Surface reconstruction using Power Watershed

Camille Couprie*, Xavier Bresson†, Laurent Najman*,
Hugues Talbot*, and Leo Grady‡

∗ Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI, ESIEE Paris (93160 Noisy-le-Grand, France)
† City University of Hong Kong, Dpt. of Computer Science (Hong Kong)
‡ Siemens Corporate Research, Dpt. Imaging Analytics & Informatics (Princeton USA)
{c.couprie,l.najman,h.talbot}@esiee.fr,
xbresson@cityu.edu.hk,leo.grady@siemens.com

Abstract. Surface reconstruction from a set of noisy point measurements has been a well studied problem for several decades. Recently, variational and discrete optimization approaches have been applied to solve it, demonstrating good robustness to outliers thanks to a global energy minimization scheme. In this work, we use a recent approach embedding several optimization algorithms into a common framework named power watershed. We derive a specific watershed algorithm for surface reconstruction which is fast, robust to markers placement, and produces smooth surfaces. Experiments also show that our proposed algorithm compares favorably in terms of speed, memory requirement and accuracy with existing algorithms.

Keywords: optimization, point measurements, Graph cuts, total variation

1 Introduction

This paper develops a watershed-based algorithm providing a global optimal solution to the surface reconstruction problem from a set of scattered points. Surface fitting is a challenging problem when dealing with data containing sparse noise, gaps, and outliers. The set of points may be for example acquired by several scans of an object (range scanning). In this context, regularization-based methods have been shown to be robust when the points are lacking connectivity, ordering information, and may be contaminated by noise. While there exist numerous explicit surface extraction techniques that estimate the exact positions of surface points, in this work we will focus on implicit surface representation. Implicit surfaces may be represented by level sets (*e.g.* [29]) or binary partitions (*e.g.* [23]).

Local methods for surface reconstruction including the MPU method [5] are sensitive to noise, as shown in the experiments of [19]. Among the recent global approaches, the Poisson method [20] and FFT-method [21] are more robust to noise, however they require orientation information.

The method of Jalba and Roerdink [19] makes it possible to avoid having to estimate orientation information. This is achieved by computing approximations of Coulomb potentials in a grid as an input for the convection method of [29]. We propose a different approach that allows us to perform a global optimization of the surface reconstruction problem without requiring the computation of such field as Coulomb potentials.

A generic optimization-based regularization formulation for shape fitting minimizes the total variation functional weighted by the distance function from the set of points P. More generally, given two positive numbers p and q, we consider an object indicator partition u solution of

$$\begin{array}{c} \min\limits_{u\in[0,1]} \int_{\Omega} w(z)^p |\nabla u(z)|^q \mathrm{dz} \\ \text{subject to } u(z) = 0 \ \ \forall z \in \Omega_{\text{in}}, \\ \text{and } u(z) = 1 \ \ \forall z \in \Omega_{\text{out}}, \end{array} \tag{1}$$

where Ω_{in} is the set of labels inside the surface and Ω_{out} is the set of labels outside the surface. The weight function w is defined at every point z in a grid as $w(z) = d_P(z)$, where $d_P(z)$ is the distance map from the points. When p is finite and $q = 1$, Eq. (1) leads to a binary solution u [27]. A solution can be deduced in the discrete setting using *e.g.* the network flow technique [12], also known as Graph cuts [4]. Augmenting path max flow implementations are fast and efficient in 2D but memory consuming as the connectivity increases, for instance in 3D. Lempitzky and Boykov [23] have overcome this problem by limiting the size of the search for a solution in the grid while still guaranteeing a global optimum. However their solution is based on restrictive assumptions assuming that the data points are provided with an estimate of the surface orientation. Furthermore, at low resolution, results exhibit metrication artifacts and look blocky, so a high resolution is essential for getting smooth results using the Graph cuts method.

We propose a watershed-based approach in order to provide a way to quickly obtain smooth surfaces at a high resolution without any need to pre-estimate the surface orientation. Recently, Couprie *et al.* introduced the power watershed method [10, 8], which can be seen as an anisotropic discretization of (1) with $p \to \infty$ and $q = 2$. Although this technique was introduced in the context of image segmentation, the authors described how the method could be used as an optimization method for various functionals in [9]. In the present paper we show that the power watershed method of Couprie *et al.* is well-suited to address the surface reconstruction problem. The idea of using watersheds for surface reconstruction from a set of points is quite natural. It can be seen as a an extension of the classical "coffee bean" segmentation example, where a watershed is applied on a filtered distance function to separate overlapping convex objects [3].

For further comparison, we also propose in this paper to examine a well-known weighted isotropic discretization of (1) (for $q = 1$) known as "Total Variation" (TV). We mention that there exist other TV-based approaches developed to overcome metrication artefacts, *e.g.* [28, 15]. However, due to space

constraints, further comparison with these techniques will be the subject of future work. Finally, we will also show that our algorithm compares favorably with existing surface reconstruction algorithms [2, 5, 21, 20, 19].

2 Method

For solving the surface reconstruction problem, we first place the points cloud onto a regular grid. Our method aims to label the nodes of the grid as an indicator of the object to reconstruct. Since the power watershed is defined on a graph, we begin by casting the surface reconstruction problem formulation in discrete terms.

A graph consists of a pair $G = (V, E)$ with vertices $v \in V$ and edges $e \in E \subseteq V \times V$ with cardinalities $n = |V|$ and $m = |E|$. An edge, e, spanning two vertices, v_i and v_j , is denoted by e_{ij}. We define an edge set corresponding to a 4 or 8-connected lattice (or 6-, 18- or 26-connected in 3D). A weighted graph assigns a (typically non-negative and real) value to each edge called a weight. The weight of an edge e_{ij} is denoted by $w(e_{ij})$ or w_{ij}. A plateau is a maximal set of connected edges with identical weight.

Given foreground F and background B node values (also called seeds), and p, q two real positive values, the energy presented for binary segmentation in [8] is a discretization of (1) given by

$$\begin{aligned} &\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q \\ &\text{s.t. } \; x(F) = 1, \; x(B) = 0. \end{aligned} \tag{2}$$

In this energy, x is a labeling indicating the foreground and background membership. The edge values w_{ij} can be interpreted as weights enforcing a regularization of the contours, such that any (usually unwanted) high-frequency content is penalized in x. The definition of the weights for surface reconstruction from a set of points P is based on the construction of a discrete distance map d_P, in a grid bounding the set of points.

$$w_{ij} = \min\left(d_P(i), d_P(j)\right), \tag{3}$$

where $d_P(i)$ is the discrete Euclidean distance between the node i and the set of points P. We recall that exact Euclidean discrete distance map may be obtained in linear time using the algorithm of Hirata [17], and that high-quality ordered algorithms also exist [25] . The background seeds may simply correspond to the frame, or bounding box of the lattice. The foreground seeds can be given by the maxima of the distance function that are not connected to the frame. The distance map may be previously filtered, for example using an attribute filter to obtain more robust markers [6, 24].

As we illustrate in the remainder, the energy defined in (2) essentially forces x to remain smooth within the object, while allowing it to vary quickly close to point clusters near the boundary of the object. The data constraints enforce

fidelity of x to a specified configuration, taking the values zero and one as the reconstructed object indicator. Observe that the values of x may not necessarily be binary when the value of q is strictly greater than one, which is a positive point for the surface reconstruction problem as we will further explain in this work.

The different values of p and q lead to different algorithms for optimizing the energy. When the power of the weight, p, is finite, and the exponent $q = 1$, we recover the Graph cuts energy which can be optimized by a max flow algorithm. When $q = 2$, we obtain a combinatorial Dirichlet problem also known as the Random walker problem [16]. As described in [8, 10], when the exponent p tends toward infinity, the cut obtained when minimizing the energy is a watershed cut [11], which has been shown to be equivalent to Maximum Spanning Forests [11] (MSF). Furthermore, an algorithm is presented to optimize the unique watershed that optimizes the energy for $q = 2$ and $p \to \infty$. The power watershed (PW) algorithm is recalled in Algorithm 1.

Algorithm 1: Power watersheds algorithm $p \to \infty, q = 2$

Data: A weighted graph $G(V, E)$ comprising known labels $x(B), x(F)$.
Result: A **labeling** x solution of (2).
while *any node has an unknown label* **do**
  Find a maximal subgraph $S \in G$ composed of edges of maximal weight;
  if *S contains any nodes with known x* **then**
    Find x_S minimizing (2) for $q = 2$ on the subset S;
    Consider all x_S values produced by this operation as known;
  else
    Merge all of the nodes in S into a single node, such that when the value of x for this merged node becomes known, all merged nodes are assigned the same value of x and considered known;

This set of parameters $q = 2$ and $p \to \infty$ is particularly interesting :

1. The power watershed algorithm has a well-defined behavior in the absence or lack of weight information (presence of plateaus). An example is shown at Figure (1).
2. The worst-case complexity of the power watershed algorithm in the case $p \to \infty$ is given by the cost of optimizing (2) for the given q. In the best-case scenario (all weights have unique values), the power watershed algorithm has the same asymptotic complexity as the algorithm used for a MSF computation (quasi-linear) (See [7] for more details). In practical applications where the plateaus have a size less than some fixed value K, then the complexity of the power watershed algorithm matches the quasi-linear complexity of the standard watershed algorithm.

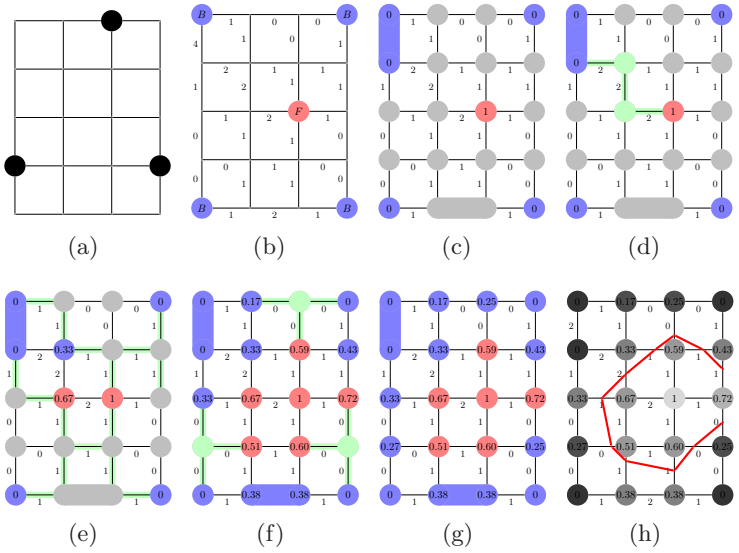
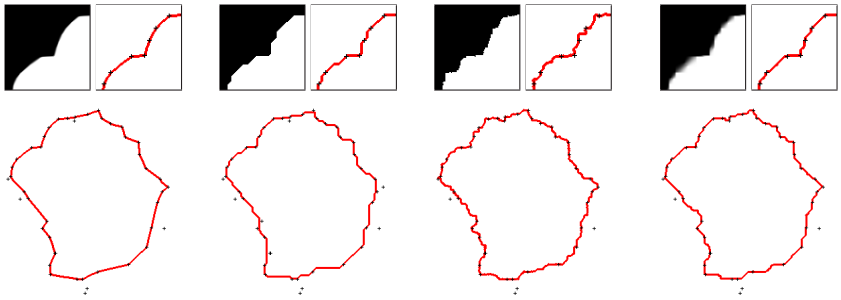


Fig. 1. (a) Three dots in a 4×5 lattice. (b) Associated lattice weighted by a distance map from the dots according to (3) and squared, with Foreground and Background seeds. Note that only the ordering of the weights counts in the power watershed algorithm, thus the squared distance map can be used directly and produces the same solution as if the weights in (3) were used. (c) First steps of the power watershed algorithm to optimize (2) in the case $q = 2$ and $p \to \infty$. Nodes having a maximum weight are merged. (d) A plateau of weight 2 (in green) including different seeded nodes is encountered. The Random walker algorithm is applied to label the nodes on the plateau. (e,f) New plateaus of weight 1 and 0 are encountered, the Random walker algorithm is applied, (g) Final labeling x solution of (2). The isocontour is represented in red.

Although the PW algorithm is fast, it can be further accelerated for the specific case of weight defined according to a distance map (3). Given the foreground and background seeds, we define a narrow band $\mathcal{S}$ as the set obtained by thresholding the distance map d_P with the the smallest threshold T_S such that the connected components are divided between at least an interior (foreground) and an exterior (background). We then compute the PW only on this incomplete distance map, which saves both time and memory. In practice, it is possible to avoid computing an exact distance map on the full grid, using for instance an ordered algorithm propagating from the point cloud [25].

Recall that the exterior seed is connected to the frame of the image, so this is a simple unambiguous connectivity criterion. Applying PW on $\mathcal{S}$ is guaranteed to provide the same (unique) global optimizer of the energy as the solution on the full grid, because the connectivity criterion ensures this computation yields a Jordan curve (2D) or surface (3D) separating foreground and background seeds and passing through only already-computed distance values. Performing the PW



(a) Total variation result (b) Graph cuts result (c) Max Spanning Forest result (d) Power watershed result

Fig. 2. Comparison of surface reconstruction from a set of points in 2D, using the total variation method, Graph cuts, a maximum spanning forest algorithm and finally the power watershed algorithm.

computation on the full map would not change this result because no distance weight would be lowered, and so no new surface with a lower weight could be found on this full map. Conversely, thresholding the distance map at a lower value than T_S *would* change the result, as any Jordan curve or surface separating foreground and background seeds and computed on this map would necessarily cross some nodes where the distance map had not been computed. Note this does not mean that this thresholding criterion is necessary and sufficient for optimal computation and least memory usage, as an adaptive threshold depending on the local point density could be used instead. However, we leave it as future work to find a better criterion than the simple global threshold.

3 Results and comparative evaluation

3.1 Comparison with Graph cuts and Total Variation

We now demonstrate the performance of the power watershed algorithm for surface reconstruction with respect to two graph-based methods discretizing the energy defined in equation (1), namely the weighted total variation (TV) and the Graph cuts (GC) method.

Our first experiment consists of finding a contour fitting sparse and noisy dots in a 2D plane. Figure 2 compares the result of TV minimization, the Graph cuts result, a maximum spanning forest result, and the result obtained with power watershed algorithm ($q = 2$, $p \to \infty$). We observe that all resulting contours are excluding outliers. The Graph cuts results demonstrate that this algorithm is less sensitive to noise, but the contours are blocky because the obtained object indicators are binary. We note that a post-processing step for producing a smooth isosurface from such a binary object reconstruction has recently been proposed by Lempitzky [22]. The TV contour is the smoothest

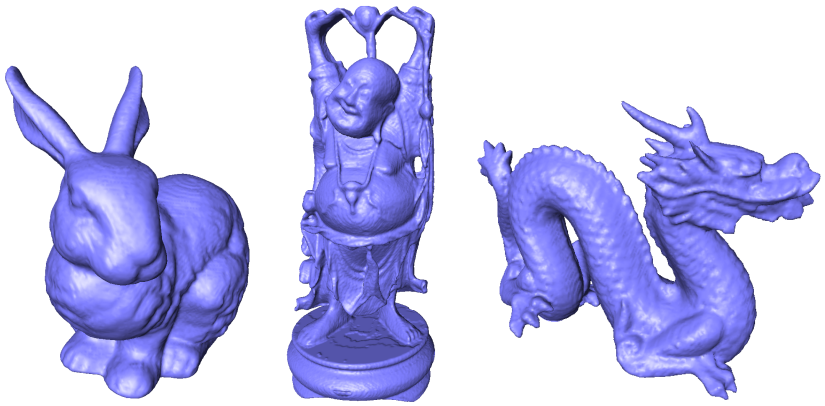


Fig. 3. Power watershed results obtained from points clouds. (a) Total size of scans : 362272 points, Grid size : $234 \times 297 \times 301$. (b) Number of scans used : 341072 points, Grid size : $275 \times 276 \times 668$ (c) Total size of scans : 2748318 points, Grid size $382 \times 270 \times 171$.

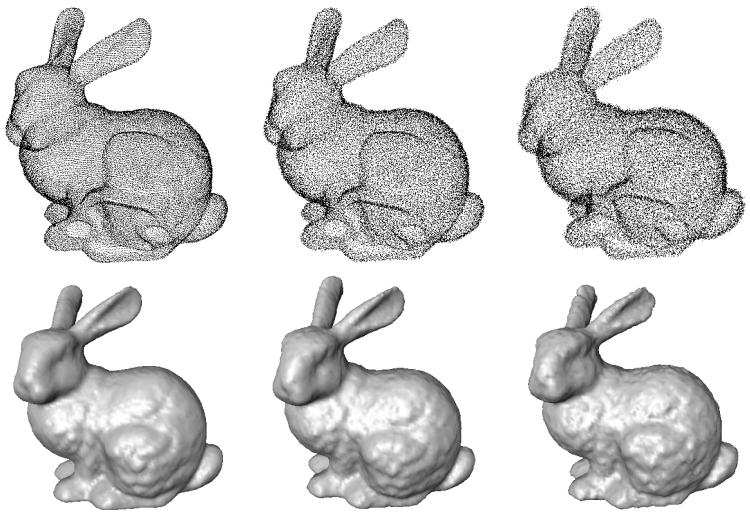
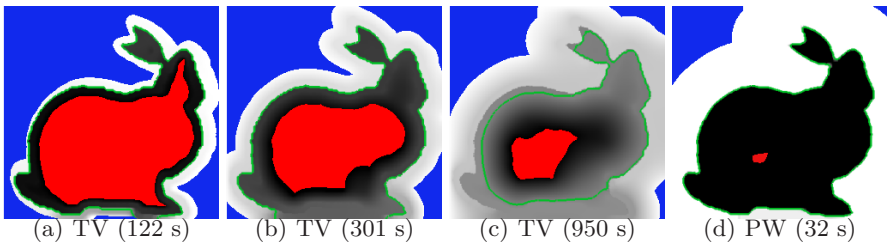


Fig. 4. Power watershed results obtained from noisy points clouds, corrupted by Gaussian noise of variance σ. The error was computed as the average distance between the obtained isosurface points to the original point cloud. The error is given in percentages of the diagonal of the bounding box of the data points.



(a) TV (122 s) (b) TV (301 s) (c) TV (950 s) (d) PW (32 s)

Fig. 5. Robustness to seed quantity: this figure shows slices of solutions obtained for reconstructing the bunny surface, obtained on a grid of size $251 \times 248 \times 195$, using different seeding strategies. The interior seed is colored in red, and the background seed in blue, and resulting isolines in green. (a,b,c) : Results obtained with TV minimization, and (d) Result obtained with Power Watershed.

one, which may be an unwanted effect for rendering details in surfaces. Thus, the TV method requires us to adapt the parameter p in the exponent of the weights to the desired smoothness of the surface. The maximum spanning forest result passes through most points and the contour looks noisy. The power watershed result demonstrates good performance for fitting the dots, while avoiding both blocky contours and noise. This good performance is due to the presence of interwoven plateaus in the distance map. During the execution of the algorithm, the Random walker (RW) algorithm is called several times around the dots, resulting in a smooth output x. An isocontour or isosurface computation at the 0.5 level is thus providing smooth contours compared to the binary results obtained with Graph cuts or maximum spanning forests.

Figure 3 shows surfaces reconstructed from noisy scanned dot sets using the power watershed (PW) algorithm. We used the coordinates of points acquired from scans of several 3D shapes (bunny, Buddha) from the Stanford database available online [1] In our experiments, we embedded those points in 6-connected grids. Quantitative comparisons for the fitting quality are difficult because not all data points are required to be part of the surface. However, we show in Figure 4 that the power watershed method is producing reasonable results even if the point cloud is corrupted by Gaussian noise. We also compared our results to the results obtained using TV minimization and Graph cuts at Figure 6. We can observe that the surface obtained with the Graph cuts method is quite blocky. Using the same rendering method to render the output x minimizing the power watershed energy, the power watershed algorithm obtains a smoother surface showing significantly more details. Figure 5 shows that the power watershed method performs well even with a small amount of seeds. In contrast, the total variation method requires a large amount of seeds placed close to the searched surface.

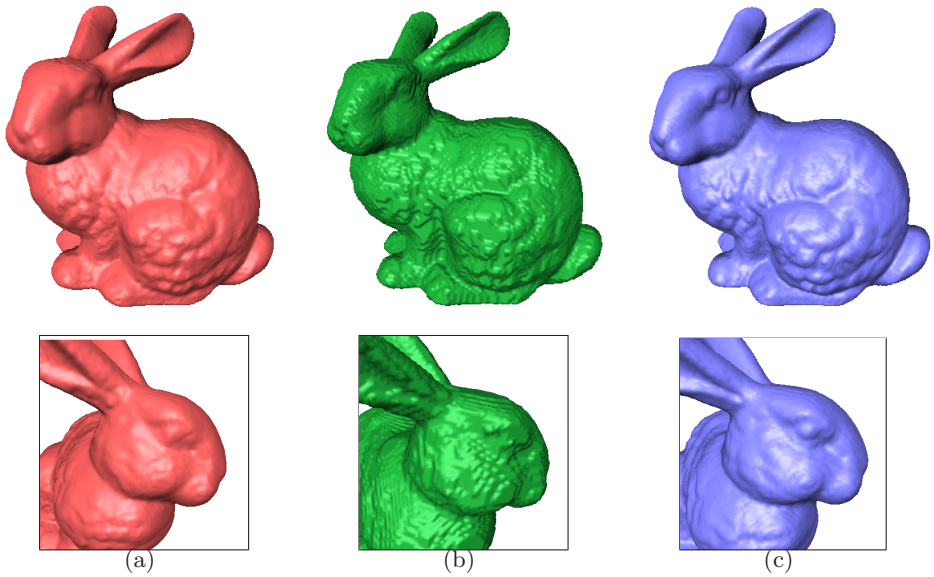


Fig. 6. Grid size : $234 \times 297 \times 301$. (a) Total variation minimization result, (b) Graph cuts result, (c) power watershed result. Isosurfaces at 0.5 have been extracted on all results, and were downsampled by 2 to render the surfaces.

3.2 Computation times and memory requirements

In our comparisons, we used the C++ software library of Lemptitzky and Boykov available online which implements the touch-expand algorithm that minimizes the GC energy. For that purpose, the touch-expand algorithm calls a max flow algorithm on a partial graph, a band around the points which is extended when the solution touches the boundary of the band. We also compare to a fast TV-based solver often called split-Bregman algorithm [14] but identical in principle to the Alternating Direction Method of Multipliers [13, 26]. This TV solver is implemented in C and may be called from Matlab. Finally, we also implemented the PW method in C. Computation times of the compared algorithms for the bunny and Buddha data sets are provided on Table 1. The three methods have large memory requirements when applied to the full grid. However, the graph cuts and power watershed banded methods are less memory intensive. For the bunny dataset, the touch-expand algorithm optimizing the Graph cuts energy only needs to allocate 3.6% of the full grid size. The solution space for the bunny dataset is much larger for our power watershed algorithm and reaches 33% of the full grid, because some points are widely spaced out. Thus, on this image, the touch expand algorithm is faster than the power watershed method. However, points clouds may require a more extensive expansion of the research area for the Graph cut touch expand method. For example, in the Buddha dataset, 8.75%

of the grid size is needed for computing the Graph cuts solution. In such cases, the power watershed approach is much faster. The TV approach has currently no guarantee to produce a global optimum when called on a banded graph, so the TV algorithm was implemented only on a full grid. On the Buddha dataset for example, given a $207 \times 505 \times 207$ grid, the TV-based method requires 3G of RAM.

Point cloud	Method	Grid size	Grid used	Peak memory	Time	Time / Nb voxels
Bunny	GC	$234 \times 257 \times 301$	3.6%	212	9	1.4×10^{-5}
Bunny	PW	$234 \times 257 \times 301$	31%	1180	51	9.1×10^{-6}
Bunny	TV	$195 \times 248 \times 251$	100%	900	122	1.0×10^{-5}
Buddha	GC	$276 \times 671 \times 277$	8.7%	850	253	5.6×10^{-5}
Buddha	PW	$276 \times 671 \times 277$	16%	1500	52	6.3×10^{-6}

Table 1. Timing experiments (in seconds) using an PC with a 3GHz Intel dual-core processor and 2G of RAM. The memory requirements are given in Mega Octets, and the computation times in seconds. Note that the seeds used are different for the three methods. For TV and PW, the seeds are imposed as hard constraints and are located far from the point cloud (see Fig. 5). The GC touch expand method uses soft constraints seeds computed from the normals given in input with the point cloud. We observe that Graph cuts are fast when the point cloud configuration does not require to expand the research area too much, like in the Bunny case (3.6%). However in the Buddha example, there is a need for a larger expansion, where the real complexity of the Graph cut algorithm become visible.

3.3 Comparison with some other approaches

Following the quantitative information given in [19], we compare the performances of power watershed for the Stanford bunny reconstruction with different methods [2, 5, 21, 20, 19]. All these methods, including the power watershed, reconstruct a surface close to the Bunny set of dots, with an error comprised between 2×10^{-4} and 6×10^{-4}. In terms of computation time, the Power Crust method [2] is about 10 times slower, and the Poisson method [20] 3 times slower than our PW approach. Although the FFT [21], MPU [5] and Hoppe *et al* [18] methods are fast, the FFT method suffers from large memory requirements limiting the grid resolution, and Hoppe *et al* and MPU methods produces artifacts in the presence of noise (See [19]). The method of Jalba and Roerdink [19], based on Coulomb potentials, uses 4 times less memory than our power watershed method, but is 5 to 10 times slower on a CPU and is still slower using a GPU. Furthermore our PW is more flexible in the choice of the markers. In our experiments, the amount of markers is not very large as shown in Fig. 5, but a

strategy using larger seeds could be employed to reduce the size of the solution space and the computation time.

4 Conclusion

The power watershed method can be used to efficiently produce surfaces fitting noisy measurements. Contrary to standard watershed algorithms and the Graph cuts approach, the unique solution provided by the power watershed is not binary, resulting in the reconstruction of both smooth and detailed surfaces. In addition, this method is fast and not limited by large memory requirements, when using a restriction of the solution space. Finally, in comparison with other methods, our power watershed is robust to seed placement, and requires fewer parameters to be set. In practice, when using the power watershed method, close-fitting markers are not as mandatory as in other methods. Future work will follow several directions: memory and computation times improvements are still achievable, in particular by improving the touch-expand idea used in the Graph cuts optimization and adapting it for power watershed. The power watershed energy could also be modified to add some priors to the reconstructed surface, such as local orientation. Finally, we hope to demonstrate the efficiency of the power watershed technique for solving related problems such as multiview reconstruction.

References

1. Stanford 3D scanning repository, http://graphics.stanford.edu/data/3Dscanrep/
2. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: Proceedings of the sixth ACM symposium on Solid modeling and applications. pp. 249–266. SMA '01, ACM, New York, NY, USA (2001)
3. Beucher, S., Gratin, C.: Micromorph reference manual, applications and solutions. Ecole des Mines de Paris (1989)
4. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)
5. Braude, I., Marker, J., Museth, K., Nissanov, J., Breen, D.: Contour-based surface reconstruction using mpu implicit models. GRAPHICAL MODELS 69, 2007
6. Breen, E., Jones, R.: Attribute openings, thinnings and granulometries. Graphical Models and Image Processing Journal 64(3), 377–389 (1996)
7. Chazelle, B.: A minimum spanning tree algorithm with inverse-ackermann type complexity. J. ACM 47, 1028–1047 (November 2000)
8. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watersheds: a new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In: ICCV'09. pp. 731–738 (2009)
9. Couprie, C., Grady, L., Najman, L., Talbot, H.: Anisotropic Diffusion Using Power Watersheds. In: ICIP'10. pp. 4153–4156 (2010)
10. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power Watersheds: A Unifying Graph Based Optimization Framework. IEEE Transactions on Pattern Analysis and Machine Intelligence (2011), to appear

11. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(8), 1362–1374 (2009)
12. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Canadian Journal of Mathematics 8, 399–404 (1956)
13. Glowinski, R., Tallec, P.: Augmented Lagrangian and operator-splitting methods in nonlinear mechanics. SIAM (1989)
14. Goldstein, T., Osher, S.: The split Bregman method for ℓ_1-regularized problems. SIIMS 2(2), 323–343 (2009)
15. Goldstein, T., Bresson, X., Osher, S.: Geometric applications of the split Bregman method: Segmentation and surface reconstruction (2009)
16. Grady, L.: Random walks for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11), 1768–1783 (2006)
17. Hirata, T.: A unified linear-time algorithm for computing distance maps. Information Processing Letters 58(3), 129–133 (1996)
18. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. SIGGRAPH Comput. Graph. 26, 71–78 (July 1992)
19. Jalba, A.C., Roerdink, J.B.T.M.: Efficient surface reconstruction using generalized coulomb potentials. IEEE Transactions on Visualization and Computer Graphics 13, 1512–1519 (November 2007)
20. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. pp. 61–70. SGP '06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)
21. Kazhdan, M.M.: Reconstruction of solid models from oriented point sets. In: Symposium on Geometry Processing. pp. 73–82 (2005)
22. Lempitsky, V.: Surface extraction from binary volumes with higher-order smoothness. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2010)
23. Lempitsky, V., Boykov, Y.: Global Optimization for Shape Fitting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Minneapolis, USA (2007)
24. Najman, L., Couprie, M.: Building the component tree in quasi-linear time. IEEE Transactions on Image Processing 15(11), 3531–3539 (2006)
25. Ragnemalm, I.: The euclidean distance transform in arbitrary dimensions 14(11), 883 – 888 (1993)
26. Setzer, S.: Split Bregman algorithm, douglas-rachford splitting and frame shrinkage. In: International Conference on Scale Space and Variational Methods in Computer Vision. pp. 464–476. SSVM '09, Springer-Verlag, Berlin, Heidelberg (2009)
27. Strang, G.: Maximum flows through a domain. Math. Prog. (26), 123–143 (1983)
28. Ye, J., Bresson, X., Goldstein, T., Osher, S.: A fast variational method for surface reconstruction from sets of scattered points (2010), submitted
29. Zhao, H.K., Osher, S., Merriman, B., Kang, M.: Implicit, nonparametric shape reconstruction from unorganized points using a variational level set method. Computer Vision and Image Understanding 80, 295–319 (1998)