Surface–Normal Estimation with Neighborhood Reorganization for 3D Reconstruction

Felix Calderon[1], Ubaldo Ruiz[1], and Mariano Rivera[2]

[1] Universidad Michoacana de San Nicolás de Hidalgo
División de Estudios de Posgrado. Facultad de Ingeniería Eléctrica
Santiago Tapia 403 Centro. Morelia, Michoacán, México. CP 58000
calderon@umich.mx ubaldo@fismat.umich.mx
[2] Centro de Investigacion en Matematicas A.C.
Apdo. Postal 402, Guanajuato, Gto. Mexico. CP 36000
mrivera@cimat.mx

Abstract. Fastest three-dimensional (3D) surface reconstruction algorithms, from point clouds, require of the knowledge of the surface–normals. The accuracy, of state of the art methods, depends on the precision of estimated surface–normals. Surface–normals are estimated by assuming that the surface can be locally modelled by a plane as was proposed by Hoppe *et. al* [1]. Thus, current methods for estimating surface–normals are prone to introduce artifacts at the geometric edges or corners of the objects. In this paper an algorithm for Normal Estimation with Neighborhood Reorganization (NENR) is presented. Our proposal changes the characteristics of the neighborhood in places with corners or edges by assuming a locally plane piecewise surface. The results obtained by NENR improve the quality of the normal with respect to the state of the art algorithms. The new neighborhood computed by NENR, use only those points that belong to the same plane and they are the nearest neighbors. Experiments in synthetic and real data shown an improvement on the geometric edges of 3D reconstructed surfaces when our algorithm is used.

Keywords: Normal Estimation, Point Cloud, Surface Reconstruction.

1 Introduction

The computational representations of physical objects have large and wide applications in distinct areas like industrial design, computer simulations and medicine, among others: an object digitalization can be easily studied, modified or replicated. In a initial stage, a complex real object is detailed scanned, by using of a proper device, for acquiring of a point cloud with thousand or millions of points. In a second stage, a reconstruction algorithm is applied on the point cloud for producing a, generally triangular, mesh that approximates the object surface. Thus the resulting mesh is a suitable representation of the real object.

The reconstruction algorithms must be able to approximate, at a reasonable computational time, the geometric features of the real object. Among the reconstruction algorithms reported in the literature, Multilevel Partition of Unity

© Springer-Verlag Berlin Heidelberg 2007

Implicits (MPUI) [2] deserves our special attention. Currently, the MPUI is considered as one of the fastest and most up-to-date algorithms for surface reconstruction, however, it is important to notice that the MPUI like the rest of the volumetric methods [1,3,4] requires an estimation of the normal at each point. Clearly, good normal estimations are necessary for a good surface reconstruction.

Some complex three–dimensional (3D) scanner devices estimate the surface normals at the acquisition time. However, in order to eliminate any dependency to those devices it is better to infer such normals from the point set. For instance, Refs. [1,5] reported algorithms for addressing the surface normal problem. But those methods fail, to estimate correctly normals, at sites close to edges or corners. In this paper we propose an edge preserving normal regularization technique based in an adaptive rest condition spring system proposed by Rivera and Marroquin in [6] that allows us to improve the normal-surface estimation at points close to edges and, consequently, improving the quality of the final surface reconstruction.

The paper is organized as follows. In Sect. 2 we present a brief review of the reconstruction algorithms that have been developed. In Sect. 3 we describe a pioneering technique [1], that still, is widely used for normal estimation in point clouds. In Sect. 4 we propose an algorithm for normal estimation that increase the accuracy in regions with edges. In Sect. 5 we compare the performance of the MPUI algorithm [2] when the surface–normal are estimated with the method presented in [1] and with the method we propose here. Finally, some conclusions are discussed in Sect. 6.

2 Surface Reconstruction

Currently, one can distinguish two main lines of research in the field of surface reconstruction from point clouds. The algorithms developed in the context of Computational Geometry (CG) constitute a line of research [7]. In CG-based methods the surface is reconstructed from the Delaunay tetrahedrization of the point cloud [8,9,10]. Unfortunately, CG-based methods are very sensitive to noise and computationally expensive; their computational complexity is of polynomial order with respect the number of points. On the other hand, volumetric methods constitute the alternative research line [1,2,3,4]. In this case, implicit functions are fitted to the point cloud and the surface is extracted (see [11,12,13]) as the zero level set of the computed functions. These methods are used extensive because their better noise tolerance and because they are able to handle large point clouds. However, it is necessary to supply reliable information about the normal at each point. The current state of the art in this approach includes the MPUI algorithm based on quadratic functions [2]. It is one of the few reconstruction algorithms able to process sharp features and also one of the fastest techniques available [2].

The MPUI algorithm works as follows. First it creates an octree-based subdivision of a box that bounds the point cloud. At each cell of the octree the local shape is approximated with a piecewise quadratic function. These functions work like a signed distance function taking a positive value inside of the point cloud and negative outside of it. The normals of the points are used to distinguish the orientation locally. If the local approximation into a cell is not accurate, then the cell is subdivided; such a procedure is repeated until a desirable accuracy level is reached. The global implicit function describing the surface is given by assembling the local approximations using local weights.

The MPUI algorithm has three types of quadratic functions that allows to model a large variety of point set configurations. Additionally, MPUI provides test–rule for choosing the appropriated quadratic form to fit at each cell. These test–rule give to the algorithm the ability to deal with surface edges. The MPUI algorithm input are the point cloud and the respective surface–normals. In the case were only the point cloud is given (surface normals are not provided), Ohtake *et al.* [2] suggest to compute such surface–normals with the technique proposed by Hoppe *et al.* in [1]. The surface-normal estimation method is described in next section.

3 Standard Surface–Normal Estimation

The estimation of surface–normals from a point cloud is usually done in two stages, as proposed Hoppe *et al.* in [1] (NE–Hoppe). In the first stage the Tangent Plane (TP) is estimated at each point. Thus the Orthogonal unitary vector to the Tangent Plane (OTP) will be used as an approximation of the normal at such point. In the second stage the orientation of the OTP, spatially coherent, is computed.

Given a point set $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$ and let be V_i the set of k nearest neighbors (neighborhood) of the point $\boldsymbol{p}_i$. Then the TP at $\boldsymbol{p}_i$ is obtained by fitting a plane to the points in V_i by using a least-squares procedure, then the surface–normal, $\boldsymbol{n}_i$, is the normal to the TP. Hoppe *et al.* [1] proposed to compute, $\boldsymbol{n}_i$, as the third eigenvector (associated with the smallest eigenvalue) of the local covariance matrix:

$$C_i = \sum_{j \in V_i} (\boldsymbol{p}_j - \boldsymbol{c}_i) \otimes (\boldsymbol{p}_j - \boldsymbol{c}_i) \tag{1}$$

where $\otimes$ denotes the outer product vector operator[1]. If $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$ are the eigenvalues of C_i, their associated eigenvectors $\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, \boldsymbol{v}_i^3$, respectively, form an orthonormal basis. Then $\boldsymbol{n}_i$ is either $\boldsymbol{v}_i^3$ or $-\boldsymbol{v}_i^3$. The neighborhood size is chosen manually based on visual inspection of the resulting normals and it is the same for each point in the set.

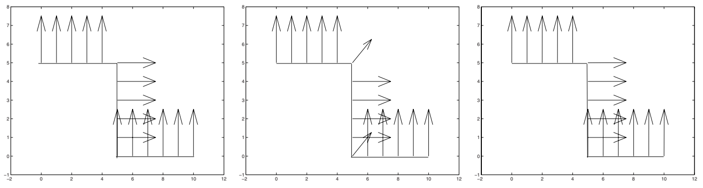
The Oriented OTP (OOTP) is computed such that nearby planes are consistently oriented. The NE–Hoppe algorithm, proposed in [1], considered the state of the art, is following described. First, an Euclidean Minimum Spanning Tree is created over the TP centers $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n\}$ and it is enriched adding the edge $< i, j >$ if either $\boldsymbol{c}_i \in V_j$, or $\boldsymbol{c}_j \in V_i$. Then the edge cost is equaled to $1 - |\boldsymbol{n}_i \cdot \boldsymbol{n}_j|$.

[1] If $\boldsymbol{v}$ and $\boldsymbol{u}$ have components v_i and u_j respectively, then the matrix $\boldsymbol{v} \otimes \boldsymbol{u}$ has $u_i v_j$ as its ij-th entry.

Next, the Minimum Spanning Tree of this graph is computed. The OTP whose TP center has the largest z coordinate is forced to point towards $+z$ axis. Rooting the tree at this node, then the tree is traversed in depth first order, if during traversal the current node i has been assigned the orientation $\boldsymbol{n}_i$ and the node j is the next node to be visited, then $\boldsymbol{n}_j$ is replaced with $-\boldsymbol{n}_j$ if $\boldsymbol{n}_i \cdot \boldsymbol{n}_j < 0$.

Pauly *et al.* [5] noticed that nearby points in the neighborhood of a point $\boldsymbol{p}_i$ should have a stronger influence than distant points. Therefore, they assign different weights to elements in the neighborhood by depending on their distance to $\boldsymbol{p}_i$. The weighting function is proposed to be the Gaussian: $w(\boldsymbol{p}_j - \boldsymbol{p}_i) = \exp(-\|\boldsymbol{p}_j - \boldsymbol{p}_i\|^2/(2\sigma^2))$, where σ is chosen as one third of the square distance between $\boldsymbol{p}_i$ and its farthest neighbor: $\sigma^2 = (1/6)\max_{\boldsymbol{p}_j \in V_i} \|\boldsymbol{p}_j - \boldsymbol{p}_i\|^2$.

Although the previous algorithms [1,5] work well in the presence of smooth regions and moderate noise, they perform poorly in those regions near corners or edges. If the neighborhood at each of the points has a fixed size and it is constructed using only the Euclidean distance then it is possible that points considered as outliers for a certain region be used in the computation of the normal. Hence, it is important to develop a new robust strategy that estimate the local surface–normal by discarding neighbor points that lay beyond of a surface edge or a corner. Figure 1 shows the normal estimation by different approaches for a step function in two dimensions. Figure 1(a) shows the ground truth and Fig. 1(b) shows the computed normals using neighborhoods based only in a proximity measure, note the effect in corners of the step function. On the other hand, Fig. 1(c) shows the results applying our approach (that will be introduced in next section).



(a) (b) (c)

Fig. 1. Normal estimation comparison in two dimensions for a step function. (a) Ground Truth, (b) NE–Hoppe algorithm and (c) NENR algorithm.

4 Normal Estimation with Neighborhood Reorganization (NENR)

The normal estimation using the OOTP is equivalent to apply a low-pass band filter to the point cloud, so the resulting normal, will have a smoothness degree which is proportional to the neighborhood size. If OOTP is used, one is implicitly assumed that the neighborhood around each point may be modelled by simple plane. Such assumption is incorrect at points close to edges or corners. An alternative approach could be to weight each point in the neighborhood; however the underlying representation of each neighborhood still it is a unique plane. A more appropriated model is to assume a plane–piecewise model. Half-quadratic regularization (HQR) is an edge-preserving regularization technique for restoring piecewise smooth signals [14,15,16,17]; the general HQR energy function is given by

$$U(h,l) = \sum_{i \in \Omega} \left[(h_i - g_i)^2 + \alpha \|\nabla h_i\|^2 (1 - l_i)^2 + \alpha \beta l_i^2 \right]; \tag{2}$$

where g is a given signal, h is the filtering signal with edge-preserving and α, β are parameters which control the signal smoothness. l_i acts as an indicator variable which disconnects the ∇h_i terms with a huge contribution to the general cost function. This technique is also applied by Calderon in [18,19], for Image Registration.

In case of a piecewise constant surface, for a given neighborhood, we approximate this surface by TPs and it is desirable to get out points of the cloud belonging to different TPs. So a new neighborhood is computed considering only the nearest neighbors who belong to the same TP. Figure 2(a) shows the standard plane estimation, for the case of three points on a corner, and Fig. 2(b) shows the plane estimation when a point is rejected, a desirable condition in those cases. We propose a HQR cost function for this purpose and the indicator variable in our case, is used as non-membership term, $l_{ij} = 1$ means that the $j - th$ point, in the original neighborhood, it is not in the same TP that $i - th$ point. The OOTP smoothness is controlled with the parameters α and β. We proposed to compute the surface-normal as the minimization of a constrained HRQ cost function. The additional constraint imposes a unitary norm to the flat piecewise normal vector $\boldsymbol{m}_i$:

$$\begin{aligned} U(\boldsymbol{m}, l) = \sum_{i \in \Omega} \left\{ \|\boldsymbol{m}_i - \boldsymbol{n}_i\|^2 + \alpha \sum_{j \in V_i} \left[\|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2 (1 - l_{ij})^2 + \beta l_{ij}^2 \right] \right\}, \\ \textbf{s.t.} \qquad \|\boldsymbol{m}_i\| = 1, \qquad \forall i \in \Omega; \end{aligned} \tag{3}$$

where $\boldsymbol{n}_i$ is the normal vector computed with the NE–Hoppe algorithm described in Sect. 3 for some neighborhood size. Then by using the Lagrange multipliers, γ, we include the constraint in the Lagrangian:

$$L(\boldsymbol{m}, l, \gamma) = U(\boldsymbol{m}, l) - \sum_{i \in \Omega} \gamma_i \left(\sum_{d=1}^{3} m_{i,d}^2 - 1 \right) \tag{4}$$

Then the solution is computed solving the Karush-Khun-Tucker conditions:

$$\nabla_{\boldsymbol{m}} L(\boldsymbol{m}, l, \gamma) = 0, \tag{5}$$

$$\nabla_l L(\boldsymbol{m}, l, \gamma) = 0, \tag{6}$$

$$\|\boldsymbol{m}_i\| = 1; \tag{7}$$

where ∇_x denotes the partial gradient operator. We propose to use the Gauss-Seidel algorithm, for solving this system of equation, due the fact that the system of equation has a banded, diagonally dominant and semi-positive defined matrix, so the t-th Gauss-Seidel Iteration is given by the equations (8, 9 and 10):

$$\widehat{m}_{i,d}^{(t)} = \frac{m_{i,d}^{(t)}}{\sqrt{\sum_k \left(m_{i,d}^{(t)}\right)^2}} \qquad \forall i \in \Omega, \tag{8}$$

$$l_{ij}^{(t)} = \frac{\left\|\widehat{\boldsymbol{m}}_i^{(t)} - \widehat{\boldsymbol{m}}_j^{(t)}\right\|^2}{\beta + \left\|\widehat{\boldsymbol{m}}_i^{(t)} - \widehat{\boldsymbol{m}}_j^{(t)}\right\|^2} \qquad \forall <i,j> \in \Omega^2, \tag{9}$$

$$m_{i,d}^{(t+1)} = \frac{n_{i,d} + \alpha \sum_{j \in V_i} \widehat{m}_{j,d}^{(t)} \left(1 - l_{ij}^{(t)}\right)^2}{1 + \gamma_i + \alpha \sum_{j \in V_i} \left(1 - l_{ij}^{(t)}\right)^2}. \tag{10}$$

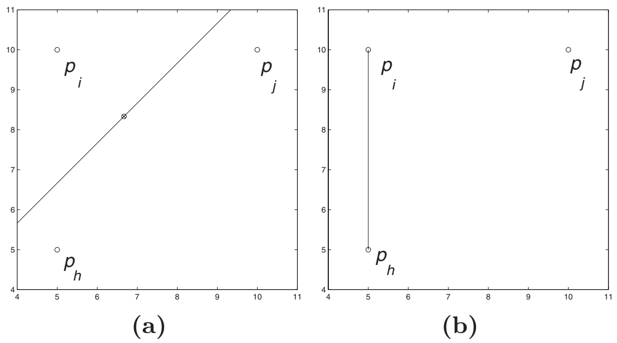


Fig. 2. Different plane estimations using three points on a corner. (a) Plane over the three points and (b) plane over points p_i and p_h when p_j is not included.

The restriction for the equation (3) is fulfill at each iteration according with equation (8), so the Lagrange multiplier γ can take any value, because their contribution to the constrained HRQ cost function will be zero at each iteration, for simplicity we put $\gamma_i = 0$ in equation (10).
The NENR algorithm is resumed in the next steps:

1. Compute the OOTP $\boldsymbol{n}_i$ for some neighborhood with size k
2. Compute the reorganized neighborhood and the filter normal $\boldsymbol{m}_i$ doing
   - (a) Set $\boldsymbol{m}_i^{(0)} = \boldsymbol{n}_i$ and $t = 0$
     (b) Normalize the vectors $\boldsymbol{m}_i^{(t)}$ applying (8)
     (c) Compute the memberships $l_{ij}^{(t)}$ using (9)
     (d) Update the normal vectors $\boldsymbol{m}_i^{(t+1)}$ applying (10)
     (e) Set $t \leftarrow t + 1$
     (f) Repeat steps (b-e) until $\left\|\boldsymbol{m}_i^{(t+1)} - \boldsymbol{m}_i^{(t)}\right\| < \varepsilon$

3. Finally compute the robust OTP with the weighted covariance matrix:

$$C_i = \sum_{j \in V_i} \left(1 - l_{ij}^*\right) \left[(\boldsymbol{p}_j - \boldsymbol{c}_i^*) \otimes (\boldsymbol{p}_j - \boldsymbol{c}_i^*) \right] \tag{11}$$

with

$$\boldsymbol{c}_i^* = \frac{\sum_{j \in V_i} \left(1 - l_{ij}^*\right) \boldsymbol{p}_j}{\sum_{j \in V_i} \left(1 - l_{ij}^*\right)} \tag{12}$$

using the finals (optimum) memberships l_{ij}^*, then compute the OOTP as was described in Sect. 3.

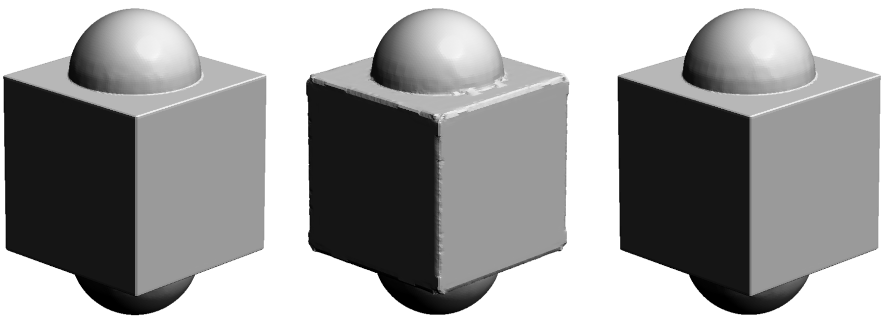
5 Experimental Results

We perform experiments in both synthetic and real data, for comparing NE–Hoppe and NENR algorithms. The synthetic data corresponds to a step function and a 3D model with a ground truth while the real data corresponds to 3D models widely used in the literature. All the 3D models were reconstructed using the Ohtake's MPUI implementation available at [20].

The results for the step function in two dimensions are presented in Fig. 1. The normal vectors in Fig. 1(a) were assigned manually according with the step function and the normal field estimated by NE–Hoppe and NENR are shown in Figs. 1(b) and 1(c), respectively. The neighborhood sizes for both algorithms were $k = 2$ and the NENR parameters were $\alpha = 50000$ and $\beta = 0.001$. Note the similarity between the NENR (Fig. 1(c)) and the original normals (Fig. 1(a)) also note the problem presented by NE–Hoppe algorithm in corners of the step function.

For the synthetic 3D model, the normals have been assigned manually according to the characteristics of the surface object and its surface reconstruction is shown in Fig. 3(a). The resulting surface is used as a reference for a qualitative comparison. The MPUI parameters in this case were a grid size of 0.005, and a max error of 0.005 at each cell. For the rest of the parameters we took the default configuration of Ohtake's MPUI implementation. The neighborhood size was taken equal to 15 for both normal estimation algorithms. Additionally, for the NERN algorithm, we took $\alpha = 1000$ and $\beta = 0.01$. The surface reconstructed using the NE–Hoppe is presented in Fig. 3(b). Note that the edges of the reconstructed surface are over–smoothed as a direct consequence of bad normal estimation near these regions. Finally, Fig. 3(c) shows the surface reconstructed using the NENR algorithm. We must notice that NENR algorithm increases the quality of the reconstruction, shown sharped geometric edges without affecting the smooth areas, as you can see comparing Figs. 3(b) and 3(c).

For a quantitative comparison between NERN and NE-Hoppe algorithms, we compute the angle between the ground truth surfaces normals and the estimated normals using both algorithms. The mean angle and standard deviation are shown in Table 1 for the step function (Fig. 1) and the synthetic 3D model (Fig. 3). Note the better results for the normal estimation using NERN than NE-Hoppe in both cases.

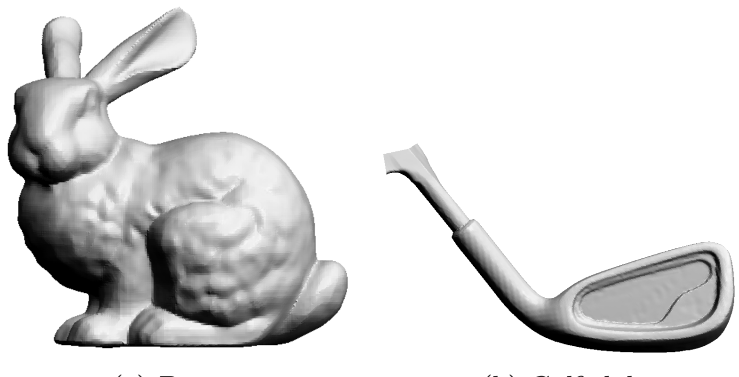


(a) ***Ground Truth*** **(b)** ***NE–Hoppe*** **(c)** ***NENR***

Fig. 3. Surface reconstruction using MPUI and different Surface–Normal Estimations

Table 1. Normal estimation angle between ground truth normals and the estimated normals of both methods

Algorithm	Step Function		Synthetic Model	
	Mean Angle	Std. Deviation	Mean Angle	Std. Deviation
NE-Hoppe	5.7169°	15.2029°	7.5544°	13.5544°
NERN	0.1764°	0.1852°	0.9973°	2.0968°



(a) ***Bunny*** **(b)** ***Golf club***

Fig. 4. Surface reconstruction using MPUI and NENR for real models

Figures 4(a) and 4(b) show reconstructions, using the MPUI method with NENR–computed normal, from a couple of real 3D models widely used in the literature. The MPUI parameters for both models were, a grid size of 0.004, and a max error of 0.002 at each cell. The rest of the parameters took the default values setting by the Ohtake's MPUI implementation. For the bunny model the NERN parameters were $k = 15$, $\alpha = 100$, $\beta = 0.01$ and in the case of the golf club model were $k = 12$, $\alpha = 1000$, $\beta = 0.01$.

6 Conclusions

In general, the normal estimation algorithms based on the covariance matrix as NE–Hoppe approximate a neighborhood by a unique plane independently of its local shape. Some algorithms, in order to improve the approximation by a unique plane, reduce the neighborhood size or weight the covariance matrix, nevertheless the approximation continues to be a unique plane.

The NENR algorithm produces a reduction in the neighborhood size, rejecting the neighbors that have large differences. This condition warranties that the neighbors have the same smoothness degree between them. The NERN algorithm was tested using synthetic examples building with shape discontinuities. The experiments presented better quantitative and qualitative results using NENR than NE–Hoppe.

In cases of real models, a couple of experiments were done and the results were very similar for both algorithm, therefore, using NENR in real smooth models does not represent a lost in quality for surface reconstruction. In general, NENR produces a better normal estimation than NE–Hoppe, in places located near to edges, corners and geometric discontinuities. The NERN algorithm does not have troubles with smooth regions or smooth models.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: SIGGRAPH 1992: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 71–78. ACM Press, New York (1992)
2. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.P.: Multi-level partition of unity implicits. In: SIGGRAPH 2003: ACM SIGGRAPH 2003 Papers, pp. 463–470. ACM Press, New York (2003)
3. Bajaj, C.L., Bernardini, F., Xu, G.: Automatic reconstruction of surfaces and scalar fields from 3d scans. In: SIGGRAPH 1995: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 109–118. ACM Press, New York (1995)
4. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: SIGGRAPH 2001: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 67–76. ACM Press, New York (2001)
5. Pauly, M., Keiser, R., Kobbelt, L.P., Gross, M.: Shape modeling with point-sampled geometry. In: SIGGRAPH 2003: ACM SIGGRAPH 2003 Papers, pp. 641–650. ACM Press, New York (2003)
6. Rivera, M., Marroquin, J.L.: The adaptive rest-condition spring system: An edge-preserving regularization techique. In: ICIP-2000, vol. II, pp. 805–807. IEEE Signal Processing Society, Vancouver, BC, Canada (2000)
7. O'Rourke, J.: Computational geometry in C. Cambridge University Press, New York (2000)

8. Amenta, N., Bern, M., Kamvysselis, M.: A new voronoi-based surface reconstruction algorithm. In: SIGGRAPH 1998: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 415–421. ACM Press, New York (1998)
9. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: SMA 2001: Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 249–266. ACM Press, New York (2001)
10. Dey, T.K., Goswami, S.: Tight cocone: a water-tight surface reconstructor. In: SM 2003: Proceedings of the eighth ACM symposium on Solid modeling and applications, pp. 127–134. ACM Press, New York (2003)
11. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH 1987: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 163–169. ACM Press, New York (1987)
12. Bloomenthal, J.: An implicit surface polygonizer, 324–349 (1994)
13. Hart, J.C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer 12, 527–545 (1996)
14. Black, M., Rangarajan, A.: On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. Int'l J. Computer Vision 19, 57–92 (1996)
15. Charbonnier, P., Blanc-Feraud, L., Aubert, G., Barluad, M.: Deterministic edge-preserving regularization in computed imaging. IEEE Trans. Image Processing 6, 298–311 (1997)
16. Rivera, M., Marroquin, J.L.: Adaptive rest condition potentials: first and second order edge-preserving regularization. Journal of Computer Vision and Image Understanding 88, 76–93 (2002)
17. Rivera, M., Marroquin, J.: Half–quadratic cost functions with granularity control. Image and Vision Computing 21, 345–357 (2003)
18. Calderon, F., Romero, L.: Non-parametric image registration as a way to obtain an accurate camera calibration. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 584–591. Springer, Heidelberg (2004)
19. Calderon, F., Romero, L., Flores, J.: Ga-ssd-arc-nlm for parametric image registration. In: Martínez-Trinidad, J.F., Carrasco Ochoa, J.A., Kittler, J. (eds.) CIARP 2006. LNCS, vol. 4225, pp. 227–236. Springer, Heidelberg (2006)
20. Ohtake, Y.: Mpui implementation in (2003), `http://www.mpi-inf.mpg.de/~ohtake/mpu_implicits/`