Title:
Distance Field Generation By Proxies

Authors:
Anna Lili Horváth, srirm5@inf.elte.hu, Eötvös Loránd University, Budapest
Gábor Valasek, valasek@inf.elte.hu, Eötvös Loránd University, Budapest
Róbert Bán, rob.ban@inf.elte.hu, Eötvös Loránd University, Budapest

Keywords:
Signed Distance Function, Implicit Representation, Geometric Modeling

DOI: 10.14733/cadconfP.2024.189-193

Introduction:

Distance computation from volume boundaries is an elementary operation in geometric modeling. However, closed-form expressions are only available in special cases and general point-boundary and boundary-boundary queries are resolved via iterative numerical methods. Although these algorithms can satisfy precision constraints and thus provide sufficiently accurate numerical substitutions to the real distance, they have to interface with every geometric representation of the system. Consequently, the implementation of such algorithms is intertwined with the combinatorial complexity of the modeling system. We propose an approach that resolves distance queries via geometric proxies and thereby the actual geometric representation is opaque to the spatial query algorithm. The geometric proxies have a prescribed order of contact with the progenitor, i.e. original geometry and offer efficient and closed-form distance computation from points.

First and second order geometric proxies:

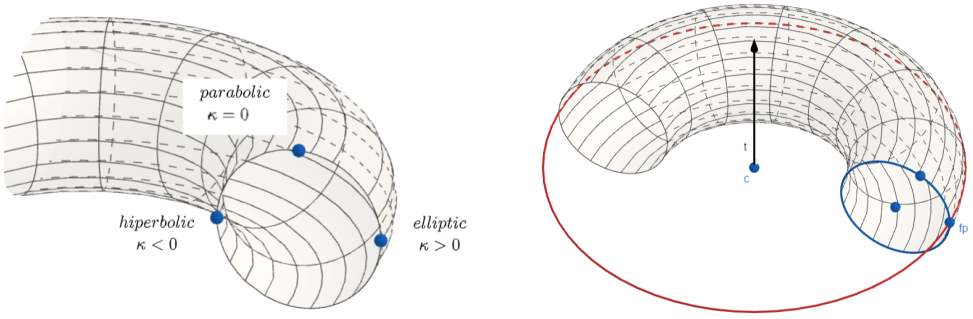
Distance fields are a set of proxies that approximate the original shape at prescribed surface points. Our goal is to establish a G^n continuous connection between the proxies and the original surface at the sample points. By this we mean that in these points there exists a regular parametrization for the proxy ensuring C^n continuity with the surface. We also want our proxies to be simple so evaluating their distance function is computationally efficient. Keeping these in mind we used the proxies described below.

The first order proxy has to reconstruct the surface point and the normal. The simplest such G^1 surface is the tangent plane, which can be defined by the said point and vector.

In addition to the mentioned quantities, the second-order field must store the principal curvatures and principal directions. The torus serves as a simple geometry capable of achieving this, as it can represent every combination of the sign of the curvature at its surface points, as depicted in Figure 1 (a).

The torus can be defined by its center, two radii and axis. However, a more fitting description for our method is presented on Figure 1 (b), where the defining parameters are a surface point, the two curvatures at that point and the axis. This also allows us to represent the degenerate cases of cylinders and planes conveniently, where one or two of the curvatures is zero.

In conclusion the general algorithm consists of 3 steps:



(a) We can find every combination of the curvatures by sign on the surface of the torus. $\kappa = \kappa_1 \cdot \kappa_2$ is the Gaussian curvature

(b) The torus is represented by a $\boldsymbol{p}$ surface point, the κ_r and κ_R principal curvatures at that point, and the $\boldsymbol{t}$ axis

Fig. 1: The second-order field stores torii as they are able to represent every combination of curvatures, and can be defined in a way that fits our concept of the geometric field.

1. take sample points from the surface
2. obtain the necessary differential geometric quantities at that point
3. calculate the parameters of the geometric proxy

In the following section, we will introduce our approach of taking surface samples and the field constructing method for parametric surfaces.

Converting parametric representations to proxy sets:

As for parametric surfaces, our algorithm follows the general algorithm described above. We take sample points from the input surface, evaluate the derivatives, and calculate the proxies described in the previous section.

To obtain sample points, we can take two different approaches:

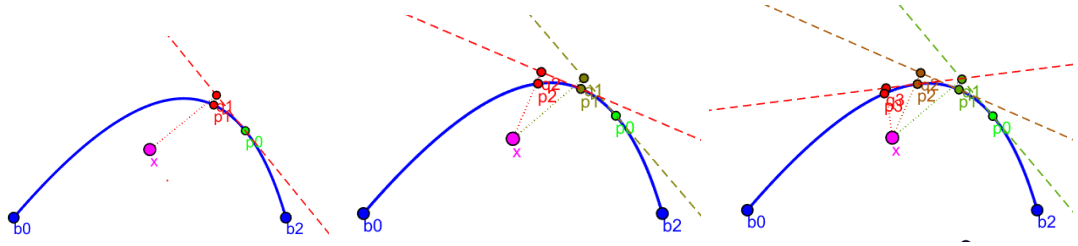
- take a uniform grid on the parameter space
- take a uniform grid in the $\mathbb{E}^3$ space and find the closest surface points (footpoints)

Opting for the first approach would result in unstructured data, making sampling slow, as each geometry would need to be checked individually to find the right distance. This challenge could be addressed by employing a spatial data structure, such as an octree. However, interpolation of the field would be still difficult for similar reasons.

To simplify the sampling and interpolation of the completed field, we opted to use a 3D uniform grid. The center position of each grid cell was taken, and the closest surface points, the footpoints were calculated for each.

To find the closest surface points we used the geometric Newton-Raphson method [1]. An outline of the method in 2D can be seen in Figure 2. The 3D version follows a similar approach but utilizes the tangent plane instead.

After computing the surface point and the corresponding u, v parameters we can calculate the first and second derivatives from the $\boldsymbol{s} : \mathbb{R}^2 \to \mathbb{E}^3$ parametric function of the surface.



(a) Step 1 (b) Step 2 (c) Step 3

Fig. 2: An example of the geometric Newton-Raphson method in 2D. We are looking for the closest point to $\boldsymbol{x}$ and our intial guess is $\boldsymbol{p}_0$. The figure shows two iterations of the method. We approximate the curve with lines and take a closest point from them $(\boldsymbol{q}_1, \boldsymbol{q}_2)$, then calculate the step in the original parameter space obtaining a better approximation of the closest point $(\boldsymbol{p}_1, \boldsymbol{p}_2)$.

The tangent plane can be defined with the $\boldsymbol{s}(u,v)$ footpoint and the normal vector there $\boldsymbol{n} = \frac{\partial_u \boldsymbol{s} \times \partial_v \boldsymbol{s}}{\|\partial_u \boldsymbol{s} \times \partial_v \boldsymbol{s}\|}$.

For the second order proxy, we can compute the κ_1, κ_2 principal curvature values with the following formulas. The first and second fundamental forms are expressed using

$$\mathbb{E} = \partial_u \boldsymbol{s} \cdot \partial_u \boldsymbol{s}, \quad \mathbb{F} = \partial_u \boldsymbol{s} \cdot \partial_v \boldsymbol{s}, \quad \mathbb{G} = \partial_v \boldsymbol{s} \cdot \partial_v \boldsymbol{s}, \quad \mathbb{L} = \partial_{uu} \boldsymbol{s} \cdot \boldsymbol{n}, \; \mathbb{M} = \partial_{uv} \boldsymbol{s} \cdot \boldsymbol{n}, \; \mathbb{N} = \partial_{vv} \boldsymbol{s} \cdot \boldsymbol{n} \; . \tag{2.1}$$

The principal curvatures are then

$$\kappa_1, \kappa_2 = h \pm \sqrt{h^2 - g}$$

where the Gauss and mean curvatures are

$$g = \frac{\mathbb{L} \cdot \mathbb{M} - \mathbb{N}^2}{\mathbb{E} \cdot \mathbb{G} - \mathbb{F}^2}$$
$$h = \frac{\mathbb{L} \cdot \mathbb{G} - 2 \cdot \mathbb{M} \cdot \mathbb{F} + \mathbb{N} \cdot \mathbb{E}}{2 \cdot (\mathbb{E} \cdot \mathbb{G} - \mathbb{F}^2)} \; .$$

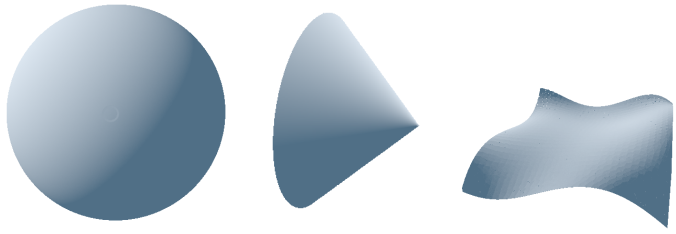
The axis of the torusis chosen as the direction of the bigger curvature.

After constructing a grid of proxies we can sample the field in runtime with a nearest sampling by simply evaluating the SDFs. The distance from the proxy plane can be computed with the general distance function. In the case of the torus, we have to separate three cases based on the curvature values:

- If $\kappa_r = \kappa_R = 0$, a plane is stored.
- If $\kappa_R = 0$, a cylinder is used
- If $\kappa_r \neq 0$ and $\kappa_R \neq 0$, the proxy is a torus.

We have to convert the parameters, then calculate the distances from the correct geometry. Because our proxies only approximate the surface in a small area around the footpoint, we restricted the geometries with a sphere by taking their intersection. This ensures that the geometries are only used in the area around the footpoint where they ensure the required approximation order.

For smoother connection between the proxies we can use trilinear interpolation. To do this we calculate the distances to the proxies in the 8 closest texels and interpolate the distance values.



(a) Sphere (b) Cone (c) Bézier surface

Fig. 3: Shapes used for accuracy measurements

Test results:

To test the accuracy of our method we conducted tests on basic shapes that are common in CAGD. Our test geometries can be seen in Figure 3. We took a fine gird with resolution of 217^3. At each grid point, we took the ground truth and evaluated our fields on multiple resolutions without filtering and with trilinear filtering. The error was the difference between the ground truth sample values and the computed distances from the lower resolution filtered approximations. We compared error vectors using the average $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ norms at three different resolutions with different sampling methods. The results are presented in Table 1 and 2.

Sphere

Res.	Norm	G1		G2	
		nearest	cubic H.	nearest	quintic H.
16^3	1	0.0019	0.003	0.00000016	**0.0000001**
	2	0.0000093	0.000015	**0**	**0**
	∞	0.069	0.044	0.02	**0.000012**
32^3	1	0.0005	0.0008	**0.0000017**	0.00012
	2	0.0000022	0.000013	**0.0000000027**	0.000014
	∞	0.033	0.29	**0.0033**	0.29

Cone

Res.	Norm	G1		G2	
		nearest	cubic H.	nearest	quintic H.
16^3	1	0.00089	0.0013	0.00055	**0.00052**
	2	0.0000039	0.0000055	0.0000012	**0.0000011**
	∞	0.085	0.087	0.082	**0.08**
32^3	1	0.00036	**0.00033**	0.00049	0.00046
	2	**0.00000099**	0.000001	0.000001	0.000001
	∞	**0.092**	**0.092**	**0.092**	0.093
64^3	1	0.00037	**0.00034**	0.00045	0.00043
	2	**0.00000077**	0.00000089	0.0000011	0.0000011
	∞	**0.073**	0.11	0.11	0.1

Table 1: Accuracy measurements of the field generated for parametric surfaces. The table shows the results with nearest sampling and blended Hermite interpolation as shown in [2]

The tests reveal that our method achieves high accuracy even with small resolutions. However, it is observed that the error does not always decrease with increasing resolution and the maximum error is high compared to the average, which can be attributed to certain limitations in our footpoint finding

Random Bézier surface

Res.	Norm	G1		G2	
		nearest	cubic H.	nearest	quintic H.
16^3	1	**0.031**	0.065	0.039	0.042
	2	**0.0028**	0.014	0.004	0.015
	∞	**0.18**	0.54	**0.18**	0.67
32^3	1	**0.02**	0.063	0.023	0.044
	2	**0.0011**	0.015	0.0013	0.015
	∞	**0.14**	0.51	**0.14**	0.67
64^3	1	**0.012**	0.064	0.017	0.044
	2	**0.00037**	0.015	0.00071	0.016
	∞	**0.11**	0.52	0.12	0.67

Table 2: Accuracy measurements of the field generated for random Bézier surfaces. The table shows the results with nearest sampling and blended Hermite interpolation as shown in [2]

algorithm. Our approach of constructing and evaluating the field is highly dependent on the accuracy of the footpoints therefore even small inaccuracies can cause large errors.

Besides, the tests also show that the order 1 field performs better than the order 2 for more complex shapes, due to the more robust construction algorithm.

Lastly, we can see that the simple interpolation we used does not fit the concept of the fields at all. The interpolation is conducted on the distance values thus we lose a lot of geometrical data.

Conclusions:

We presented a method for constructing distance fields for parametric surfaces using order 1 proxy geometries. Our approach demonstrated efficiency on simple shapes and Bézier surfaces. In the future, our plan is to improve this method by using a more stable footpoint finding algorithm. Additionally, we aim to apply the same approach to construct distance fields for implicit surfaces. We also plan on developing a filtering technique that keeps the geometric meaning of the field.

Acknowledgement:

Supported by the ÚNKP-23-2 and ÚNKP-23-4 New National Excellence Program of the Ministry for Culture and Innovation from the National Research, Development and Innovation Fund.

Anna Lili Horváth, https://orcid.org/0009-0006-2956-9227
Gábor Valasek, https://orcid.org/0000-0002-0007-8647
Róbert Bán, https://orcid.org/0000-0002-8266-7444

References:

[1] Kallay, M.: A geometric Newton–Raphson strategy,Computer Aided Geometric Design, 2001. https://doi.org/10.1016/S0167-8396(01)00070-X

[2] Valasek, G. and Bán, R.: Higher Order Algebraic Signed Distance Fields, Computer-Aided Design and Applications, 2023. https://doi.org/10.14733/cadaps.2023.1005-1028