Reconstruction of Surfaces from Point Clouds Using a Lagrangian Surface Evolution Model

Patrik Daniel, Matej Medl'a, Karol Mikula, and Mariana Remešíková(✉)

Department of Mathematics, Faculty of Civil Engineering,
Slovak University of Technology, Radlinského 11, 81005 Bratislava, Slovakia
patrik.daniel1@gmail.com, {medla,mikula,remesikova}@math.sk

Abstract. We present a method for reconstruction of surfaces in R^3 from point clouds. Given a set of points, we construct a triangular mesh approximation of a surface that they represent. The triangulation is obtained by a Lagrangian surface evolution model consisting of an advection and a curvature term. To construct them, we compute the distance function d to the given point cloud. Then the advection evolution is driven by ∇d and the curvature term depends on d and the mean curvature of the evolving surface. In order to control the quality of the mesh during the evolution, we perform tangential redistribution of mesh points as the surface evolves.

Keywords: Point cloud · Surface evolution · Lagrangian methods · Tangential redistribution

1 Introduction

One of the main tasks of modern computer graphics and computer vision is to obtain digital representation of real world objects. A common way of obtaining a representation of a 3D object is scanning a set of points lying on the object's surface. However, such a basic representation is not sufficient for most applications. Therefore, a lot of effort has been put in designing algorithms for obtaining a surface representation of an object given a representative point cloud [1].

We present a method that constructs a triangular mesh representation of an object's surface. As an input, we need the corresponding point cloud and a triangulated surface – an approximation of the desired surface. After, we apply an appropriately designed Lagrangian surface evolution model to obtain the representation we are looking for. The driving force of the evolution is the distance function d to the point cloud – the model consists of an advection term with the velocity proportional to ∇d and a curvature term with the evolution speed proportional do d. Moreover, the model is enriched with a specifically designed tangential movement term that moves the points around the evolving surface. This term is added to overcome one of the main difficulties arising in Lagrangian manifold evolution – the possible deterioration of the mesh quality during the evolution process.

J.-F. Aujol et al. (Eds.): SSVM 2015, LNCS 9087, pp. 589–600, 2015.
DOI: 10.1007/978-3-319-18461-6_47

The method that we propose is that it provides a straightforward way to obtain a good quality triangular representation of an object's surface. The initial condition for the evolution process is usually a simple surface (e.g. a sphere or an ellipsoid) that can be easily triangulated. Assuming that the initial condition is topologically equivalent to the desired surface, the topology of the triangulation remains intact during the evolution. The orientation of normals is also preserved; having oriented normals is important for determining the interior and exterior of the surface, resolving visibility, shading etc. However, a practically applicable method should also be robust enough to manage difficulties caused by various data imperfections. The scanned point clouds often suffer from defects such as non-uniform point distribution, noise, outlying points or missing parts (Figure 1). As we demonstrate in the last section, our method is well able to deal with such problematic situations due to the properties of the model that we use.

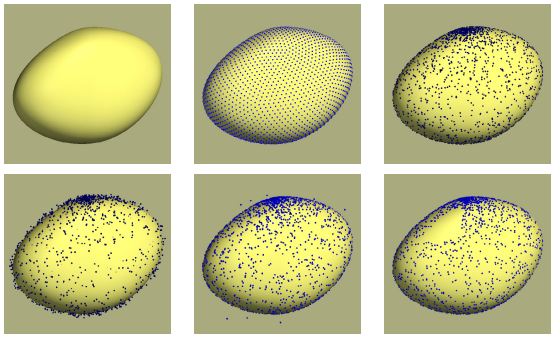

Fig. 1. A surface and its various point cloud representations – an ideal uniform point cloud, a non-uniform point cloud, a point cloud with 5% noise, a point cloud with several outlying points and a point cloud with a missing part.

The reason why we use a Lagrangian method is that obtaining a triangular representation of an object is more simple than in the case of, e.g., level set methods. Also, though not shown here, Lagrangian methods can be more easily applied to surfaces with boundaries. To our knowledge, Lagrangian evolution methods were so far only rarely applied to the point cloud problem and the existing works do not consider any mesh adjustment [7]. The strategy that we use in this paper is based on some of our previous works concerning surface evolution problems [5,6]. This paper extends the ideas to the point cloud problem that has not appeared in the cited works. Also, we suggest a new tangential redistribution technique based on the curvature of the evolving surface.

2 Mathematical Model

Let $\Omega \subseteq \mathbb{R}^3$ and let $C \subset \Omega$ be a set of n_p points with elements denoted by P_i, $i = 1 \ldots n_p$. Our goal is to find a closed surface approximating C.

Let $d_0 \colon \Omega \to \mathbb{R}$ represent the distance function to C. Starting at any point of Ω, we can approach C following the direction of ∇d_0. The basic idea of our

mathematical model is to take a smooth closed surface surrounding C and let it evolve based on ∇d_0. Since d_0 is not everywhere differentiable, instead we will consider $d = G_\sigma * d_0$, where G_σ is a Gauss kernel.

Now, let X be a two-dimensional Riemannian manifold equipped with the metric g_X. Let $F\colon X \times \langle 0, t_s\rangle \to \Omega$ denote a time-dependent embedding of X in Ω and let g_F represent the corresponding induced metric. The image of $F^t = F(\cdot, t)$ will be denoted by S^t. The embedding F is the solution of the evolution equation

$$\partial_t F = w_a \left(\nabla d \cdot N\right) N + w_c d \Delta_{g_F} F + v_T, \tag{1}$$

where N is a unit normal to S. The symbol $\Delta_{g_F} F$ is the Laplace-Beltrami operator with respect to g_F that is equal to the mean curvature vector h of F. The parameters w_a and w_c are non-negative reals. The last term v_T represents the velocity of the tangential movement. This model, besides the tangential movement, is a Lagrangian analogue of the level set model used by Zhao [9].

To understand the mechanism of action of this model, let us take a closer look at the three terms on the right hand side. As written above, the first term represents the driving force that makes S^t approach C. The projection to the surface normal eliminates tangential movement that does not affect S^t and it also makes the movement of a point dependent on its neighborhood – the surface is moving as a whole rather than a set of independent points. If ∇d points in a tangential direction, the advective evolution stops and thus a surface patch is formed in the empty space between the points of C. The second term represents evolution by mean curvature that speeds up the evolution in the regions distant to C and it regularizes the surface during the evolution. It also causes forming of straight patches between the points of the point cloud. Moreover, as we will see later, it helps to deal with artifacts like noise or outlying points. The third term is used to control the induced metric g_F; in the discrete setting it means we can distribute mesh points around the surface according to our needs.

In our previous work [6], we suggest a method for constructing the tangential velocity based on the evolution of the induced metric, particularly the area density $G\colon X \times \langle 0, t_s\rangle \to \mathbb{R}$ defined as $G = \frac{\partial \mu_F}{\partial \mu_X}$, where μ_F and μ_X are the measures induced on X by g_F and g_X. This quantity expresses how much the embedding F locally shrinks or expands areas. It is easy to see this considering G constant over a domain $U \subset X$; then $\mu_F(U) = G\mu_X(U)$. From the discrete point of view it means that push-forwarding a discretization of X along F, the increase of the density of discretization points will be higher in regions with lower values of G. The evolution of G is given by [3]

$$\partial_t G = \left(-v_N \cdot h + \operatorname{div}_{g_F} w_T\right) G, \tag{2}$$

where h is the mean curvature vector, $v_N = w_a \left(\nabla d \cdot N\right) N + w_c d \Delta_{g_F} F$ and w_T is a vector field on X constructed as the pull-back of v_T along F. The area of X measured by μ_F (the area of S) evolves as

$$\partial_t G = \int_X \left(-v_N \cdot h\right) G \,\mathrm{d}\mu_F. \tag{3}$$

An embedding F with constant G can be called *area-uniform* with respect to g_X. Let us use the notation $A_X = \mu_X(X)$, $A = \mu_F(X)$. Since

$$\int_X G \, \mathrm{d}\mu_X = A, \tag{4}$$

then for a constant G we must have $G = \frac{A}{A_X}$.

An embedding with this property provides a straightforward way to construct a discretization mesh with uniformly sized 2D mesh elements. This type of mesh has several practical advantages – it prevents mesh degeneration, it provides a reasonable discrete representation of a surface and it is likely to capture important information coming from the external vector field that drives the evolution. Thus, it has sense to require $G \to \frac{A}{A_X}$ as $t \to \infty$. The corresponding dimensionless condition is

$$\frac{G}{A} \underset{t\to\infty}{\longrightarrow} \frac{1}{A_X}.$$

This can be guaranteed, if

$$\partial_t \left(\frac{G}{A} \right) = \omega \left(\frac{1}{A_X} - \frac{G}{A} \right), \tag{5}$$

where $\omega \colon \langle 0, t_s \rangle \times \mathbb{R}_+$. This equality combined with (2) and (3) yields a condition for the divergence of w_T,

$$\operatorname{div}_{g_F} w_T = v_N \cdot h - \frac{1}{A} \int_X v_N \cdot h \, \mathrm{d}\mu_F + \omega \left(\frac{A}{A_X G} - 1 \right). \tag{6}$$

To obtain a unique w_T, we can suppose, for example, that $w_T = \nabla_{g_F} \psi$, $\psi \colon X \times \langle 0, t_s \rangle \to \mathbb{R}$. Then we have

$$\Delta_{g_F} \psi = v_N \cdot h - \frac{1}{A} \int_X v_N \cdot h \, \mathrm{d}\mu_F + \omega \left(\frac{A}{A_X G} - 1 \right). \tag{7}$$

If we prescribe the value of ψ in one point of X, (7) has a unique solution.

An area-uniform mesh might not always be the best representation of a surface. In some applications, it makes sense, for example, to concentrate the mesh points in the regions of higher curvature. If we consider, for example, the mean curvature H, this can be achieved if we require a constant value of $Gf(H)$, where f is a positive increasing function. Similarly as in the previous case, (4) must hold and thus we get the condition

$$\frac{G}{A} \underset{t\to\infty}{\longrightarrow} \frac{\frac{1}{f(H)}}{\int_X \frac{1}{f(H)} \, \mathrm{d}\mu_X}.$$

This leads to the condition for ψ,

$$\Delta_{g_F} \psi = v_N \cdot h - \frac{1}{A} \int_X v_N \cdot h \, \mathrm{d}\mu_F + \omega \left(\frac{A}{G} \frac{\frac{1}{f(H)}}{\int_X \frac{1}{f(H)} \, \mathrm{d}\mu_X} - 1 \right). \tag{8}$$

3 Discretization of the Mathematical Model

Since the problem of point cloud reconstruction is in principle analogous to the 3D image segmentation problem, we refer to our previous works concerning this topic [5,6] and we only provide a brief explanation of the numerical method.

Before we can apply our model, we need to compute the distance function d. For this purpose, we consider Ω to be a box and we discretize it by constructing a voxel mesh where each voxel is a cube of side length h. Then we approximate the value of d in each voxel. This approach is necessary since a brute-force computation of the distance function is practically inapplicable in most cases. Having an approximation of d, we approximate ∇d simply by central differences. To identify the voxel corresponding to a point $F(P,t)$, $P \in X$, we round its coordinates divided by h.

We consider a uniform discretization of the time interval and we use the notation $F^n = F^{t_n}$ and $S^n = S^{t_n}$. The time discretization of (1) is semi-implicit,

$$\frac{F^n - F^{n-1}}{\tau} = w_a \left(\nabla d \cdot N^{n-1}\right) N^{n-1} + w_c d \Delta_{g_{F^{n-1}}} F^n + v_T^{n-1}. \tag{9}$$

Now, we consider a triangular structure on X consisting of vertices X_i, $i = 1 \dots n_v$, edges e_j, $j = 1 \dots n_e$, and triangles $\mathcal{T}_k$, $k = 1 \dots n_t$. We construct a piecewise linear approximation of F^n denoted by $\bar{F}^n$ – we set $\bar{F}^n(X_i) = F^n(X_i)$ and then, for any triangle $\mathcal{T}_p$ with vertices X_i, X_{i_p}, $X_{i_{p+1}}$, we set $\bar{F}^n(\lambda_1 X_i + \lambda_2 X_{i_p} + \lambda_3 X_{i_{p+1}}) = \lambda_1 F^n(X_i) + \lambda_2 F^n(X_{i_p}) + \lambda_3 F^n(X_{i_{p+1}})$. The embedding $\bar{F}^n$ induces a metric g^n on X which induces a measure μ^n on X. The approximation of the unit normal to $S^n = \bar{F}^n(X)$ at F_i^n is denoted by N_i^n. The numerical scheme that we apply uses the angles of $\mathcal{T}_p$ adjacent to X_{i_p} and $X_{i_{p+1}}$ measured in the metric g^n. We denote them by $\theta_{p,1}^n$ and $\theta_{p,2}^n$.

The space discretization of (9) is done by the finite volume approach. The control volume mesh is constructed by barycentric subdivision of the triangles $\mathcal{T}_k$ (Figure 2). We will denote by $\nu_{p,1}^n$, $\nu_{p,2}^n$ the outward unit normals to the control volume edges $\bar{F}^n(\sigma_{p,1})$ and $\bar{F}^n(\sigma_{p,2})$ in the plane of $\bar{\mathcal{T}}_p^n$. The principle of the method is to integrate (9) over a control volume V_i,

$$\begin{aligned}\int_{V_i} \frac{F^n - F^{n-1}}{\tau} \,\mathrm{d}\mu_{F^{n-1}} = &\int_{V_i} w_a \left(\nabla d \cdot N^{n-1}\right) N^{n-1} \,\mathrm{d}\mu_{F^{n-1}} \\ &+ \int_{V_i} w_c d \Delta_{g_{F^{n-1}}} F^n \,\mathrm{d}\mu_{F^{n-1}} + \int_{V_i} v_T^{n-1} \,\mathrm{d}\mu_{F^{n-1}},\end{aligned} \tag{10}$$

and then to approximate the integrals that we obtain.

For the first term of the right hand side, we need to approximate the surface normal at F_i^n. We take the arithmetic mean of the normals to all triangles containing F_i^n. For the second term, we use the cotangent scheme [4],

$$\int_{V_i} w_c d \Delta_{g_{F^{n-1}}} F^n \,\mathrm{d}\mu_{F^{n-1}} \approx w_c d_i \frac{1}{2} \sum_{p=1}^{m} \left(\cot \theta_{i,p-1,1}^{n-1} + \cot \theta_{i,p,2}^{n-1}\right) (F_i^n - F_{i_p}^n), \tag{11}$$

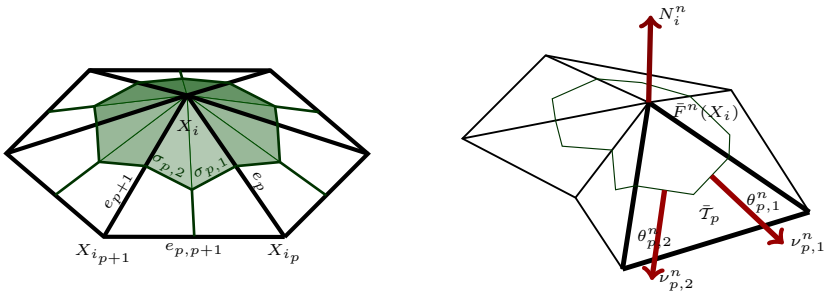

Fig. 2. The surface discretization mesh. Left, the triangulation of the abstract surface X. Right, the corresponding approximation of the embedded surface $F^n(X)$.

where m is the number of vertices connected to X_i by an edge and d_i is the value of d in the voxel corresponding to F_i^n. Dividing the right hand side by $\mu^{n-1}(V_i)$, we obtain an approximation of h^{n-1}.

To discretize the integral of the tangential velocity, we recall that w_T^n is a gradient field and we apply the following version of the Stokes theorem [2]

$$\int_{V_i} v_T^{n-1} \,\mathrm{d}\mu_{F^{n-1}} = \int_{\partial V_i} \psi^{n-1} \nu_i^{n-1} \,\mathrm{d}H_{\mu_{F^{n-1}}} - \int_{V_i} \psi^{n-1} h^{n-1} \,\mathrm{d}\mu_{F^{n-1}}.$$

This gives

$$\begin{aligned} \int_{V_i} v_T^{n-1} \,\mathrm{d}\mu_{F^{n-1}} \approx \sum_{p=1}^{m} \big(\|\sigma_{i,p,1}\|_{n-1} \psi_{i,p,1}^{n-1} \nu_{i,p,1}^{n-1} + \|\sigma_{i,p,2}\|_{n-1} \psi_{i,p,2}^{n-1} \nu_{i,p,2}^{n-1} \big) \\ -\mu^{n-1}(V_i) \psi_i^{n-1} h_i^{n-1} \end{aligned} \tag{12}$$

where $\|\cdot\|_{n-1}$ denotes the length computed by the metric g^{n-1} and $\psi_{i,p,1}^{n-1}$, $\psi_{i,p,2}^{n-1}$ are the values of ψ^{n-1} in the midpoints of $\sigma_{i,p,1}$ and $\sigma_{i,p,2}$. We obtain these values by linear interpolation.

The function ψ is computed from (7) where, again, we use the cotangent scheme to discretize the Laplace-Beltrami operator of ψ^{n-1}. The volume density is approximated by

$$G_i^{n-1} = \mu^{n-1}(V_i) \frac{n_v}{A_X}. \tag{13}$$

4 Experiments and Results

In the first phase of testing, we used the surface and the uniform point cloud shown in Figure 1. The cloud contains 2562 points. The initial condition was a triangulated sphere with 2562 vertices. The distance function was computed in a box volume of $200 \times 200 \times 200$ voxels. In all experiments presented in this paper, it was approximated by the fast sweeping method [8]. The parameters of the model were set to $\tau = 0.002$, $w_a = 300.0$, $w_c = 100.0$. We used the area-uniform redistribution with the redistribution speed $\omega = 100.0$. The evolution

was stopped after 400 time steps. Since d is a distance function smoothed by a heat kernel, it is nowhere zero and thus there is no stopping time implied by the model. However, we can stop the evolution if the difference between S^{n-1} and S^n is lower than some threshold. Figure 3 shows the initial condition as well as a few stages of the evolution. Since a point cloud is usually a rough approximation of an object, the resulting representative surface might benefit from some additional smoothing after the evolution stops. In our case, this can be done by using the same algorithm; we only need to set $d = 1$ in all voxels and we obtain a mean curvature flow model. The last two pictures in Figure 3 show the result of such smoothing. Here, we set $w_c = 300.$, $\omega = 2000$, $\tau = 0.0005$ and we performed only 20 time steps to prevent excessive shrinking of our surface.

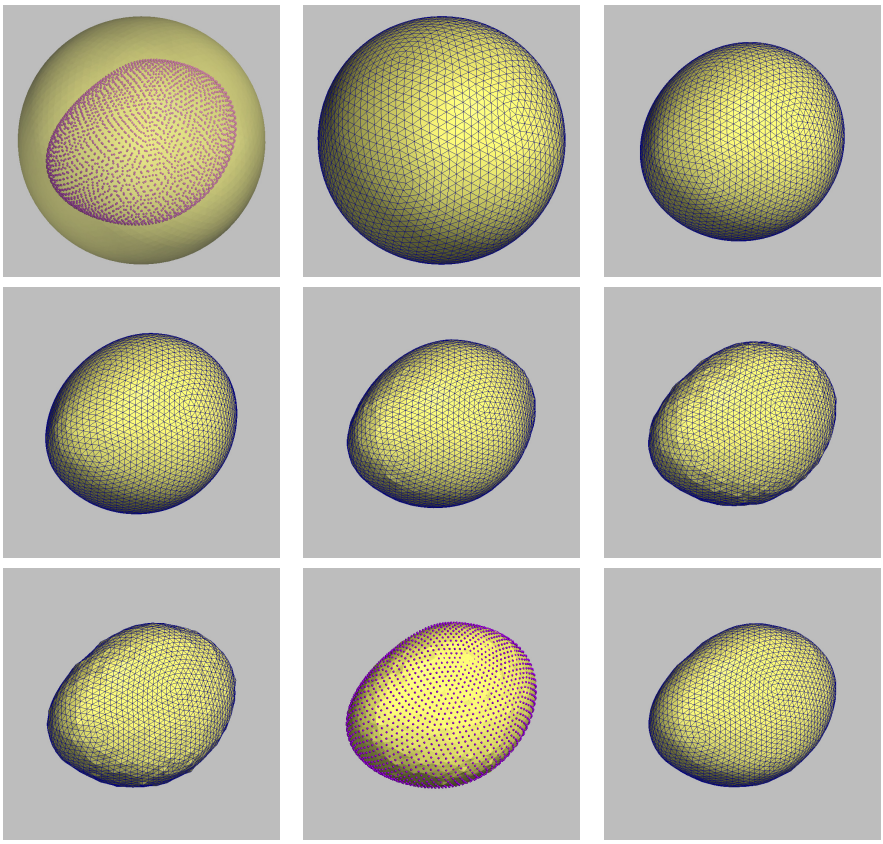

Fig. 3. Reconstruction of a surface from a uniform point cloud representation. We can see the initial surface together with the point cloud and the triangulation of this surface. What follows is the evolved surface after 100, 150, 200 and 250 time steps, the surface at the end of the evolution displayed with its triangulation and with the point cloud. Finally, we show the smoothed surface after applying 20 steps of mean curvature flow.

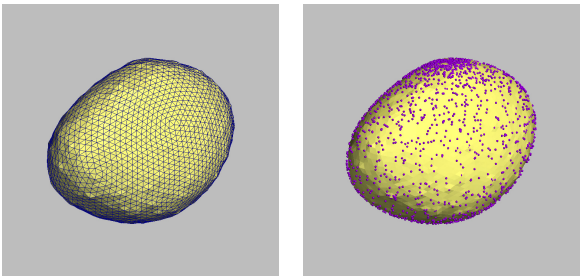

Fig. 4. Reconstruction of a surface from a non-uniform point cloud representation. We show the results obtained by 400 time steps of the evolution.

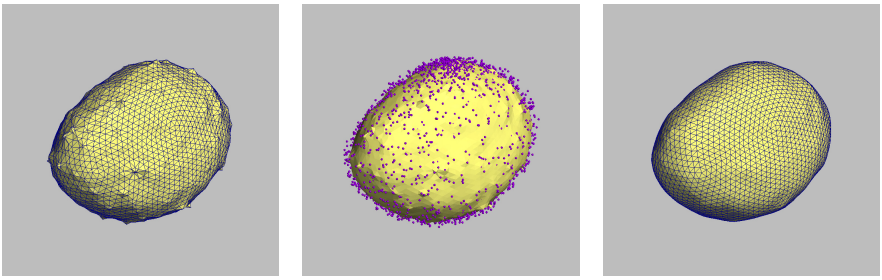

Fig. 5. Reconstruction of a surface from a noised point cloud representation. We show the results obtained by 400 time steps of the evolution and then after 30 additional steps of mean curvature flow.

The following experiments test our method on potentially more problematic point clouds (Figure 1). The number of points, the initial condition and model parameters were the same as in the case of the uniform point cloud. The results are displayed in Figures 4–7. The non-uniform point cloud did not yield any special issues and lead to a good surface representation (Figure 4). We then used it to construct the other point clouds. As for the noisy point cloud, we shifted each point P_i in the direction of $P_i - (0,0,0)$ by $r\|P_i\|$, where r is a random real number from $\langle -0.05, 0.05\rangle$. To construct the point cloud with outlying points, we shifted 10 randomly selected points by $0.3\|P_i\|$. Finally, by deleting all points lying in a selected region, we obtained the last point cloud for our tests.

The result that we obtained by using the noised point cloud was, as expected, quite bumpy. In this case, we applied 30 steps of the mean curvature flow to get a more acceptable surface representation. On the contrary, the point cloud with several outlying points did not cause any major problems. Using the regularized distance function allows the evolving surface to run through an isolated point due to the non-vanishing curvature term. As we can see in Figure 6, the outliers are noticed by the evolving surface but afterwards, they are overcome. Finally, the ability of our method to patch empty regions has been already explained in Section 2. If some reasonably large part of the data is missing, the model will create a planar patch as a representation of this region (Figure 7).

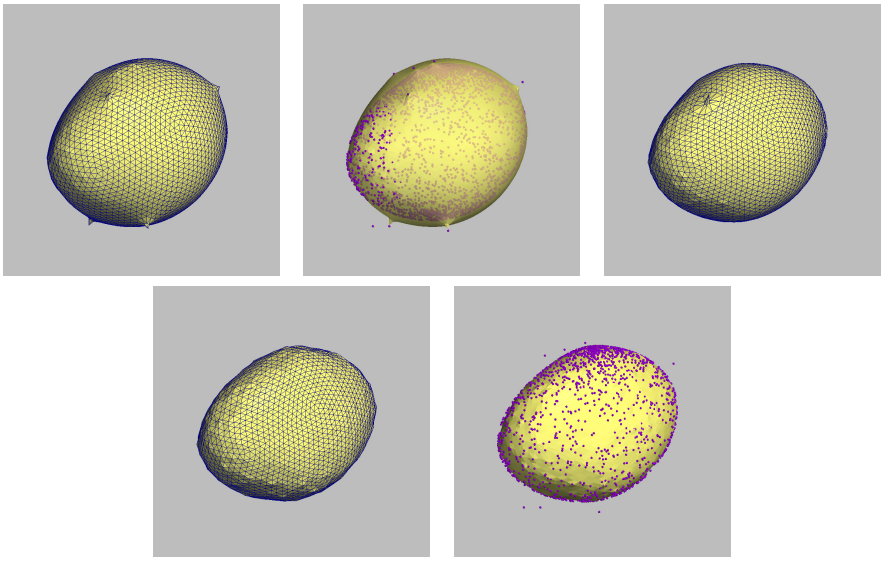

Fig. 6. Reconstruction of a surface from a point cloud representation with outlying points. We show the evolved surface after 180 time steps (first its triangulation and then displayed together with the corresponding point cloud), after 220 and 400 time steps of the evolution.

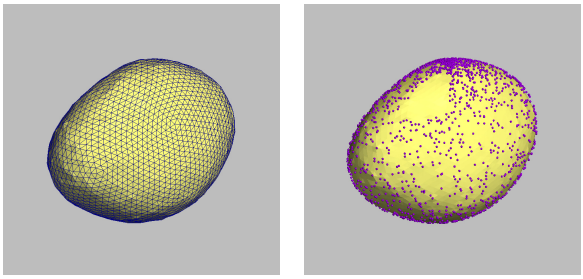

Fig. 7. Reconstruction of a surface from a point cloud representation with a missing part. We show the results obtained by 400 time steps of the evolution.

In the next example, we took a point cloud representing a much more complicated object (Figure 8). In this case, we computed the distance function on a finer grid of $300 \times 300 \times 300$ voxels. The initial surface was also more finely discretized, it was a sphere with 16386 vertices. The values of the model parameters were $\tau = 0.002$, $w_a = 300.$, $w_c = 30.$, $\omega = 100.$. We show the result obtained after 900 time steps and then after a slight smoothing by 5 steps of the mean curvature flow with parameter values mentioned above.

Finally, we present an experiment illustrating the effect of the curvature driven tangential redistribution. We use the uniform point cloud shown in Figure 3. The curvature driven redistribution was not used during the whole evolution

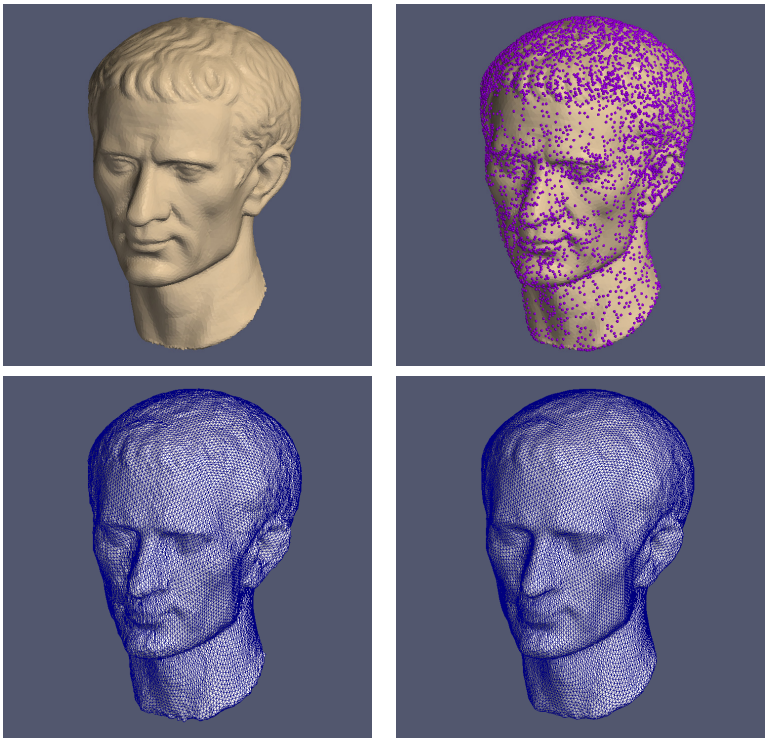

Fig. 8. Reconstruction of a more complicated surface (surface taken from http://segeval.cs.princeton.edu/) . We show the original surface and the point cloud representing it. In the second row, we can see our reconstruction obtained after 900 time steps of evolution and the surface smoothed by 5 additional steps of mean curvature flow.

process, but we rather used the area-uniform redistribution to optimally capture the external vector field. After 400 time steps and additional 20 steps of smoothing, we obtained the surface representation shown in the last two pictures of Figure 3. This was the starting point for the curvature driven redistribution. We kept evolving the surface by mean curvature flow but it was now by orders of magnitude slower; we set $w_c = 1.0$. The redistribution speed changed to $\omega = 200.0$ and we performed 15 steps of the evolution. The function f used to redistribute the points was $f(H) = e^{20H}$.

The result is shown in Figure 9. As we can see, the points are more densely distributed in the regions with higher mean curvature (compare to the resulting mesh shown in Figure 3). To provide an evaluation of the redistribution other than a visual inspection of the resulting mesh, we add two graphs. Here, we consider the ratio $r_{A,i} = \frac{\mu^n(V_i)}{A_V}$, where A_V is the average control volume area, $A_V = A/n_v$. The graphs show the dependence of $r_{A,i}$ on H – each point of the plot represents one pair $(H_i, r_{A,i})$. We can see that before the curvature driven

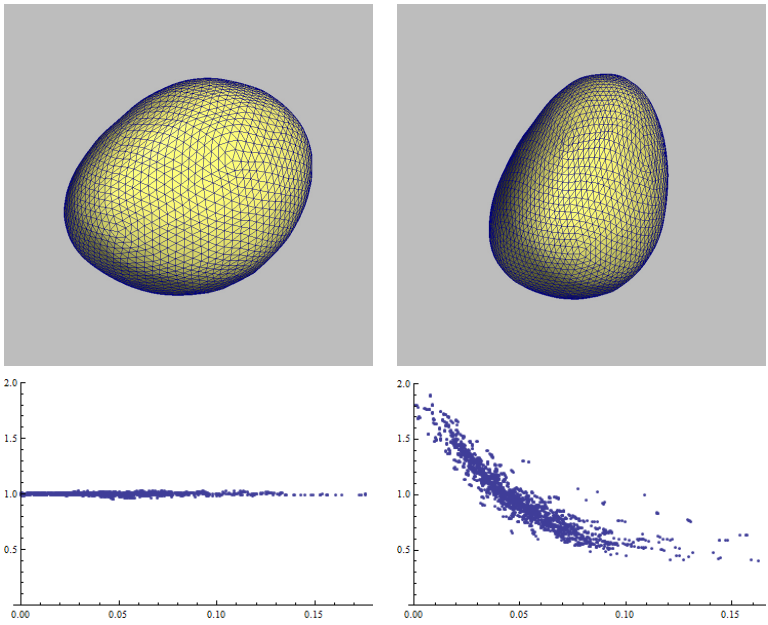

Fig. 9. A test example of a representation obtained with the help of curvature driven tangential redistribution. We show the resulting surface representation obtained after 15 time steps of the redistribution in two different views. Note that the density of discretization points is higher in the regions with higher mean curvature. The graphs show the dependence of $r_{A,i}$ on H before and after the curvature driven redistribution.

redistribution, $r_{A,i}$ is close to 1 in all vertices. After 15 steps of the redistribution, we can observe that $r_{A,i}$ is clearly decreasing with increasing H_i.

Acknowledgments. This work was supported by the grants APVV-0072-11 and APVV-0161-12.

References

1. Berger, M., Tagliasacchi, A., Seversky, L. M.: State of the Art in Surface Reconstruction from Point Clouds. In: Eurographics 2014 - State of the Art Reports, pp. 161–185. The Eurographics Association (2014)
2. Dziuk, G., Elliott, C.M.: Finite elements on evolving surfaces. IMA J. Numer. Anal. **27**, 262–292 (2007)
3. Mantegazza, C.: Lecture Notes on Mean Curvature Flow. Springer (2011)
4. Meyer, M., Desbrun, M., Schroeder, P., Barr, A.H.: Discrete differential geometry operators for triangulated 2manifolds. Visualization and Mathematics III, 35–57 (2003)
5. Mikula, K., Remešíková, M.: 3D Lagrangian Segmentation with Simultaneous Mesh Adjustment. In: Finite Volumes for Complex Applications VII. Springer Proceedings in Mathematics and Statistics, vol. 77, pp. 685–694 (2014)

6. Mikula, K., Remešíková, M., Sarkoci, P., Ševčovič, D.: Surface evolution with tangential redistribution of points. SIAM J. Sci. Comp. **36**(4), 1384–1414 (2014)
7. Surynková, P.: Curve and Surface Reconstruction Based on the Method of Evolution. In: WDS 2010 Proceedings, Part I, MATFYZPRESS, pp. 139–144 (2010)
8. Zhao, H.: A Fast Sweeping Method for Eikonal Equations. Mathematics of Computation **74**(250), 603–627 (2004)
9. Zhao, H.K., Osher, S., Fedkiw, R.: Fast Surface Reconstruction Using the Level Set Method. In: Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision 2001, pp. 194–201. IEEE (2001)