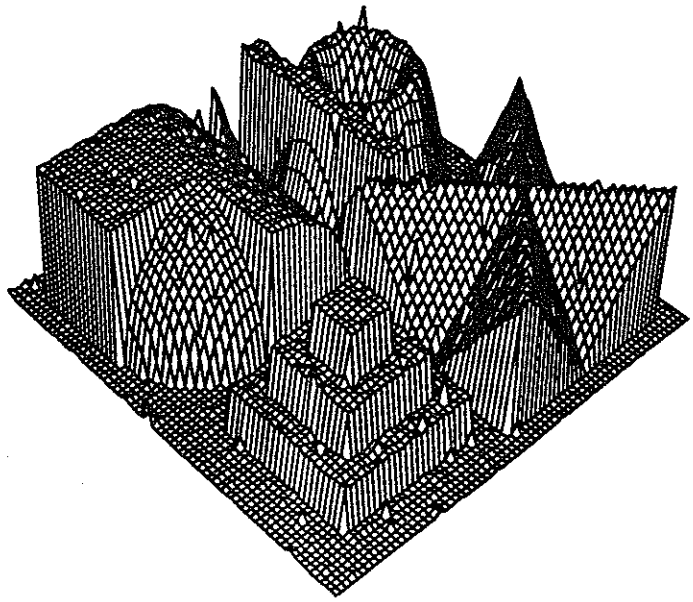

Figure 7.3: low noise image postprocessed (3×3 window).

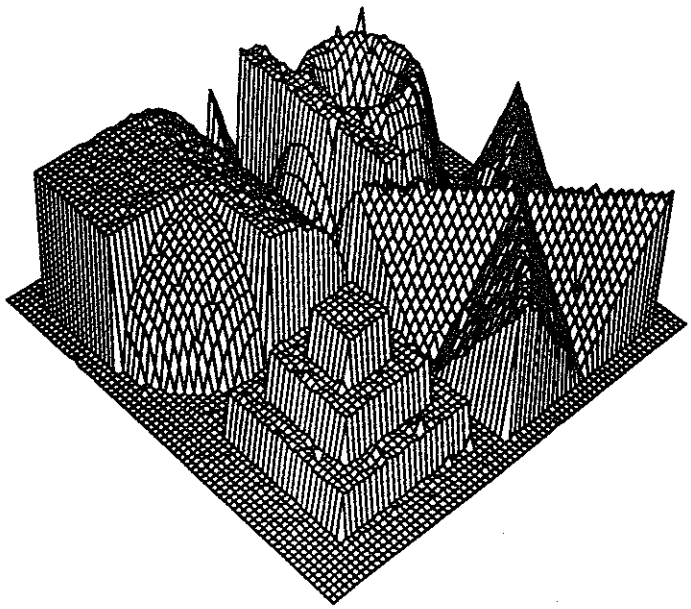

Figure 7.4: low noise image postprocessed (5×5 window).

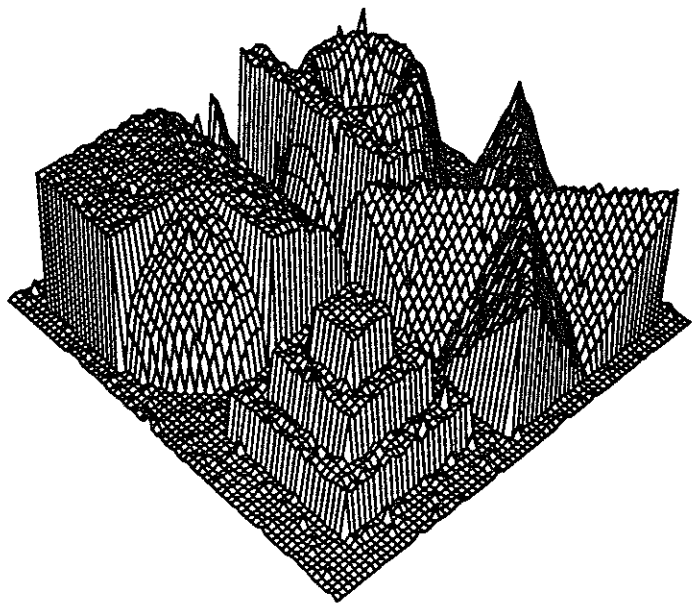

Figure 7.5: medium noise image postprocessed (3×3 window).

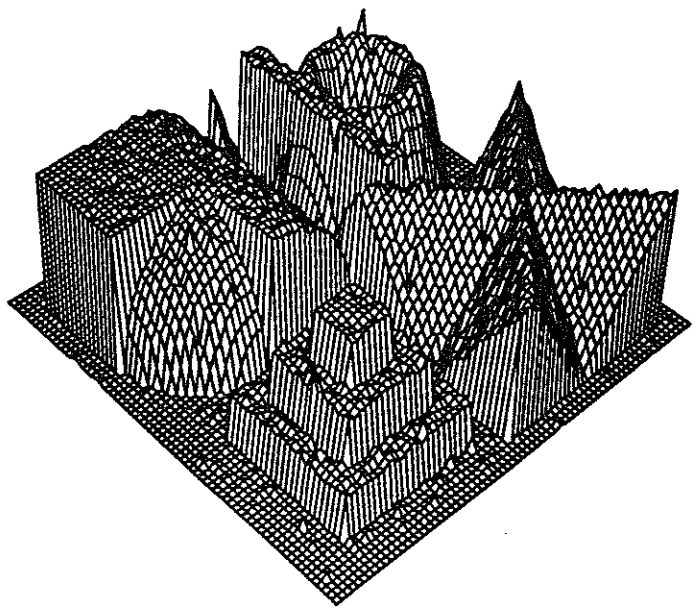

Figure 7.6: medium noise image postprocessed (5×5 window).

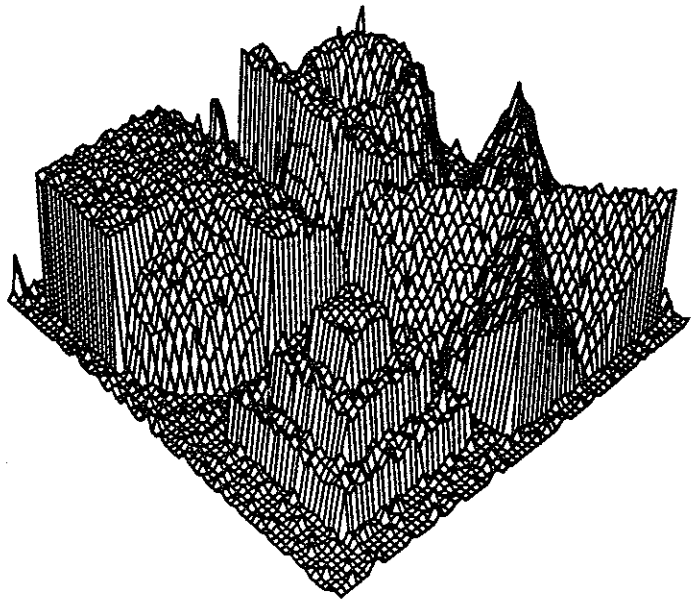

Figure 7.7: high noise image postprocessed (3×3 window).

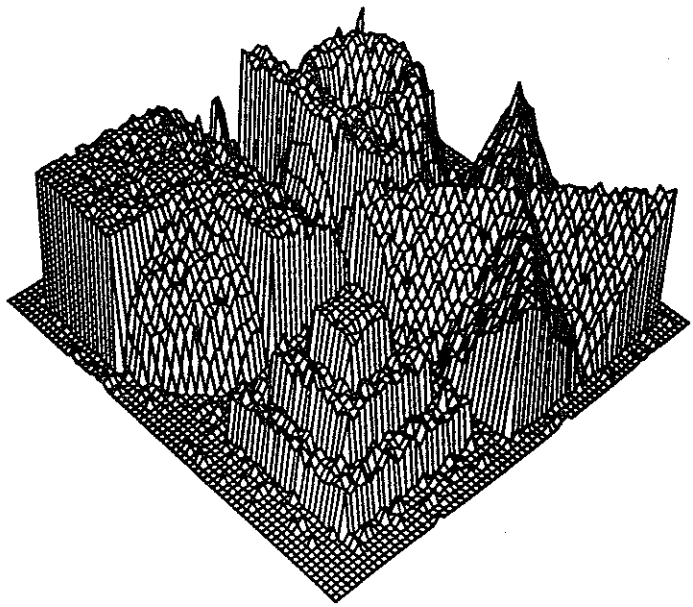

Figure 7.8: high noise image postprocessed (5×5 window).

8. Conclusion.

Both the M-estimation and LMS estimation procedures involve several parameters: for M-estimation, the choice of ψ-function, the number of IRLS iterations, the cutoff value; for LMS estimation, the number of p-tuples used, how the p-tuples are generated. The Monte Carlo studies show convincingly that, within reasonable bounds, the values of these tuning parameters are not at all crucial.

The *robust variable order facet model* paradigm proposed here has been proved to be a powerful tool to filter noise and to model the underlying surface in an image. We have shown the usefulness of a variable order model and of postprocessing in order to fully recover all the geometric features in an image, including edges and corners. Two new measures have been introduced: the consistency measure (γ_p) to resolve ties between equally good fits and a new goodness of fit measure (g_a) which takes into account the inlier/outlier information. We have also demonstrated that M-estimation offers much better performance than least median of squares to locally estimate a polynomial model. It seems that the variable order paradigm in conjunction with the postprocessing compensate for the low breakdown value of M-estimation while simultaneously exploiting the statistical efficiency of M-estimation.

An alternative paradigm would be based on using a robust F-statistic [6] to measure the significance of terms in the fitting polynomials, both during the processing and postprocessing phases. A F-test on the different fitting polynomials would complement the information obtained with the goodness of fit measure, η_p, and the consistency measure, γ_p, and could improve the choice of the model order. Another possibility would be to cascade the *robust variable order facet model* operator, i.e., make two passes, the second pass using the first pass output and presumably smoothing the region interiors and sharpening edges and corners even better.

The variable order postprocessed paradigm we have created is really intented to be a basis for further image processing like segmentation. Since at every pixel the first and second partial derivatives of the underlying surface can be evaluated through the fitting model (constant, planar or quadratic), segmentation of the surface could easily be performed by region growing as in [3]. From any given processed pixel, one can estimate the pixels in its neighborhood by extrapolation, and then check how well the estimated values compare to the actual pixel values. One could even keep track of the goodness of fit measures to weight those comparisons.

Our paradigm is still far too slow for use in real time systems; an obvious solution is to move to a parallel architecture. There are several ways to implement parallelism in our paradigm. One can split the image into several regions and then dedicate a processor to each region. Another approach would be to develop a parallel implementation of the QR decomposition (Section 2.2) or special QR hardware, since this is the most time consuming part of our paradigm.

9. Acknowledgement.

The authors wish to thank Paul Besl, Robert Haralick, and Ramesh Jain for valuable comments.

10. References.

[1] P. R. Beaudet, Rotationally invariant image operators, *Proceedings of 4th International Conference Pattern Recognition*, Kyoto, Japan, Nov. 7-10, 1978, 579–583.

[2] P. J. Besl and R. C. Jain, Invariant surface characteristics for three-dimensional object recognition in range images, *Computer Vision, Graphics, Image Processing*, **33**, 1 (January), 1986, 33–80.

[3] P. J. Besl and R. C. Jain, Segmentation via variable-order surface fitting, *IEEE Trans. Pattern Analysis Machine Intelligence*, **PAMI-10**, 2 (March), 1988, 167–192.

[4] P. J. Besl, J. B. Birch and L.T. Watson, Robust window operators, *Proceedings of the Second International Conference on Computer Vision*, December 5–8, Tampa, Florida, 1988, 591-600. See also Machine Vision and Applications **2**, 1989, 179–214.

[5] J. B. Birch, Some convergence properties of iteratively reweighted least squares in the location model, *Commun Stat.*, **B9**, 1980, 359–369.

[6] J. B. Birch and D. B. Agard, Robust Inference in Multiple Regression, *Proceedings of the Statistical Computing Section, American Statistical Association*, (to appear in 1992), 6 pages.

[7] R. M. Bolle and D. B. Cooper, Bayesian recognition of local 3-D shape by approximating image intensity functions with quadric polynomials, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6, 4 (July), 1984, 418–429.

[8] R. C. Bolles and M. A. Fischler, A RANSAC-based approach to model fitting and its application to finding cylinders in range data, *Proceedings of 7th International Joint Conference on Artificial Intelligence*, (Vancouver, B.C., Canada, Aug. 24–28), 1981, 637–643.

[9] D. S. Chen, A data-driven intermediate level feature extraction algorithm, *IEEE Trans. Pattern Analysis Machine Intelligence*, **PAMI-11**, 7 (July), 1988, 749–758

[10] L. S. Davis and A. Rosenfeld, Noise cleaning by iterated local averaging, *IEEE Trans. Systems, Man, Cybernetics*, **SMC-8**, 9 (September), 1978, 705–710.

[11] R. O. Duda and P. E. Hart, The use of Hough transform to detect lines and curves in pictures, *Comm. ACM*, **15**, 1972, 11–15.

[12] W. Forstner, Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision, *Computer Vision, Graphics, Image Processing*, **40**, 1987, 273–310.

[13] N. C. Gallagher and G. L. Wise, A theoretical analysis of the properties of median filters, *IEEE Trans. Acous., Speech, Signal Processing*, **ASSP-29**, 6, (December), 1981, 1136–1141.

[14] R. R. Hansen and R. Chellappa, Two-dimensional Robust Spectrum Estimation, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **36**, 7, (July), 1988, 1051–1066.

[15] R. M. Haralick and H. Joo, 2D-3D pose estimation, *Proceedings of the 9th International Conference on Pattern Recognition*, (November 14–17, Rome, Italy), 1988, 385–391.

[16] R. M. Haralick, H. Joo, C. N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, Pose estimation from corresponding point data, *IEEE Trans. on Systems, Man, and Cybernetics*, **19**, No. 6 (November-December), 1989, 1426–1446.

[17] R. M. Haralick and L. T. Watson, A facet model for image data, *Computer Graphics Image Proc.*, **15**, 1981, 113–129.

[18] R. M. Haralick, L. T. Watson, and T. J. Laffey, The topographic primal sketch, *Int. J. Robotics Res.*, **2**,1 (Spring), 1983, 50–72.

[19] D. Harwood, M. Subbarao, H. Hakalahti, L. and Davis, A new class of edge-preserving smoothing filters, *Pattern Recognition Letters*, **6** (August), 1987, 155–162.

[20] R. Hoffman and A. K. Jain, Segmentation and classification of range images, *IEEE Trans. Pattern Anal. Machine Intell.*, **PAMI-9**, 5 (September), 1987, 608–620.
[21] T. S. Huang, G. J. Yang, and G. Y. Tang, A fast two-dimensional median filtering algorithm, *IEEE Trans. Acoust., Speech, Signal Processing*, **ASSP-27**, 1979, 13–18.
[22] P. J. Huber, *Robust Statistics*, Wiley, NY., 1981.
[23] M. Hueckel, A local operator which recognizes edges and lines, *J. Assoc. Comp. Mach.*, **20**, 1973, 634–647.
[24] S. L. Hurt and A. Rosenfeld, Noise reduction in three-dimensional digital images, *Pattern Recognition*, **17**, 4, 1984, 407–421.
[25] J.-M. Jolion, P. Meer, and S. Bataouche, Robust clustering with applications in computer vision, *IEEE Trans. Pattern Anal. Machine Intell.*, **PAMI-13**, 8 (August), 1991, 791–801.
[26] R. L. Kashyap and K. Eom, Robust image modeling techniques with an image restoration application, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **36**, 8, (August), 1988, 1313–1325.
[27] R. L. Kashyap and K. Eom, Robust image models and their applications, in *Advances in Electronics and Electron Physics*, **70**, Academic Press, San Diego, CA, 1988, 80–157.
[28] G. Medioni, Smoothing by diffusion, DARPA Image Understanding Workshop. Also USC Technical Report, 1987.
[29] P. D. Meer, D. Mintz and A. Rosenfeld, Least Median of Squares Based Robust Analysis of Image Structure, CS-TR-2428, Center for Automation Research, University of Maryland, College Park, MD, 1990.
[30] P. Meer, D. Mintz, D. Y. Kim, and A. Rosenfeld, Robust Regression Methods for Computer Vision: A review, *Int. J. Computer Vision*, **6**, 1991, 59–70.
[31] D. Mintz, P. Meer, and A. Rosenfeld, Consensus by Decomposition: A Paradigm for Fast High Breakdown Point Robust Estimation, Technical Report CAR-TR-525, Center for Automation Research, University of Maryland, College Park, (December), 1990.
[32] D. Mintz, Robust Consensus Based Edge Detection, Technical Report CAR-TR-546, Center for Automation Research, University of Maryland, College Park, (March), 1991.
[33] J. Prewitt, Object enhancement and extraction, *Picture Processing and Psychopictorics*, B. Lipkin and A. Rosenfeld, Eds., Academic Press, New York, 1970, 75–149.
[34] L. G. Roberts, Machine perception of three-dimensional solids, *Optical and Electro-Optical Information Processing.*, J.T. Tippett et al., Eds., MIT Press, Cambridge, Mass., 1965, 159–197.
[35] G. Roth and M. D. Levine, Segmentation of geometric signals using robust fitting, *IEEE 10th International Conference on Pattern Recognition*, (16–21 June), 1990, 826–831.
[36] P. J. Rousseeuw and A. Leroy, *Robust Regression and Outlier Detection*, John Wiley and Sons, New York, 1987.
[37] G. Sharma and R. Chellappa, An iterative algorithm for 2-D robust spectral estimation, *Proc. Int. Conf. ASSP*, San Diego, CA, 1984.
[38] S. S. Sinha and B. G. Schunck, A robust method for surface reconstruction, in *Proceedings of the International Workshop on Robust Computer Vision*, (Seattle, WA, USA, 1–3 October), 1990, 183–199.
[39] A. Tirumalai and B. G. Schunck, Robust Surface Approximation Using Least Median Squares Regression, University of Michigan, 1989, CSE-TR-13-89.
[40] A. Waks and O. J. Tretiak, Robust detection of region boundaries in a sequence of images, *IEEE 10th International Conference on Pattern Recognition*, (16–21 June), 1990, 947–952.

[41] Y. T. Zhou, V. Venkateswar, and R. Chellappa, Edge detection and linear feature extraction using a 2-D random field model, *IEEE Trans. Pattern Anal. Machine Intel.*, **PAMI-11**, 1, 1989, 84–95.

[42] X. Zhuang and R. M. Haralick, Developing robust techniques for computer vision, in *Proceedings of the International Workshop on Robust Computer Vision*, (Seattle, WA, USA, 1–3 October), 1990, 19–38.

[43] X. Zhuang and R. M. Haralick, A highly robust estimator for computer vision, *IEEE 10th International Conference on Pattern Recognition*, (16–21 June), 1990, 545–550.