ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN
http://www.wscg.eu
WSCG 2025 Proceedings

Distance Fields in Monte Carlo Geometry Processing

Anna Lili Horváth
Eötvös Loránd University
Pázmány P. sétány 1/C.
Hungary 1117, Budapest
srirm5@inf.elte.hu

Gábor Valasek
Eötvös Loránd University
Pázmány P. sétány 1/C.
Hungary 1117, Budapest
valasek@inf.elte.hu

Róbert Bán
Eötvös Loránd University
Pázmány P. sétány 1/C.
Hungary 1117, Budapest
rob.ban@inf.elte.hu

Abstract

We propose to use discrete signed distance representations in Monte Carlo geometry processing simulations. In particular, we investigate the application of algebraic and geometric generalizations of traditional signed distance fields. These are means to unify signed distance and closest point queries that are required by the walk-on-spheres algorithmic framework. We apply these to plane shapes enclosed by parametric polynomial boundaries. Our tests quantify the performance-accuracy trade-off compared to brute force closest point queries on test shapes.

Keywords— Computer Graphics, Geometry Processing, Monte Carlo

1 Introduction

Rendering photorealistic images can be formulated as the solution to a Fredholm integral equation of the second kind [1]. Monte Carlo approaches have proven to be highly practical means to solve these and have been adapted in production graphics renderers [2]. Following the introduction of graphics cards with hardware accelerated ray tracing support, Monte Carlo path tracing methods have been also brought to the real-time setting [3].

In contrast, finite element methods (FEM) are most often used to analyze geometric objects by means of solving various partial differential equations (PDEs). These, however, rely on the existence of sufficiently high quality lower order approximation to the input - most often triangular or tetrahedral meshes.

Sawhney and Crane [4] proposed to adapt Monte Carlo methods to the geometry processing context. In particular, they showed how Monte Carlo formulations can be applied to the solution of some common partial differential equations (PDE) defined over geometric shapes. Their work generalizes the class of problems that can be addressed within this framework, building upon the idea of walk-on-spheres [5]. They have also demonstrated how acceleration techniques developed within the photorealistic image synthesis framework can be applied to geometry processing.

Within the Monte Carlo geometry processing (MCGP) framework, a significant computational effort is spent on distance queries to closest boundary points. To accelerate these, we propose to adapt precomputed discrete distance representations.

We show that standard zero order algebraic distance fields, most often implemented as bilinearly filtered textures [6], are efficient means to accelerate MCGP tasks on 2D shapes defined by paremetric boundaries. Moreover, we show that higher order constructs, in particular, geometric distance fields (GDFs) [7] improve on accuracy further.

In Section 2, we present the related work in the area of Monte Carlo algorithms. In Section 3, we review the concept of geometric distance fields, and how to generate and query them. In the last Section 4, we examine the accuracy and performance of Monte Carlo geometry processing method with distance field closest point proxies.

2 Related Work

Monte Carlo geometry processing solves PDEs by evaluating random walks. By noting that a continuous random walk within an open ball reaches any point on the boundary of the ball with equal probability, one can substitute the walk by taking uniform samples on the surface of the sphere instead [5]. In case of interior starting points, this leads to the most efficient walks by using maximal spheres in the sense that their radii are the largest such that no boundary point is contained within the interior points. The walk terminates when it reaches a sufficiently small neighborhood of the boundary. Sawhney and Crane's proposed solution for the Laplace equation is summarized in Algorithm 1 [4].

MCGP is agnostic to the geometric representation, as it only relies on the existence of closest boundary point distance queries. While this can be done efficiently on signed distance function (SDF) representations, it requires acceleration structures for other cases. Still, even in the presence of such, robust and high performance closest point finding algorithms are not trivial to implement for parametric descriptions [8] and general implicitly defined volumes.

Our paper proposes to use algebraic and geometric distance fields as closest point query interfaces for MCGP methods. We examine the accuracy and per-

Algorithm 1 Walk on spheres for solving $\Delta u = 0$ on Ω, $u = g$ on $\partial\Omega$

```
u = 0                                   ▷ Solution estimate
for i = 1 ... nWalks do
    x = x_0                             ▷ Start a new walk
    do
    ▷ Move to random point on
    ▷ largest empty sphere
        r = distance(x, ∂Ω)
        x = randomSphere(x, r)
    While (r > ε)                       ▷ Close enough
    ▷ Sample boundary value
    u = u + g(closestPoint(x, ∂Ω))
end for
return u/nWalks
```

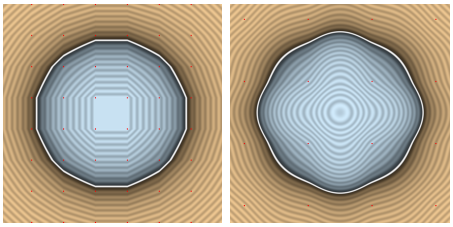


Figure 1: *The reconstruction of a circle using a* 8×8 *order 0 field (64 scalars) and a filtered* 4×4 *order 1 geometric field (54 scalars)*

formance of these methods in the plane by solving Poisson equations on shapes bounded by low degree polynomial curves. In general, we investigate if there is a systematic performance and accuracy characteristic to various discrete SDF representations.

GDFs are discrete data structures that store simple basic shapes – geometric proxies – that can be used to approximate the SDF of the original input. The proxies are chosen such that their SDF is trivial to evaluate and are higher order approximations to the input SDF locally. For example, a first order GDF in the plane stores halfplanes at each sample. The SDF of such a halfplane reconstructs both the original SDF and its gradient [7] at the sample position. Upon distance queries, the SDFs of the closest halfplanes are evaluated and the results are blended together in a sufficiently continuous manner. In general, it can be shown that the SDFs of the tangent line and the osculating circle at the closest point are first and second order approximations to the original SDF [7], respectively. Figure 1 demonstrates the quality of SDF reconstruction of GDFs compared to traditional bilinearly filtered distance textures.

Similarly to MCGP, GDFs are also agnostic to the input representation and only require the existence of closest point queries. However, these queries on the original input are only used during GDF construction, i.e., in preprocessing. At runtime, the GDF functions as a complete replacement of the SDF of the original input. Although the reconstructed isosurface may not be exact or continuous to the order of the original, MCGP methods are robust against perturbations and defects of the input such as these.

3 Monte Carlo Geometry Processing on Parametric Input

We present a pipeline to compute various kinds of distance fields as closest point proxies in 2D. We assume that the shapes in question are defined by Bézier curve segment boundaries.

In Section 3.1, we describe the input data consisting of Bézier curves and present our method of obtaining closest points on them. In Section 3.2, we introduce algebraic and geometric distance fields, and we show how to calculate them. Finally, in Section 3.3, we present our method of querying the different fields to obtain the approximation of the SDF and introduce a gradient-preserving filtering method.

3.1 Brute Force Footpoint Queries

Our input geometry of FreeType characters are given as lists of oriented outline segments. The segments are either linear or quadratic Bézier curves. Each segment is parameterized over $[0, 1]$. We assume that all segments are simple, without self intersections. To determine the closest surface point – called footpoint – at a query location $\boldsymbol{q}$, we iterate over all outline segments and calculate the distance between each segment and $\boldsymbol{q}$ by solving the corresponding cubic algebraic equation explicitly.

The sign of the distance, indicating whether the query point is inside or outside, is calculated from the local geometric data of the nearest segment. If the footpoint is an inner point of the segment (i.e., the corresponding t parameter is such that $0 < t < 1$), the sign is determined by the normal vector at the footpoint. Otherwise, if the footpoint is at a vertex, we need to account for the normal vector of both connecting segments.

3.2 Distance Field Generation

A distance field is a grid of samples, each of which stores data that can be used to reconstruct the SDF value locally. This value is exact at the sample position and an approximation elsewhere, in general. Upon querying the distance field, multiple samples are used to compute individual approximations to the SDF at the query position and these approximations are combined by blending. The blending may not necessarily improve the accuracy of the overall approximation, however, it does alter the order of continuity.

A distance field can be stored as a 2D texture, where the sample positions are the centers of rectangular

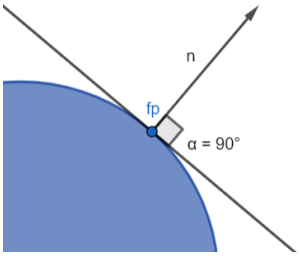


Figure 2: *The blue shape is our original geometry for which we want to construct the order 1 field. The first order geometric proxy is a half plane defined by the footpoint and the normal at the footpoint. The outside of the half plane is where the normal points.*

cells. We investigated two approaches as to what data to store: algebraic and geometric.

Algebraic fields use polynomials to approximate the SDF at the sample positions. The degree of the polynomial depends on the order of the field. The order 0 algebraic field (A0) stores one constant value in each sample, which is the SDF value at that position. When evaluating the field, we fetch the 4 closest samples to our query position and apply a bilinear filter on the stored distance values.

The first order approximation of the SDF can be represented by the three coefficients of a linear polynomial. These polynomials can be obtained with the value and partial derivatives of the SDF at each query point as shown in [9]. An order 1 algebraic (A1) field is composed of such polynomials.

Geometric distance fields have been introduced in [7]. These constructs use a proxy geometry at each sample that has the same SDF as the original shape at the sample position. The order 0 geometric distance field stores the footpoint to the query position and the sign of the SDF. This allows us to infer not only an approximate distance, similarly to an A0 field, but also to reason about the location of the footpoint. The existence of multiple footpoints is possible at samples along the medial axis, however, as the cut locus is a set of measure zero, this configuration has a very low probability. For this reason we only used one footpoint.

An order 1 approximation has to reconstruct the SDF value and gradient at the footpoint. As such, the tangent line at the footpoint is the correct first order geometric proxy since $\|\nabla f\|_2 \equiv 1$ for an SDF everywhere outside the cut locus. As an inside-outside partitioning of the plane is also needed, the simplest order 1 proxy is the half-plane defined by the tangent line at the footpoint with a normal vector coinciding with the normal vector at the footpoint. We use an outward pointing normal. This definition of the half plane can be seen on Figure 2.

A half-plane is defined by a point and a vector. We store the proxy with 3 scalar values, as the normal is parallel to the vector from the footpoint to sample position and its length is exactly the SDF value at the sample. So it is enough to store the distance value and the normal and the position of the footpoint can be obtained during evaluation.

Note that while we used shapes with linear and quadratic Bézier segments, the construction of the fields is general, and can be extended to higher order curves with the same algorithm.

3.3 Distance Field Queries

The distance field is stored in a 2D texture, with the defining properties described in the previous section. When evaluating the field, we can read the values from the field at each query point and calculate the distance in the following way.

When applying nearest point sampling at a point $\boldsymbol{p}$, we fetch the closest sample and read the data from the corresponding texel.

For algebraic fields, we can evaluate the linear polynomial with the stored coefficients.

For geometric fields, we calculate the $\boldsymbol{c}_p$ center of the field cell containing the query point $\boldsymbol{p}$. For each sample, we obtain the footpoint $\boldsymbol{f_p}$ from the distance value d and normal direction $\boldsymbol{n}$ as $\boldsymbol{f_p} = \boldsymbol{c}_p - d \cdot \boldsymbol{n}$. Finally, we calculate the signed distance from the half plane, with $d_p = (\boldsymbol{p} - \boldsymbol{f_p})^T \cdot \boldsymbol{n}$.

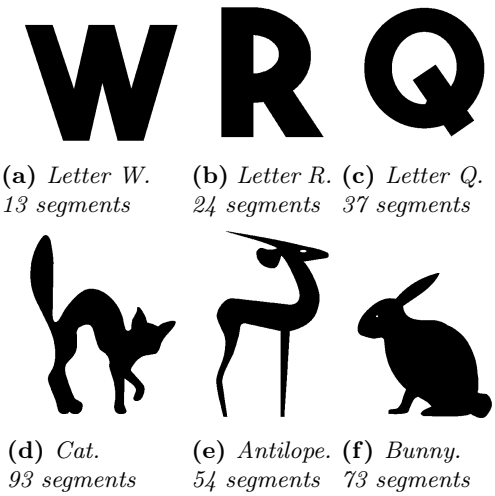
For a globally smoother result, we can use filtering by taking the distances inferred from the four closest samples and blend them. An optimization to the algebraic case was shown in [9], by highlighting that this can be done by first blending the coefficients and evaluating the resulting single polynomial. Note that this is not possible in the geometric case, instead, each geometric proxy has to be individually reconstructed and the blending has to be applied to the individual approximations.

The bilinear filtering would not preserve the gradients at the sample positions, only the SDF values. To achieve exact first order reconstruction, a gradient-preserving filtering has to be applied to the SDF values inferred from the samples, such as Hermite-based blending [7]. In particular, $h_3(t) = -2t^3 + 3t^2$ is used as the interpolating coefficient instead of the general value $t \in [0, 1]$ that represents the position of the query point between the two closest samples.

4 Test Results

We based our tests on an open source implementation of the MCGP method, in particular the Poisson solver of [10]. The program creates an image with given resolution where the color of each pixel represents the solution of the equation at that position. The solution is based on the walk-on-spheres algorithm and uses SDFs to calculate the step lengths.

The original algorithm used simple closed-form SDF functions, such as a circle. First, we expanded its ca-



(a) *Letter W.* *13 segments* **(b)** *Letter R.* *24 segments* **(c)** *Letter Q.* *37 segments*

(d) *Cat.* *93 segments* **(e)** *Antilope.* *54 segments* **(f)** *Bunny.* *73 segments*

Figure 3: *Shapes used for testing the accuracy and performance of the algorithm. All shapes are from the Freetype library [11] and consist of degree 2 and 3 Bézier curves. The number of the Bézier segments of each shape is presented in the captions.*

pabilities to handle shapes with linear and quadratic Bézier boundaries, so that it can support our FreeType [11] test shapes. Then we expanded it to evaluate more complex shapes by using precomputed geometric distance fields. The distance fields were generated with a fast GPU based algorithm, then transferred to main memory so that it can be evaluated on the CPU during the walk-on-spheres method.

All tests were done with a simluation grid of resolution 256×256. We used double precision binary64 numbers throughout the tests on the CPU. The number of walks performed for each point was 4096. We used the shapes presented in Figure 3. We solved the Poisson equation $-\nabla^2 u = c \cdot \|\boldsymbol{x}\|_2 - 0.1,\ \forall \boldsymbol{x} \in D$ where c is a scaling constant for displaying the result.

As ground truth, we solved the Poisson equation using the exact distance values obtained as discussed in Section 3.1. We iterated on the Bézier segments of the shapes and solved the exact algebraic equation to obtain the footpoint.

Then we compared the results with bilinearly filtered A0, bicubically filtered A1 and G1, and nearest neighbor filtered G1 fields. As our input data was analytical, the partial derivatives used for the A1 field are the same as the normal used for the G1 field, thus the two fields are equivalent and their evaluation gives the exact same value at any query position. Thus, we only present the results for the nearest neighbor and bicubically filtered G1 fields.

First, we investigated the accuracy of the solution by comparing the result of the PDE on the ground truth shape and the evaluated fields at each pixel using 4096 steps. We calculated the root mean square error of the values. The results are presented in Table 1. Although all runs are stochastic by nature, multiple measurements confirmed the systematic difference in terms of RMS between the different fields and filters.

The results show that the geometric field has the highest accuracy generally and in some cases an unfiltered geometric field is as accurate as an A0 field with a 4 times higher size. In the last test, we can see that on higher resolution the G1 field underperforms, this can be due to the fact that the geometric data is not always C^0 continuous as it depends on the sample positions, while the filtered A0 field is always continuous. This can be improved by applying an appropriate filtering method on the geometric field.

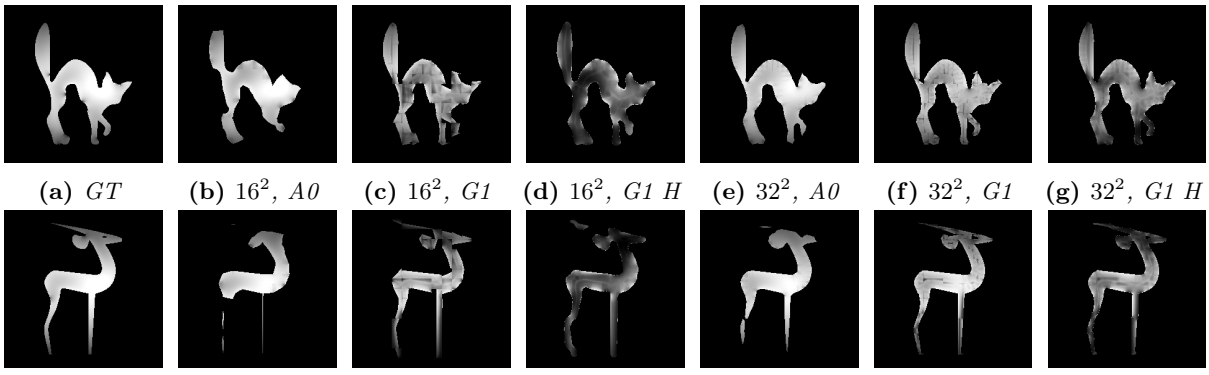
However, the results show that the Hermite filtering does not improve on the quality of the geometric field. This can be attributed to the fact that the cubic Hermite blend interpolation method does not use the geometric properties of the stored data, therefore, we loose information about the footpoints and the normals. To improve the quality, we need a filtering method that keeps the geometric interpretation of the data and uses geometric operations to improve on the approximation.

We visualized the solution to the PDE using the different fields in Figure 4. We solved the PDE $-\nabla^2 u = c \cdot cos(\boldsymbol{p}_x) \cdot sin(\boldsymbol{p}_y),\ \forall \boldsymbol{p} \in D$ and the color of each pixel shows the result of the equation at that point. As per results, the G1 field reconstructs the border more accurately and is close to the ground truth in most areas. There are artifacts around the cut locus caused by discontinuity in the field. The Hermite interpolation keeps the artifacts.

To examine this topic further, we compared three fields that reconstruct the border of a shape with high accuracy using the Antilope and Cat from Figure 3 (d) and (e) with a constant 1 impulse function as an example. The results can be seen on Figure 5. These show that the G1 field can reconstruct the border with a much smaller resolution. However the field has small artifacts on the inside, due to the discontinuities in the data. The G1 field is not C^0 everywhere, especially around the cut locus, as presented in Figure 6. This results in small errors, while still maintaining an overall high accuracy as presented in Table 1, because outside the problematic areas the field has a high quality. This means that the visual problems do not necessarily mean that the field is inaccurate, and even a visually imperfect geometric field can be used well for calculations. Also note that the A0 field does not reconstruct the cut locus exactly either, but its continuous nature prevents visible artifacts.

The discontinuities in the G1 field can be improved by using an appropriate filtering method. However, Figures 5 (d) and 6 (d) emphasize that the Hermit filtering is not suitable, as it ignores the geometric interpretation of the data and introduces further errors. In the future a new filtering technique is needed that can correct these issues in the neighborhood of the cut locus without decreasing the quality elsewhere.

Next, we compared the performance of the different methods. We compared the performance of the fields on an AMD Ryzen 7 5800H laptop CPU. The perfor-



(a) *GT* **(b)** 16^2, *A0* **(c)** 16^2, *G1* **(d)** 16^2, *G1 H* **(e)** 32^2, *A0* **(f)** 32^2, *G1* **(g)** 32^2, *G1 H*

(h) *GT* **(i)** 16^2, *A0* **(j)** 16^2, *G1* **(k)** 16^2, *G1 H* **(l)** 32^2, *A0* **(m)** 32^2, *G1* **(n)** 32^2, *G1 H*

Figure 4: *Visual comparison of Monte Carlo simulation on different fields. GT refers to ground truth and G1 H refers to a Hermite filtered G1 field. We solved the Poisson equation* $-\nabla^2 u = c \cdot cos(\boldsymbol{p}_x) \cdot sin(\boldsymbol{p}_y),\ \forall \boldsymbol{p} \in D$ *where c is a scaling constant for displaying the result. The figure shows that the geometric field reconstructs the border better and is accurate in most areas. It can also be seen that the G1 field is not continuous especially around the cut locus, causing large errors in the simulation. The Hermite filtering, while continuous, decreases the quality in the problematic areas. Further visualization of this topic is presented in Figure 5*

mance results are presented in Table 2. The time of generating the field is a few milliseconds during preprocessing, so it is not included in the results. It can be seen that using any geometric field improves the performance considerably. All distance fields are 1.5-2 orders of magnitude faster than using our naïve analytic closest point search. The bicubically filtered G1 fields are the slowest among the former.

The performance varies even between the different resolutions of the same field which can be attributed to the random nature of the Monte Carlo algorithm. The algorithm does not slow down as the resolution increases as it only has an influence on the size of the texture we have to read. If the texture fits in the cache of the computer, the time of reading is around the same, and calculating the distance from the values is independent of the resolution.

Lastly, as the Monte Carlo algorithm is a random algorithm, we tested the variation of the runtime and the RMS error. We used the Cat model from earlier with 32×32 resolution and run the algorithm 5 times. The results are presented in Table 3. The results have very little variation, suggesting that the tests and conclusions presented in the section are not impacted by the random nature of the algorithm.

5 Conclusions

We showed that signed distance fields can be used as an interface for SDF queries in a Monte Carlo algorithm. The results proved that the fields improve the performance while retaining the accuracy.

The first order geometric distance fields performed the best. These offered the same accuracy as the order 0 field with 4 times higher resolution in multiple tests and reconstructed the shapes well, even with small resolution fields. As for performance, simulations using order 0 and 1 fields had a similar runtime.

We saw that the filtering methods do not necessarily improve on the quality of the geometric field, because they ignore the geometric meaning of the data. Moreover, Monte Carlo geometric processing is tolerant to contour and shape defects, which makes nearest neighbor filtered first order geometric fields a good distance query interface.

Future research is required to devise filtering methods that can utilize the geometric nature of the samples used in geometric distance fields. Moreover, adapting our extensions to a GPU solution should improve simulation times considerably.

Acknowledgment

We want to thank Viktor Vad for the help with testing different filtering methods.

Supported by the EKÖP-24 University Excellence scholarship program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation fund. Supported by ELTE Eötvös Loránd University, Budapest, Hungary.

References

[1] James T. Kajiya. The rendering equation. SIGGRAPH Comput. Graph., 20(4):143–150, August 1986.

[2] Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, and Brian Green. Path tracing in production - part 1: Production renderers. In ACM SIGGRAPH

Table 1: *RMS error using different distance fields with respect to a ground truth obtained as the result of the walk on spheres with exact distances from the Bézier curves as presented in Section 3.1. First column denotes input field resolution. The results shows that overall the unfiltered G1 field performs best.*

Letter W			
Res.	A0	G1	G1 Hermite
16^2	0.022448	**0.015945**	0.027503
32^2	0.008155	**0.007294**	0.015496
64^2	0.005515	**0.004078**	0.009043
Letter R			
Res.	A0	G1	G1 Hermite
16^2	0.007907	**0.00667**	0.013495
32^2	0.005706	**0.004502**	0.009495
64^2	0.003413	**0.003205**	0.005489
Letter Q			
Res.	A0	G1	G1 Hermite
16^2	0.01184	**0.010659**	0.021224
32^2	**0.005602**	0.007097	0.012242
64^2	**0.003385**	0.004037	0.006192
Cat			
Res.	A0	G1	G1 Hermite
16^2	0.025754	**0.014624**	0.02721
32^2	0.01357	**0.010074**	0.019193
64^2	0.00679	**0.007743**	0.01106
Antilope			
Res.	A0	G1	G1 Hermite
16^2	0.025467	**0.017683**	0.026284
32^2	0.019596	**0.011498**	0.020464
64^2	0.009058	**0.007917**	0.014003
Bunny			
Res.	A0	G1	G1 Hermite
16^2	0.03196	**0.02434**	0.041565
32^2	**0.014702**	0.016008	0.029797
64^2	**0.00857**	0.011043	0.016562

2017 Courses, SIGGRAPH '17, pages 13:1–13:39, 2017.

[3] Alexey Panteleev and Christoph Schied. Real-time path tracing and denoising in 'quake 2'. Game Developer's Conference, 2019.

[4] Rohan Sawhney and Keenan Crane. Monte carlo geometry processing: a grid-free approach to pde-based methods on volumetric domains. ACM Trans. Graph., 39(4), August 2020.

[5] Mervin E. Muller. Some Continuous Monte Carlo Methods for the Dirichlet Problem. The Annals of Mathematical Statistics, 27(3):569 – 589, 1956.

[6] Chris Green. Improved alpha-tested magnification for vector textures and special effects. In ACM SIGGRAPH 2007 Courses, SIGGRAPH '07, page 9–18, New York, NY, USA, 2007. Association for Computing Machinery.

[7] Róbert Bán and Gábor Valasek. Geometric distance fields of plane curves. Acta Cybernetica, 25(2):187–203, Aug. 2021.

[8] Hongling Wang, Joseph K. Kearney, and Kendall Atkinson. Robust and efficient computation of the closest point on a spline curve. 2002.

[9] Gábor Valasek and Róbert Bán. Higher order algebraic signed distance fields. Computer-Aided Design and Applications, pages 1005–1028, 01 2023.

[10] Xiaochun Tong. Implementation of monte carlo geometry processing. `https://github.com/shiinamiyuki/mcgp`. Accessed: 2025-01-17.

[11] David Turner, Robert Wilhelm, and Werner Lemberg. The FreeType Project.

[12] Inigo Quilez. 2d sdf primitives. `https://www.shadertoy.com/playlist/MXdSRf`, 2023. Accessed: (2023-11-29).

Table 2: *Performance of the Monte Carlo algorithm on different input fields in seconds. The first row of each test case shows the simulation time required to obtain the ground truth. Bilinearly filtered A0 and unfiltered G1 fields have a similarly high performance, but compared to ground truth times, the computational overhead of Hermite blending is still negligible.*

Letter W			
ground truth		around 2 hours	
Resolution	A0	G1	G1 Hermite
16^2	61 s	64 s	168 s
32^2	59 s	66 s	165 s
64^2	57 s	67 s	171 s

Cat			
ground truth		around 3 hours	
Resolution	A0	G1	G1 Hermite
16^2	59 s	25 s	72 s
32^2	35 s	31 s	79 s
64^2	27 s	34 s	88 s

Letter R			
ground truth		around 2 hours	
Resolution	A0	G1	G1 Hermite
16^2	42 s	43 s	103 s
32^2	37 s	44 s	111 s
64^2	34 s	44 s	115 s

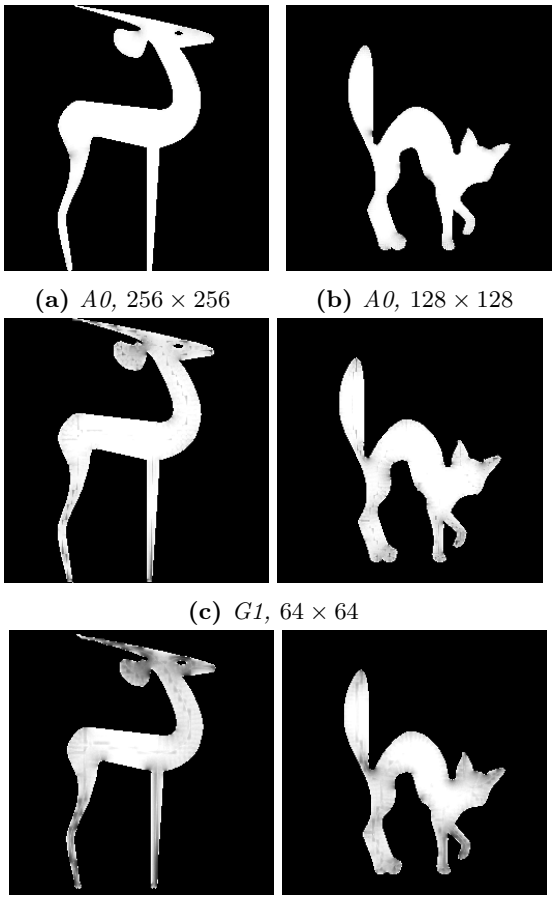
Antilope			
ground truth		around 3 hours	
Resolution	A0	G1	G1 Hermite
16^2	44 s	28 s	72 s
32^2	45 s	31 s	78 s
64^2	45 s	32 s	115 s

Letter Q			
ground truth		around 2 hours	
Resolution	A0	G1	G1 Hermite
16^2	62 s	59 s	129 s
32^2	47 s	53 s	134 s
64^2	41 s	57 s	143 s

Bunny			
ground truth		around 3 hours	
Resolution	A0	G1	G1 Hermite
16^2	71 s	66 s	175 s
32^2	55 s	66 s	171 s
64^2	61 s	70 s	176 s

Table 3: *The performance and accuracy of the algorithm after 5 runs. The test was done on the Cat model with 32×32 fields. These results show that there is small variance in both runtime and error.*

	A0		G1		G1 Hermite	
	Performance	RMS error	Performance	RMS error	Performance	RMS error
run 1	35.24 s	0.013567	31.77 s	0.010090	78.54 s	0.019194
run 2	35.43 s	0.013568	31.01 s	0.010076	78.03 s	0.019194
run 3	35.77 s	0.013570	30.82 s	0.010079	78.93 s	0.019194
run 4	35.49 s	0.013571	30.93 s	0.010092	78.58 s	0.019196
run 5	35.55 s	0.013569	30.79 s	0.010092	78.25 s	0.019186

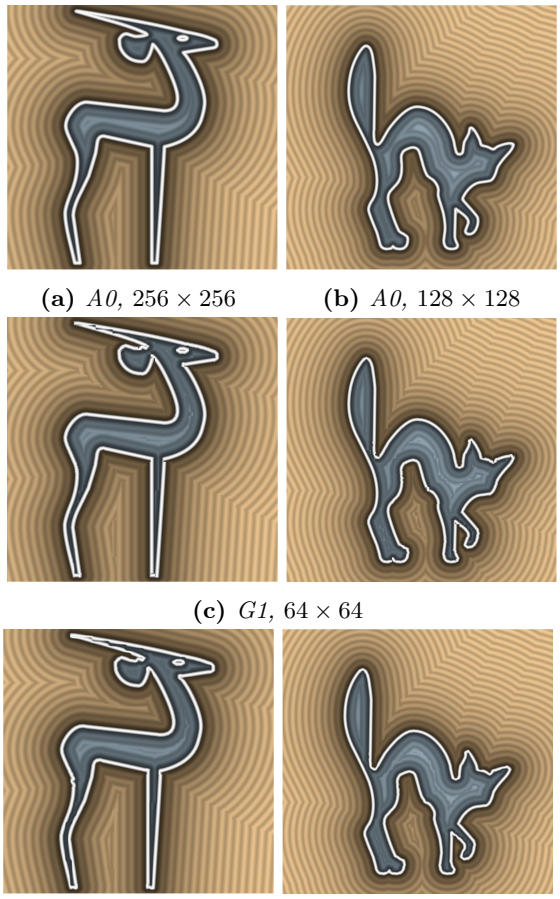


(a) *A0,* 256×256 **(b)** *A0,* 128×128

(c) *G1,* 64×64

(d) *G1 Hermite,* 64×64

Figure 5: *Comparison of a bilinearly filtered A0 (a), (b) an unfiltered (c) and an Hermite filtered (d) G1 field that reconstruct the border of the shape with high accuracy. We solved the PDE* $-\nabla^2 u = 1,\ \forall \boldsymbol{p} \in D$ *using the fields, and the pictures present the results of the equation as the color of each pixel. It can be seen that the A0 field needs a much higher resolution. The G1 field shows small artifacts on the inside. The unfiltered G1 field is not* C^0 *continuous everywhere, especially around the cut locus, resulting in small inaccuracies in the area, while keeping a generally high quality elsewhere as proved by the RMS errors in Table 1. Although the Hermite interpolation helps with the discontinuity, it does not fit the approach of the geometric data and introduces further error in the problematic areas. Further visualisation can be seen on Figure 6.*



(a) *A0,* 256×256 **(b)** *A0,* 128×128

(c) *G1,* 64×64

(d) *G1 Hermite,* 64×64

Figure 6: *Comparison of the distance function computed from a bilinearly filtered A0 (a), (b) an unfiltered (c) and an Hermite filtered (d) G1 field that reconstruct the border of the shape with high accuracy. The coloring is adapted from [12]. The G1 field is not continuous around the cut locus, while the distance value is accurate elsewhere. The discontinuity can be solved using an appropriate filtering method that keeps the overall good quality of the field. The figures in (d) show that the Hermite filtering is not sufficient, as it is performed on the distance values, and makes the area of the cut locus inaccurate while decreasing the quality in other areas too.*