Periodica Polytechnica
Electrical Engineering
and Computer Science

59(4), pp. 175-180, 2015
DOI: 10.3311/PPee.8529

Filtering and Gradient Estimation for Distance Fields by Quadratic Regression

László Szirmay-Kalos[1*]

RESEARCH ARTICLE

Received 28 August 2015; accepted 18 November 2015

Abstract

Distance fields show up in many problems of 3D vision and rendering, for example, a volumetric fusion of depth images results in such a field. Distance fields obtained from measured values are inherently noisy, so its filtering is needed before isoparametric surfaces are extracted from them, and robust normal vector estimation also requires a local smoothing since differentiation is especially sensitive to noise. In this paper, we use regression to find a quadratic function that approximates the zero level surface of the distance field, and apply this both for filtering and normal vector estimation. We also present a computationally efficient method that exploits the regular structure of samples, the symmetry and separability of the weighting functions, and thus avoids the solution of larger linear equations, which otherwise would become necessary when regression is generally attacked. The algorithm is a part of a real-time volumetric fusion application running on the Graphics Processing Units (GPU).

Keywords

distance fields, volumetric models, filtering, regression, 3d reconstruction

[1] Department of Control Engineering and Information Technology,
Faculty of Electrical Engineering and Informatics,
Budapest University of Technology and Economics,
H-1521 Budapest, P.O.B. 91, Hungary

[*] Corresponding author, e-mail: szirmay@iit.bme.hu

1 Introduction

Distance fields are volumetric models or 3D scalar fields where each point is associated with a scalar value of its distance from a given surface. A 2D distance field is shown by Fig. 1 [6] where gray levels encode distance values from a mouse figure and curves depict manifolds of constant distance. In 3D, distance fields are usually represented by a voxel array, which samples the distance values on a 3D Cartesian lattice a.k.a. regular grid.

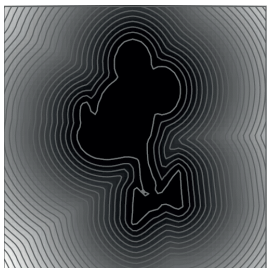

Fig. 1 2D distance map of a mouse figure: Distances are depicted by gray levels and isoparametric curves are also drawn.

Distance fields are closely related to *implicit surfaces* [3,8] defined by implicit equation $f(x,y,z) = 0$, which defines a surface by stating that the surface is a collection of those points whose distance from the surface is zero. One difference between implicit surfaces and distance fields is that we do not require implicit function f to give an Euclidean distance from the surface, but it is enough that, for example, it is positive if the point is outside of the object, negative if it is inside, and continuous, so the zero level set is just the surface. Another related paradigm is the *functional representation* [11], which is an intuitive mechanism to create implicit functions representing surfaces or solids. In geometric modeling the objective is to find a function for a given object, and rendering generates an image of that object. In reverse engineering, however, we have images and wish to find the function from these [13].

2 Previous work

Distance fields have been extensively used in computer graphics as a sampled representation of complex geometry

[10,3,12,8]. Efficient algorithms also exist for their generation and rendering (Fig. 2).

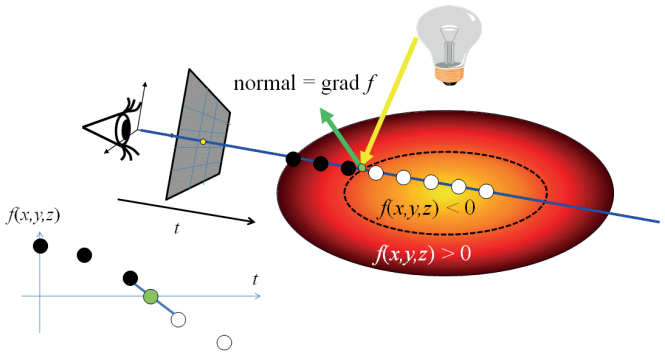

Fig. 2 Distance field rendering with ray marching: Along the rays emanated from the eye through the pixels, the algorithm makes small steps and reads the distance value at the visited point. When the distance value changes its sign, the zero level surface is crossed. The intersection of the surface and the ray can be further refined by local tri-linear interpolation. The pixel color is obtained by evaluating an illumination formula at the identified point, which also needs the normal vector of the intersected surface, which is computed as the gradient of the distance field.

In volumetric rendering, the normal vector of the surface is usually obtained as the approximated gradient of the 3D scalar field. As differentiation amplifies noise and numeric evaluation of the derivatives makes the underlying grid structure more visible, robust and artifact free normal vector estimation has been in the focus of research for many years. A particularly elegant technique attacks the problem as a 4D linear regression and finds a plane that approximates the isoparametric surface locally. The gradient is then the normal vector of this approximating plane [9]. Exploiting the fact that samples are in a regular structure, the regression problem can be solved as a convolution. The idea of finding normal vectors with regression has been generalized to quadratic approximation surfaces in [5], but that paper has not presented an efficient solution method. Instead, it requires the solution of a system of linear equations with ten unknowns in each grid point, which is unacceptable in a real-time application.

In this current paper we follow the direction proposed in [9], but use quadratic approximation and also exploit the fact that the scalar field is a distance field. The main contributions of this paper are as follows:

- We extend the idea of linear regression to quadratic regression and show that the quadratic regression problem can also be solved with similar efficiency as linear regression.
- We apply the proposed method not only for normal vector estimation but also for filtering distance values.
- An efficient and GPU-friendly computation scheme is presented, which allows the real-time filtering of distance fields emerged in volumetric fusion applications.

The organization of the remaining part of this paper is as follows. In Section 3, we present the proposed quadratic regression for 3D distance fields and discuss how it can be efficiently computed if the weighting scheme is symmetric. We also provide the formulae for filtering and normal vector estimation. Section 3.1 discusses further speed ups obtained when the weighting scheme is a separable function. Section 4 briefly summarizes our volumetric fusion application for 3D reconstruction from noisy depth images, which is the main application of the proposed regression scheme. Finally, Section 5 presents results from the volumetric fusion application.

3 Quadratic regression

Suppose that we have a 3D distance field F defined on a 3D grid. This 3D distance field is approximated in the neighborhood of a grid point (U, V, W) by a quadratic function using those neighboring grid points that are in an Axis Aligned Bounding Box (AABB) centered at the given grid point. For the sake of notational simplicity, we set up another coordinate system x, y, z with an origin at the given grid point, so these coordinates express the distances between the given grid point and a point in the neighborhood along the coordinate axes.

The approximating quadratic function is

$$f(x,y,z) = p_{200}x^2 + p_{020}y^2 + p_{002}z^2 + p_{110}xy + p_{101}xz + p_{011}yz + p_{100}x + p_{010}y + p_{001}z + p_{000}.$$

Unknown parameters $p_{200},\ldots, p_{000}$ are determined by minimizing the square error

$$E(p_{200},\ldots,p_{000}) = \sum_i w_i \left(f(x_i,y_i,z_i) - f_i\right)^2$$

computed between unknown function f and the original values f_i representing distance field F at relative grid point (x_i, y_i, z_i). In this equation w_i is the weight of sample i, which may depend only on relative coordinates (x_i, y_i, z_i).

To get the optimal coefficients, we should make the partial derivatives equal to zero with respect to each of them. For example, the partial derivative of the error term with respect to p_{200} is

$$\begin{aligned}\frac{\partial E}{\partial p_{200}} &= 0 \\ &= \left(\sum_i w_i x_i^4\right) p_{200} + \left(\sum_i w_i x_i^2 y_i^2\right) p_{020} + \left(\sum_i w_i x_i^2 z_i^2\right) p_{002} \\ &+ \left(\sum_i w_i x_i^3 y_i\right) p_{110} + \left(\sum_i w_i x_i^3 z_i\right) p_{101} + \left(\sum_i w_i x_i^2 y_i z_i\right) p_{011} \\ &+ \left(\sum_i w_i x_i^3\right) p_{100} + \left(\sum_i w_i x_i^2 y_i\right) p_{010} + \left(\sum_i w_i x_i^2 z_i\right) p_{001} \\ &+ \left(\sum_i w_i x_i^2\right) p_{000} - \left(\sum_i w_i x_i^2 f_i\right).\end{aligned} \tag{1}$$

If we assume that the domain of sample points is symmetric onto the coordinate planes and weighting scheme w_i depends just on the distance from the coordinate planes, all those terms $\sum_i w_i x_i^n y_i^m z_i^k$ are zero where at least one from n, m, k is an odd number. To prove this, let us consider the symmetric pair of a grid point, that is in the mirror position of the coordinate plane where the exponent is an odd number. All factors are the same for the original and symmetric pair except for the factor having odd exponent, which is just the negative version of the original term, thus the mirror pair cancels out the term of the original point. Thus, only those terms remain where n, m, k are all even numbers, and Eq. (1) can be written as:

$$\left(\sum_i w_i x_i^4\right) p_{200} + \left(\sum_i w_i x_i^2 y_i^2\right) p_{020} + \left(\sum_i w_i x_i^2 z_i^2\right) p_{002} + \left(\sum_i w_i x_i^2\right) p_{000} = \left(\sum_i w_i x_i^2 f_i\right).$$

To further exploit the symmetry, let us express Cartesian grid point coordinates x, y, z as products of integer grid point coordinates X, Y, Z and the distances of the grid points along the three axes, $\Delta x, \Delta y, \Delta z$:

$$x_i = X_i \Delta x,\ y_i = X_i \Delta y,\ z_i = Z_i \Delta z.$$

Note that we do not require the same spacing along the different coordinate axes.

Exploiting the symmetry of the three coordinate axes, we get

$$\sum_i w_i X_i^4 = \sum_i w_i Y_i^4 = \sum_i w_i Z_i^4 = A,$$

$$\sum_i w_i X_i^2 Y_i^2 = \sum_i w_i X_i^2 Z_i^2 = \sum_i w_i Y_i^2 Z_i^2 = B,$$

$$\sum_i w_i X_i^2 = \sum_i w_i Y_i^2 = \sum_i w_i Z_i^2 = C,$$

$$\sum_i w_i X_i^2 f_i = f_{200},$$

with which we can further simplify Eq. (1) to

$$A p_{200} (\Delta x)^2 + B p_{020} (\Delta y)^2 + B p_{002} (\Delta z)^2 + C p_{000} = f_{200}.$$

The symmetry and uniformity of the grid structure can be exploited also for the other partial derivatives as well. Computing the partial derivatives also with respect to p_{020}, p_{002} and p_{000}, we obtain the following system of equations:

$$B p_{200} (\Delta x)^2 + A p_{020} (\Delta y)^2 + B p_{002} (\Delta z)^2 + C p_{000} = f_{020},$$

$$B p_{200} (\Delta x)^2 + B p_{020} (\Delta y)^2 + A p_{002} (\Delta z)^2 + C p_{000} = f_{002},$$

$$C p_{200} (\Delta x)^2 + C p_{020} (\Delta y)^2 + C p_{002} (\Delta z)^2 + D p_{000} = f_{000}.$$

with the following new shorthand notations

$$\sum_i w_i Y_i^2 f_i = f_{020},\ \sum_i w_i Z_i^2 f_i = f_{002},\ \sum_i w_i f_i = f_{000},$$

$$\sum_i w_i = D.$$

This system of equations can be solved in closed form:

$$p_{000} = \frac{(A+2B) f_{000} - C(f_{200} + f_{020} + f_{002})}{(A+2B) D - 3C^2}. \quad (2)$$

If the distance field were exact, then the distance between the current grid point and the zero level surface would be p_{000}. Thus, the proposed filtering operation replaces the distance value at the center grid point by p_{000}.

If we wish to approximate the normal vector, i.e. the gradient of the distance field as well, then other polynomial coefficients are also needed, so we take the remaining unknown parameters and make the partial derivatives with respect to them equal to zero. The solution is

$$p_{200} = \frac{f_{200} - f_{000} - (C - BD/C) p_{000}}{(A-B)(\Delta x)^2},$$

$$p_{020} = \frac{f_{020} - f_{000} - (C - BD/C) p_{000}}{(A-B)(\Delta y)^2},$$

$$p_{002} = \frac{f_{002} - f_{000} - (C - BD/C) p_{000}}{(A-B)(\Delta z)^2},$$

$$p_{110} = \frac{f_{110}}{B \Delta x \Delta y},\ p_{011} = \frac{f_{011}}{B \Delta y \Delta z},\ p_{101} = \frac{f_{101}}{B \Delta x \Delta z},$$

$$p_{100} = \frac{f_{100}}{C \Delta x},\ p_{010} = \frac{f_{010}}{C \Delta y},\ p_{001} = \frac{f_{001}}{C \Delta z},$$

where

$$f_{110} = \sum_i w_i X_i Y_i f_i,\ f_{011} = \sum_i w_i Y_i Z_i f_i,\ f_{101} = \sum_i w_i X_i Z_i f_i,$$

$$f_{100} = \sum_i w_i X_i f_i,\ f_{010} = \sum_i w_i Y_i f_i,\ f_{001} = \sum_i w_i Z_i f_i.$$

The gradient of the distance field is then

$$(\nabla f)_x = 2 p_{200} x + p_{110} y + p_{101} z + p_{100},$$

$$(\nabla f)_y = 2 p_{020} x + p_{110} x + p_{011} z + p_{010}, \quad (3)$$

$$(\nabla f)_z = 2 p_{002} x + p_{101} x + p_{011} y + p_{001}.$$

Note that the gradient can be evaluated not only in grid points but everywhere, and the gradient estimation changes linearly between the grid points, which leads to smooth surface rendering.

3.1 Performance issues

The proposed method calculates independent parameters A, B, C, D and dependent parameters $f_{000}, \ldots, f_{200}$ and substitutes them into Eqs. (2) and (3). Note that parameters A, B, C, D

are independent of the actual distance field and can be obtained knowing only the weighting function and the size of the neighborhood. So these parameters are pre-computed and stored as constants.

The computation of dependent parameters $f_{000},\ldots, f_{200}$ is equivalent to the execution of discrete convolutions of distance field $F(U, V, W)$. If we are calculating, for example, f_{200} for grid point U, V, W and the domain of sample points is a cube, then index i corresponds to that grid point where the differences between the grid coordinates and U, V, W are equal to $X, Y, Z,$ respectively:

$$f_{200}(U,V,W) = \sum_i w_i X_i^2 f_i$$
$$= \sum_X \sum_Y \sum_Z W(X,Y,Z) F(U-X, V-Y, W-Z) = W * F$$

where $W(X, Y, Z) = w(X, Y, Z) \cdot X$. If the cubic domain covers N voxels along all three dimensions, then it contains N^3 voxels, so the algorithm has cubic complexity with respect to the filter size.

The $O(N^3)$ complexity can be reduced to linear, $O(N)$, complexity in case of separable filters where the filter kernel can be expressed as a product of three factors depending just on a single coordinate, i.e.

$$W(X,Y,Z) = W_X(X) W_Y(Y) W_Z(Z),$$

by executing three consecutive filtering steps along the three axes:

$$F_Z(U,V,W) = \sum_Z W_Z(Z) F(U,V,W-Z),$$

$$F_{YZ}(U,V,W) = \sum_Y W_Y(Y) F_Z(U,V-Y,W),$$

$$f_{200}(U,V,W) = \sum_X W_X(X) F_{YZ}(U-X,V,W).$$

Note that kernels W of all equations needed to compute the dependent parameters are products of weights w and coordinates X, Y, Z. So if w is a separable function, all kernels W will also be separable functions. Of course, the filter kernels will be different in different dimensions because they are occasionally similar to the weight and to the product of the weight and the coordinate in other cases. For example, if w is a box or a Gaussian filter, then it is separable, which also makes W separable, allowing the processing of larger neighborhoods in real-time.

4 Application in volumetric fusion

Depth cameras like Kinect obtain the distance of points visible in different pixels, i.e. geometric information about the scene. Moving the camera around the object, geometric information of different images can be fused together into a valid 3D model. Additionally, the temporary result of the fused data can be used to find where the camera is moved, executing simultaneous localization and mapping (SLAM). A famous approach to depth image fusion is the Curless-Levoy algorithm [2] that results in a 3D distance field of the reconstructed surface. Scene objects can then be extracted as finding the zero level surface of this distance field. The output can be a point cloud generated by the intersection of the zero level surface and the lattice edges of the grid, or a valid triangle mesh generated with the marching cubes algorithm [7].

If the camera does not move, the Curless-Levoy algorithm estimates the signed distance between the surface and the grid point along the ray defined by the current pixel. The estimate is simply the arithmetic mean of the distances obtained from different measurements. To avoid the interference of surfaces on opposite sides, this signed distance function is truncated. If the camera moves, then the same arithmetic mean is used to update distance values. As it is shown in [2], this corresponds to the reconstruction of the isosurface with minimal squared error where the weighting depends on how long the surface is seen recently from different angles.

We have one remaining problem, to determine how the camera moves. The method is incremental, if the camera parameters are known in the previous frame, just the rotation and translation between the current and previous frames should be computed, and this new transformation is concatenated to the camera transformation of the previous frame. If we could identify a set of corresponding point pairs on two images, then we could find that rotation and translation which would align the two point clouds making the distance of the new point cloud and the transformation of the previous point cloud close to zero. This is an optimization problem, which can be solved iteratively.

To find possibly corresponding pairs, we exploit the fact that just a little time has elapsed between two frames, so a point remains in the same pixel with high probability [4]. This is not always true, so outliers must be detected and rejected. So we take the distance field and generate a point cloud from the camera of the previous frame. Then we take the current depth map and back project it to obtain a point cloud in the new camera coordinate system. The camera transformation between the current and previous frames is expected to align the two point clouds. Alignment is detected when back projected points are on the zero level surface of the distance field and also that the normal vectors of the back projected mesh at these points are similar to the normal vectors of the zero level surface. If either the distance or the normal vector difference is very large, then probably two non-corresponding points are taken, so they are rejected as outliers. Otherwise, we consider this pair as corresponding and use them while minimizing the total distance of the two point clouds.

The discussed camera tracking algorithm assumes a reasonably accurate distance field and normal vectors. This requirement makes our quadratic filtering scheme a good candidate to improve the SLAM approach.

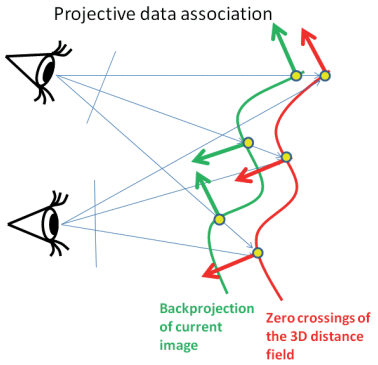

Fig. 3 Modified ICP algorithm applied in volumetric fusion

5 Results

We have added the proposed method to our volumetric fusion application that is similar to KinectFusion [4,1]. This system is implemented in CUDA and runs on NVIDIA 690 GT GPUs at 30 FPS when fusing 640×480 resolution depth images into 128^3 or even higher resolution hierarchical voxel arrays.

In the implementation of the quadratic regression, we use Gaussian weighting and execute three filters along the three axes applying the concept of separable filtering. In each of the filtering steps, a computational thread is assigned to the output voxel addressed by U, V, W. When the third linear filtering is done, all parameters are available, thus the distance value can be modified according to Eq. (2).

Figure 4 shows the reconstruction from a depth image sequence that is generated in OpenGL with controllable noise. Note that we added significant amount of noise of amplitude equal to the 3 percent of the total depth range of the camera. The figure shows two frames, one is at the beginning of the fusion, so here the distance field is rather noisy since there were not enough depth images to construct a noise-free 3D distance field. In the second frame, the fusion also reduces noise at those surfaces that have been visible for a longer period of time. In both cases, the proposed filtering scheme can greatly reduce the noise, which is crucial in camera tracking as well.

Figure 5 shows the reconstruction from a depth image sequence provided by a Kinect2 depth camera in our laboratory. Compared to Kinect1, the noise level of this new camera is much lower, but newly seen surfaces and large camera motions are still helped by our proposed method.

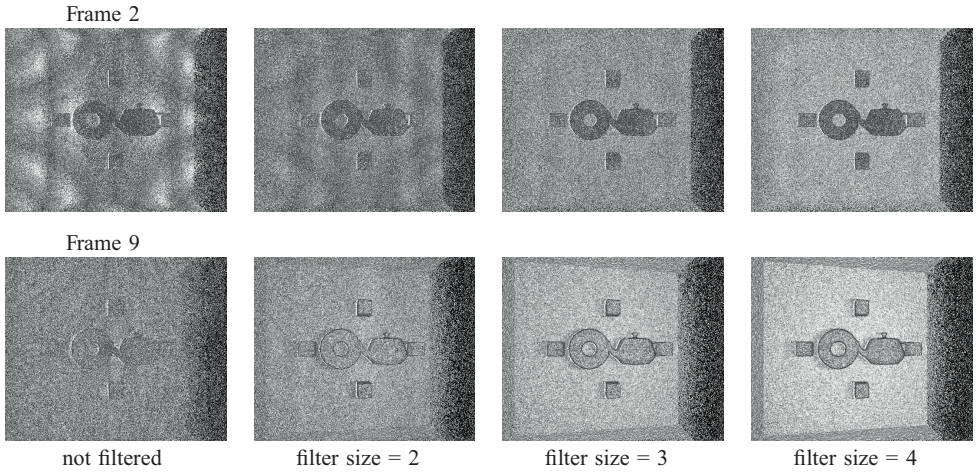

Fig. 4 Teapot with torus and cubes in a room. The upper row shows the second frame, the lower row the ninth frame of the video captured during the reconstruction and camera localization.

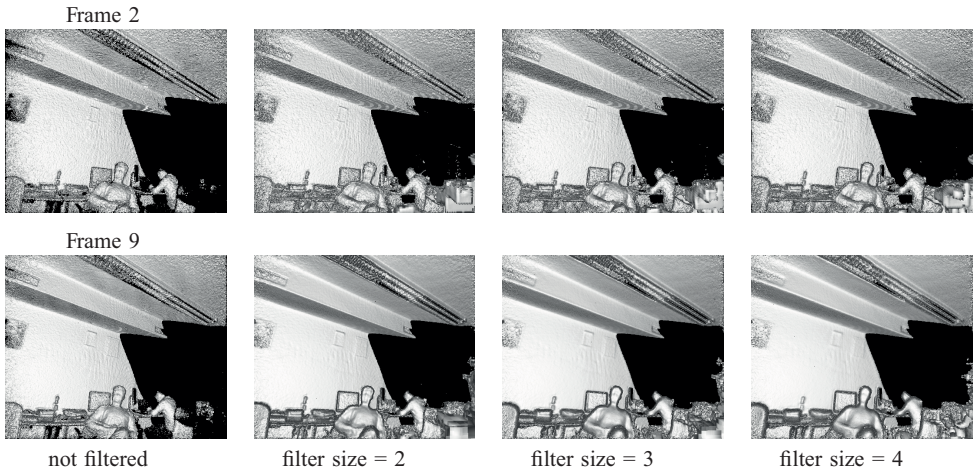

Fig. 5 IB312C scene with hard working researchers. The upper row shows the second frame, the lower row the ninth frame of the video captured during the reconstruction and camera localization.

6 Conclusions

In this paper we proposed a filtering and gradient estimation scheme for 3D distance fields. The basic idea is to compute a three-variate quadratic regression for the noisy data, then the samples are replaced by the values of the quadratic function. This can be used as filtering and also to robustly estimate the gradient, i.e. the normal vector of the isosurfaces. To reduce the computational complexity, we generalized a technique that has been invented for linear regression, and exploited the regular distribution of the sample values.

The method runs on the GPU and is a part of a real-time camera tracking and scene reconstruction application.

Acknowledgement

This work has been supported by OTKA K–104476 and by VKSZ 14-1-2015-0072 SCOPIA.

References

[1] Chen, J., Bautembach, D., Izadi, S. "Scalable real-time volumetric surface reconstruction." *ACM Transactions on Graphics (TOG)*. 32 (4). pp. 113:1-113:16, 2013. DOI: 10.1145/2461912.2461940

[2] Curless, B., Levoy, M. "A volumetric method for building complex models from range images." In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, (SIGGRAPH '96). New York, NY, USA, pp. 303-312. 1996. ACM. DOI: 10.1145/237170.237269

[3] Hart, J. C. "Sphere tracing: Simple robust antialiased rendering of distance-based implicit surfaces." In: SIGGRAPH 93 Course Notes: Modeling, Visualizing, and Animating Implicit Surfaces. pp 14:1-14:11. 1993.

[4] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A. "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera." In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, (UIST '11). New York, NY, USA, pp. 559-568. 2011. ACM. DOI: 10.1145/2047196.2047270

[5] Jirka, T., Skala, V. "Gradient vector estimation and vertex normal computation." In: Szczyrk International Workshop. pp. 27-32. 2003.

[6] Kimmel, R., Kiryati, N., Bruckstein, A. M. "Sub-pixel distance maps and weighted distance transforms." *Journal of Mathematical Imaging and Vision*. 6 (2-3). pp. 223-233. 1996. DOI: 10.1007/BF00119840

[7] Levoy, M. "Display of surfaces from volume data." *IEEE Computer Graphics and Application*. 8 (3). pp. 29-37. 1988. DOI: 10.1109/38.511

[8] Liktor, G. "Ray tracing implicit surfaces on the GPU." In: Central European Seminar on Computer Graphics, CESCG '08, 2008.

[9] Neumann, L., Csébfalvi, B., König, A., Gröller, E. "Gradient estimation in volume data using 4D linear regression." In: Computer Graphics Forum, 19(3). pp. 351-358, 2000.

[10] Paglieroni, D. W., Petersen, S. M. "Height distributional distance transform methods for height field ray tracing." *Transactions on Graphics*. 13 (4). pp. 376-399. 1994. DOI: 10.1145/195826.197312

[11] Pasko, A., Adzhiev, V., Sourin, A., Savchenko, V. "Function representation in geometric modeling: concepts, implementation and applications." *The Visual Computer*. 11(8). pp. 429-446. 1995. DOI: 10.1007/BF02464333

[12] Szirmay-Kalos, L., Aszódi, B., Lazányi, I., Premecz, M. "Approximate ray-tracing on the GPU with distance impostors." Computer Graphics Forum (Eurographics '05), 24 (3). pp. 695-704, 2005.

[13] Várady, T., Martin, R. R., Cox, J. "Reverse engineering of geometric models - An introduction." *Computer-Aided Design*. 29(4). pp. 255-268. 1997. DOI: 10.1016/S0010-4485(96)00054-1