Proceedings of EQUADIFF 2017
pp. 387–396

NEW EFFICIENT NUMERICAL METHOD FOR 3D POINT CLOUD SURFACE RECONSTRUCTION BY USING LEVEL SET METHODS.

BALÁZS KÓSA*, JANA HALIČKOVÁ–BREHOVSKÁ†, AND KAROL MIKULA‡

Abstract. In this article, we present a mathematical model and numerical method for surface reconstruction from 3D point cloud data, using the level-set method. The presented method solves surface reconstruction by the computation of the distance function to the shape, represented by the point cloud, using the so called Fast Sweeping Method, and the solution of advection equation with curvature term, which creates the evolution of an initial condition to the final state. A crucial point for efficiency is a construction of initial condition by a simple tagging algorithm which allows us also to highly speed up the numerical scheme when solving PDEs. For the numerical discretization of the model we suggested an unconditionally stable method, in which the semi-implicit co-volume scheme is used in curvature part and implicit upwind scheme in advective part. The method was tested on representative examples and applied to real data representing the historical and cultural objects scanned by 3D laser scanners.

Key words. point cloud, level set methods, reconstruction

AMS subject classifications. 65M06, 65Y20, 53A05, 65D17

1. Introduction. The aim of our work is to create a reliable and efficient numerical method which can easily create computerized 3D models from point cloud data that resembles the original object as much as possible. This type of data can be obtained by 3D scanning or by photogrammetric methods. The data created in this manner contains three coordinates for every scanned point. For further processing and the creation of an exact digital model of the scanned object this information is not enough. The point cloud lacks the information of the connectivity between the points, thus making the reconstruction of the surface a difficult task. Papers as [1, 2] have shown us that for solving this problem the level-set method can be applied. We follow basic ideas from these papers, but we take a different approach in the solution of the partial differential equation presented here.

In the following parts of our paper after the Mathematical Formulation of the applied level set equation in the section Algorithm for point cloud surface reconstruction we will present our method and its numerical discretization and solution. After the theoretical deduction of the method and the description of a short algorithm for computing the initial condition in Computation acceleration we suggest a way to accelerate the computational time, making the algorithm really efficient. In the last section Numerical results we present created 3D models which we obtained so far. We achieved this by implementing our method in the language C with the use of the programming environment of Visual Studio. The example pictures of the results used in this article are direct outputs from our application processed in the freely available open-source visualization software Paraview. With the help of this software we

*Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering, Slovak University of Technology, Radlinskeho 11, 810 05 Bratislava, Slovakia (kosa@math.sk).

† Monument Board of the Slovak Republic Cesta na Červený most 6, 81406 Bratislava, Slovakia (jana.halickova@gmail.com)

‡Department of Mathematics and Descriptive Geometry, Faculty of Civil Engineering, Slovak University of Technology, Radlinskeho 11, 810 05 Bratislava, Slovakia (mikula@math.sk).

can easily compare the initial point cloud data and our results, to confirm that our assumptions regarding this new numerical method are right.

2. Algorithm for point cloud surface reconstruction. The level set method, which we are using is based on the solution of the advection equation with curvature term

$$\begin{gathered} u_t - \nabla d \cdot \nabla u - \delta \left|\nabla u\right| \nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) = 0 \\ (x,t) \in \Omega \times [0,T] \end{gathered} \tag{2.1}$$

where $u(x,t)$ is an unknown function, $v = -\nabla d$ is the advective velocity defined by the gradient of distance function d to the point cloud, parameter $\delta > 0$ determines influence of the curvature to the result, Ω is the computational domain and $[0,T]$ is a time interval. This equation is coupled with homogeneous Neumann boundary conditions and an initial condition which we will discuss later.

To obtain numerical solution of the model created from point cloud data, denoted by $\Omega_0 \subset \Omega$ and determined by equation (2.1), following steps have to be executed. First we have to computate the distance function to the point cloud. For computation we use the Fast sweeping method, as introduced in [3].The initialization of distance function in the Fast sweeping method is done in such way, that we prescribe exact distance to the nearest point from the cloud in the grid points next to the points in the cloud. After that we have to find a subvolume containing Ω_0, which will be used to set the initial function u^0 for the generation of the final solution of the equation. This subvolume is defined on discrete grid in subsection 2.2. The final solution (created 3D model) will be represented by an isosurface of the computated function $u\,(x,T)$ with value 0.5.

2.1. Numerical scheme for solving advection equation with curvature term. The numerical scheme is obtained by discretization of equation (2.1). We will do this analogically to the discretization used in [4].

2.1.1. Time discretization. For time discretization, we have to choose a uniform discrete time step, denoted by τ. We can replace the time derivative in (2.1) with a backward difference. Then we can formulate our semi-implicit time discretization in the following way:

Let τ be a fixed number and u^0 a function representing the initial surface of our mathematical model. Then at every discrete time $t_n = n\tau,\ n = 1, ..., N$ we search for the function u^n as the solution to equation

$$\frac{u^n - u^{n-1}}{\tau} - \nabla d \cdot \nabla u^n - \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|}\right) = 0 \tag{2.2}$$

2.1.2. Spatial discretization. Our discretized model consists of a 3D grid, which is built of voxels with cubic shape and an edge size h. We will interpret spatial discretization of the level set function u as numerical values $u_{i,j,k}$ at the voxel centres. In order to easily computate the gradient of the level-set equation $\left|\nabla u^{n-1}\right|$ in every time step of (2.2) we induct a 3D tetrahedral grid into the voxel structure and take a piecewise linear approximation of $u\,(x)$ on such a grid. This way we obtain a constant value of the gradient for each tetrahedron, by which we can construct in a simple and clear way the fully discrete system of equations.

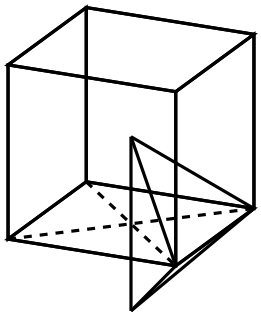

Fig. 2.1: Our initial voxel grid cell with a tetrahedral grid cell

The 3D tetrahedral finite element grid is created by the following approach. Every voxel is divided into six pyramid shaped elements with base surface given by the voxel's walls and vertex by the voxel centre. Each one of these pyramids is joined with neighbouring pyramids with whom they have a common base surface. These newly formed octahedrons are then split into four tetrahedrons as seen in Figure 2.1. In our new grid $\mathcal{T}_h$ the level-set function will be updated only at the centres of the voxels. They will represent so called degree of freedom (DF) nodes.

For the tetrahedral grid we construct a co-volume mesh, which will consist of cells p associated only with DF nodes of $\mathcal{T}_h$. We denote all neighbouring cells q of p by C_p. The cells q are all connected to the cell p by a common edge of four tetrahedrons, which is denoted by σ_{pq} with length h_{pq}. Each cell p is bounded by a plane for every $q \in C_p$ which is perpendicular to σ_{pq} and is denoted by e_{pq}. The set of tetrahedrons which have σ_{pq} as an edge are denoted by ε_{pq}. For every $T \in \varepsilon_{pq}$, c_{pq}^T is the area of the intersection of e_{pq} and T. N_p will be a set of tetrahedrons that have DF node associated with cell p as a vertex. On this grid u_h will be a piecewise linear function. Then we can use the notation $u_p = u_h(x_p)$, where x_p denotes the center coordinates of cell p.

Now that we have all notations which are needed we can begin the derivation of the spatial discretization of (2.2). We will do this by using a following modified form of the equation:

$$\frac{u^n - u^{n-1}}{\tau} + v \cdot \nabla u^n = \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|}\right) \tag{2.3}$$

where $v = -\nabla d$.

As the first step we will integrate (2.3) over every cell p.

$$\int_p \frac{u^n - u^{n-1}}{\tau} dx + \int_p v \cdot \nabla u^n dx = \int_p \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{|\nabla u^{n-1}|}\right) dx \tag{2.4}$$

For the first term on the left-hand side of (2.4) we get the approximation

$$\int_p \frac{u^n - u^{n-1}}{\tau} dx = m(p) \frac{u_p^n - u_p^{n-1}}{\tau} \tag{2.5}$$

where $m(p)$ is a measure in $\mathbb{R}^d$ of the cell p.

For the second term on the left-hand side of (2.4) we are using the implicit upwind approach and get

$$\int_p v \cdot \nabla u^n dx = \sum_{q \in C_p} min\left(v_{pq}, 0\right)\left(u_q^n - u_p^n\right) \tag{2.6}$$

where $v_{pq} = h_{pq}^2 v \cdot n$.

Now what remains is the discretization of the right-hand side of (2.4). We use the divergence theorem to get

$$\int_p \delta \left|\nabla u^{n-1}\right| \nabla \cdot \left(\frac{\nabla u^n}{\left|\nabla u^{n-1}\right|}\right) dx = \delta \left|\nabla u_p^{n-1}\right| \sum_{q \in C_p} \int_{e_{pq}} \frac{1}{\left|\nabla u^{n-1}\right|} \frac{\partial u^n}{\partial n} d\sigma \tag{2.7}$$

The integral part $\int_{e_{pq}} \frac{1}{\left|\nabla u^{n-1}\right|} \frac{\partial u^n}{\partial n} d\sigma$ and $\left|\nabla u_p^{n-1}\right|$ from (2.7) will be approximated numerically using piecewise linear reconstruction of u^{n-1} on the tetrahedral grid $\mathcal{T}_h$, thus we get

$$\delta \left|\nabla u_p^{n-1}\right| \sum_{q \in C_p} \left(\sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left|\nabla u_T^{n-1}\right|} \right) \frac{u_q^n - u_p^n}{h_{pq}}$$

$$M_p^{n-1} = \left|\nabla u_p^{n-1}\right| = \sum_{T \in N_p} \frac{m\left(T \cap p\right)}{m\left(p\right)} \left|\nabla u_T^{n-1}\right|$$

and the final form of equation (2.3) after reorganization will be

$$\begin{aligned} u_p^{n-1} = u_p^n + \frac{\tau}{m\left(p\right)} & \left(\sum_{q \in C_p} min\left(v_{pq}, 0\right)\left(u_q^n - u_p^n\right) \right. \\ & \left. -\delta\, M_p^{n-1} \sum_{q \in C_p} \left(\sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left|\nabla u_T^{n-1}\right|} \right) \frac{u_q^n - u_p^n}{h_{pq}} \right) \end{aligned} \tag{2.8}$$

From this form, we are able to derive the system of linear equations which we will solve at every time step. For the linear equations, we will define regularized gradients by

$$\left|\nabla u_T\right|_\varepsilon = \sqrt{\varepsilon^2 + \left|\nabla u_T\right|^2} \tag{2.9}$$

After we arrange all parts of equation (2.8) we get the following coefficients

$$a_{pq}^{n-1} = \frac{\tau}{m\left(p\right)} \left(min\left(v_{pq}, 0\right) - \delta\, M_p^{n-1} \frac{1}{h_{pq}} \sum_{T \in \varepsilon_{pq}} c_{pq}^T \frac{1}{\left|\nabla u_T^{n-1}\right|_\varepsilon} \right) \tag{2.10}$$

thus, we can formulate our semi-implicit co-volume scheme:

Let $u_p^0,\ p = 1, ..., M$ be given discrete initial values of the level-set function. Then, for $n = 1, ..., N$ we look for $u_p^n,\ p = 1, ..., M$, satisfying

$$u_p^n + \frac{\tau}{m\left(p\right)} \sum_{q \in N_p} a_{pq}^{n-1}\left(u_q^n - u_p^n\right) = u_p^{n-1} \tag{2.11}$$

With addition of homogeneous Neumann boundary conditions to our fully discrete scheme we obtain a system of linear equations. Since a_{pq}^{n-1} are non-negative we can prove the following statement.

Theorem. *There exists unique solution* $(u_1^n, ..., u_M^n)$ *of* (2.11) *for any* $\tau > 0$, $\varepsilon > 0$, *and for every* $n = 1, ..., N$. *The system matrix is a strictly diagonally dominant M-matrix. For any* $\tau > 0$, $\varepsilon > 0$, *the following* L_∞ *stability holds:*

$$\min_p \ u_p^0 \le \min_p \ u_p^n \le \max_p \ u_p^n \le \max_p \ u_p^0, \ \ 1 \le n \le N. \tag{2.12}$$

The number of time steps N is determined by the difference of the solution in current and previous time steps in discrete L^2 norm. The computation is stopped if this difference is less than the prescribed tolerance, which we usually set to 10^{-6}. Then the stopping time $T = N\tau$.

If we denote the DF nodes with indexes (i, j, k) and rearrange (2.11) to obtain the coefficients for every node we can define for a DF node the equation

$$\begin{aligned} &c_{i,j,k}u_{i,j,k}^n + b_{i,j,k}u_{i,j,k-1}^n + t_{i,j,k}u_{i,j,k+1}^n + n_{i,j,k}u_{i+1,j,k}^n \\ &+s_{i,j,k}u_{i-1,j,k}^n + e_{i,j,k}u_{i,j+1,k}^n + w_{i,j,k}u_{i,j-1,k}^n = u_{i,j,k}^{n-1} \end{aligned} \tag{2.13}$$

When we collect the equations for all DF nodes and take into account Neumann boundary conditions we get the linear system which we have to solve. For the solution of this system we choose the SOR (Successive Over Relaxation) iterative method. We start the iterations by setting $u_{i,j,k}^n = u_{i,j,k}^{n-1}$, then in every iteration $l = 1, ...$ we use the following two step procedure:

$$\begin{aligned} Y = (&u_{i,j,k}^{n(0)} - b_{i,j,k}u_{i,j,k-1}^{n(l)} - t_{i,j,k}u_{i,j,k+1}^{n(l-1)} - n_{i,j,k}u_{i+1,j,k}^{n(l-1)} \\ &- s_{i,j,k}u_{i-1,j,k}^{n(l)} - e_{i,j,k}u_{i,j+1,k}^{n(l-1)} - w_{i,j,k}u_{i,j-1,k}^{n(l)})/c_{i,j,k} \\ u_{i,j,k}^{n(l)} &= u_{i,j,k}^{n(l-1)} + \omega\left(Y - u_{i,j,k}^{n(l-1)}\right) \end{aligned} \tag{2.14}$$

We define squared L_2 norm of residuum at current iteration by

$$\begin{aligned} R_l = \sum_{i,j,k} (&c_{i,j,k}u_{i,j,k}^{n(l)} + b_{i,j,k}u_{i,j,k-1}^{n(l)} + t_{i,j,k}u_{i,j,k+1}^{n(l)} + n_{i,j,k}u_{i+1,j,k}^{n(l)} \\ &+ s_{i,j,k}u_{i-1,j,k}^{n(l)} + e_{i,j,k}u_{i,j+1,k}^{n(l)} + w_{i,j,k}u_{i,j-1,k}^{n(l)} - u_{i,j,k}^{n(0)})^2 \end{aligned}$$

The iterative process is stopped if $R^l < TOL$.

2.2. Computation of the initial condition. As mentioned, this method needs an initial condition, represented by the initial function $u^0(x)$, which will be deformed to get the solution, that is the final form of the created 3D model. Theoretically any initial surface that contains the point cloud data set could be used, but an optimal initial guess is crucial for the efficiency of the method. We can find this optimal surface by identifying all points for which the value of the distance function is greater or equal to a parameter β. For simplicity let us call these points, exterior points. To find all these points we will use the following algorithm:

- Mark all points on the borders of the grid as exterior and add them to set E.
- For every point in the set E check all neighbouring points in the grid.

- If the neighbouring point is not an exterior point and its distance from the point cloud is greater or equal to β add it to the set E and mark as exterior.
- Continue until you get to the last point of E.

When we found all the exterior points we set $u^0(x)$ to be equal 0 in every exterior point and 1 in every other point. With this approach, we can find an initial surface close to the final shape as seen on the Figure 2.2.

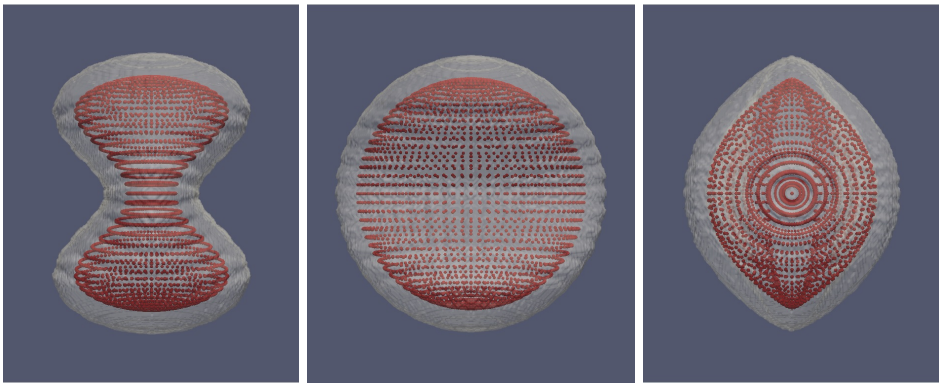

Fig. 2.2: Example for the initial condition used in our method. Object is shown from different angles.

3. Computation acceleration. The part of our algorithm which consumes the most time during computation is the solution of the linear system of equations (2.11) coupled with the computation of its coefficients. To reduce this time, we came up with the following idea. First, we construct a band around the area between the initial surface and the point cloud data. To find the surface which we want to reconstruct it is sufficient to update the values on grid cells contained in such a band, thus we can computate coefficients and evaluate the SOR method (2.14) only in this new subset of all grid cells. On Figure 3.1 we can see an example of this subset. For easier visualization, we show this on a slice with the plane $x = 0$. Here the red line marks the point cloud data, the purple line the initial surface and the white lines the borders of the created band. In the background of the picture we show the values of the distance function.

To find this area we adopted the algorithm mentioned in the previous section, which was used to find the initial surface, to this task. To obtain an outer border for the band which contains the initial surface we chose a new parameter $\gamma = 2\beta$. With this additional parameter and the introduction of a new set denoted F the algorithm for finding the band is given as follows.

- Tag all points on the borders of the grid and add them to the set E.
- For every point in the set E check all neighbouring points in the grid.
- If the neighbouring point is not tagged execute the following steps.
  - If the neighbouring point's distance from the point cloud is smaller or equal to γ add it to the set F.
  - If the neighbouring point's distance from the point cloud is greater or equal to β add it to the set E as well and tag it.
- Continue until you get to the last point of E. When we finish with set E we start a new cycle for set F.

- For every point in the set F check all neighbouring points in the grid.
- If the neighbouring point is not tagged and its distance from the point cloud is smaller or equal to γ add it to the set F.
- Continue until you get to the last point of F.

While we look for points with distance smaller or equal to γ we will cross the border with distance 0, represented by the point cloud, thus set F will contain also grid points from the inner region of the object. From the set F we can create an array consisting of values 0, for points not in the band, and 1, for points in the band. This will serve as a mask for the SOR method, thus in the computation loops we can determine if it is necessary to computate the new value or if we can skip to the next grid point.

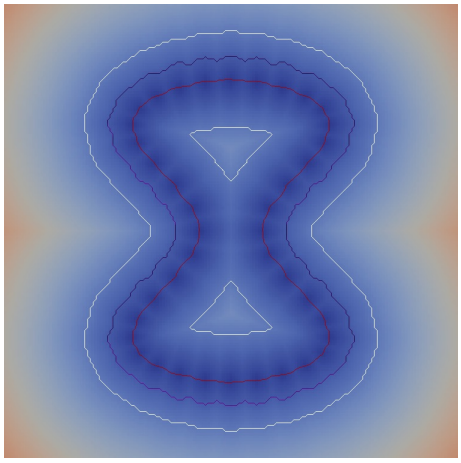

Fig. 3.1: The slice of our new computational area on the plane $x = 0$.

We measured how much time we managed to save with this new approach on real-life data sets representing a bracelet and a sealer. The tests were executed on a personal notebook with a dual core processor and 4 GB of memory. Our results are listed in the tables 3.1 and 3.2. We tested the algorithm on grids containing 40^3, 80^3 and 160^3 grid cells. All tests were performed with the same parameter β and stopping criteria for the iterations.

In the second column of the tables we recorded the number of points contained by the band. This number depends on the size and form of the original object represented by the data set. In columns three and four we see the measured times for the original and optimized implementation. In the tests, we achieved not only reduced times but also better convergence, so fewer time steps were needed. This led to computations which were 20 to 60 times faster.

Visually we cannot detect any difference between the created 3D models computed by the two methods, original and optimized. We measured the mean value of squared differences between all grid values and listed the obtained values in the third column. We can see that these values are in the tolerable range.

Number of grid cells	Points in band	CPU time (s) Original	CPU time (s) Optimized	Mean squared difference
40^3	4 636	4.269	0.261	8.90795e-7
80^3	37 640	34.247	1.677	2.27554e-8
160^3	304 456	895.68	13.385	1.92055e-8

Table 3.1: CPU times comparison for the bracelet data set

Number of grid cells	Points in band	CPU time (s) Original	CPU time (s) Optimized	Mean squared difference
40^3	6 075	13.914	0.537	1.75849e-6
80^3	48 710	88.673	3.470	4.38982e-8
160^3	392 185	2 051.402	72.846	9.36878e-9

Table 3.2: CPU times comparison for the sealer data set

4. Numerical results. In this section, we present the reconstruction of the point cloud surfaces on a representative testing example and real data. These examples are a good display of the quality of our method.

Figure 4.1 illustrates the test example. This object was used for the verification of the correct behaviour of our method during the implementation phase. The point cloud data was generated with corresponding parametric equations of the object. The representative example was created on a grid containing 80^3 cells. We can see that for this test with such a sparse grid we already got good results.

On Figure 4.2 and 4.3 we can see real-life data. These items where archaeological finds and the point cloud scans were provided by the Monuments Board of the Slovak republic to which we express our great thanks. On Figure 4.2 we can see a bracelet. The created 3D model was computed on a grid with 160^3 cells. On Figure 4.3 we can see a sealer, with a very interesting surface structure. The created 3D model was computed on a grid with 320^3 cells.

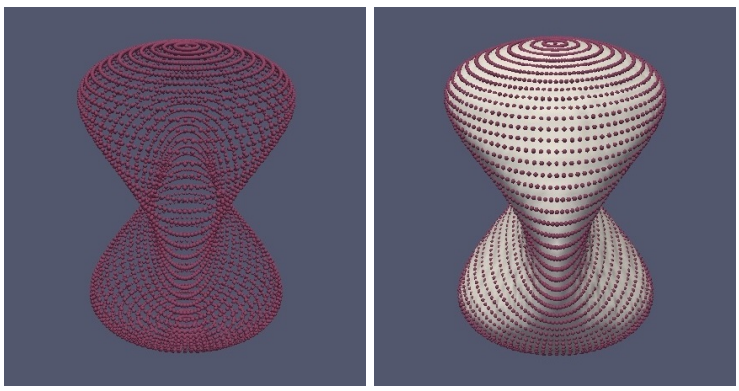

Fig. 4.1: On the left, we see the point cloud data, on the right the point cloud with the created 3D model.

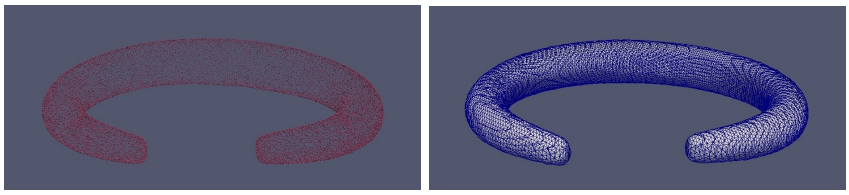

Fig. 4.2: Archaeological finds: bracelet. On the left, we see the point cloud data, on the right the final result with triangulated surface.

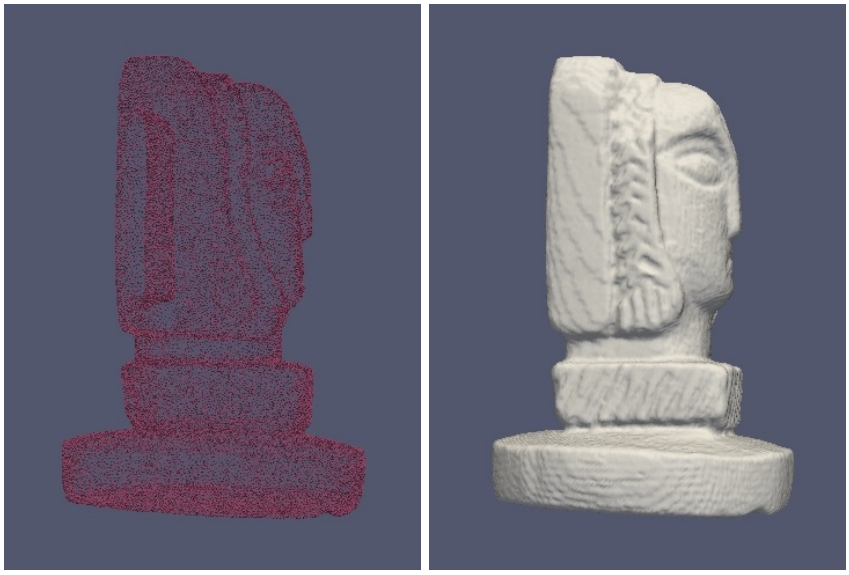

Fig. 4.3: Archaeological finds: sealer. On the left, we see the point cloud data, on the right the final result.

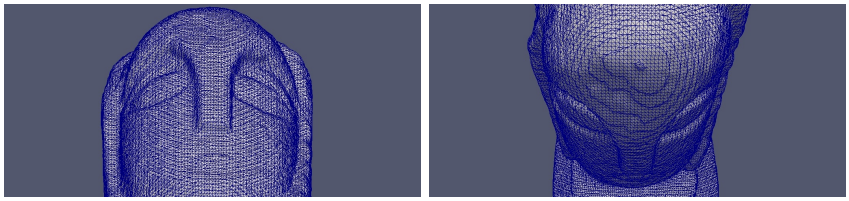

Fig. 4.4: Details of the sealer with triangulated surface.

We also tested our method on data sets with noise. In the point cloud data of the sealer we added artificial noise by changing the coordinates of 100 random points. Thanks to the curvature part of equation (2.1) this kind of noise has no effect on our created 3D model. We can observe that fact in Figure 4.5.

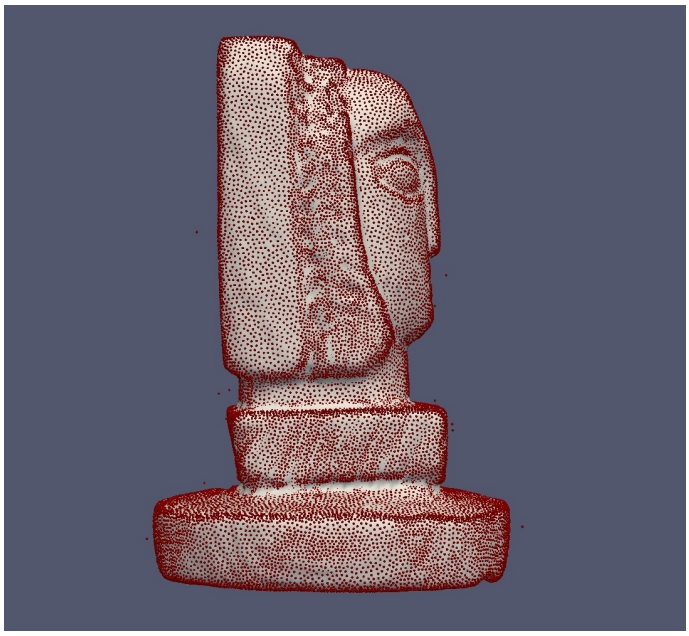

Fig. 4.5: Sealer point cloud data with noise, visualized with the final result.

5. Conclusions. In this work we presented our approach for surface reconstruction from point cloud data utilizing the level set method. We formulated the mathematical model, derived the time and spatial discretization and provided the reader with an exact description of the numerical solution. By implementing the method we obtained several interesting results for numerical tests and real-life data which we presented as examples in the last section. Our results show that for smoother objects a sparse grid already shows good result, but for an object with more detail we need more grid points. With adjusting the SOR method to our needs we achieved significant reduction of the required computational time, thus making our method more suitable for real-life application.

Acknowledgments. This work was supported by the grants APVV-15-0522 and VEGA 1/0608/15.

REFERENCES

[1] J. Haličková, K. Mikula, Level set method for surface reconstruction and its application in surveying, Journal of Surveying Engineering 143 (3). doi:10.1061/(ASCE)SU.1943-5428.0000159.

[2] H. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method, Computer Vision and Image Understanding 80 (2000) 295–319. doi:10.1006/cviu.2000.0875.

[3] H. K. Zhao, A fast sweeping method for eikonal equations, Mathematics of Computation 74 (2004) 603–627. doi:10.1090/S0025-5718-04-01678-3.

[4] S. Corsaro, K. Mikula, A. Sarti, F. Sgallari, Semi-implicit covolume method in 3d image segmentation, SIAM Journal on Scientific Computing 28 (6) (2006) 2248–2265. doi:10.1137/060651203.