Analysis of the Weight Function for Implicit Moving Least Squares Techniques

by

Zhujun Yao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, I analyze the weight functions used in moving least squares (MLS) methods to construct implicit surfaces that interpolate or approximate polygon soup. I found that one previous method that presented an analytic solution to the integrated moving least squares method has issues with degeneracies because they changed the weight functions to decrease too slowly. Inspired by their method, I derived a bound for the choice of weight function for implicit moving least squares (IMLS) methods to avoid these degeneracies in two-dimensions and in three-dimensions. Based on this bound, I give a theoretical proof of the correctness of the moving least squares interpolation and approximation scheme with weight function used in Shen et al. [34] when used on closed polyhedrons. Further, previous IMLS implicit surface reconstruction algorithms that fill holes and gaps create surfaces with obvious bulges due to an intrinsic property of MLS. I propose a generalized IMLS method using a Gaussian distribution function to re-weight each polygon, making nearer polygons dominate and reducing the bulges on holes and gaps.

Acknowledgements

I would like to thank my supervisor, Prof. Stephen Mann, for his always constructive suggestions, feedback and support. I would like to thank Prof. Bill Cowan and Prof. Christopher Batty for being my thesis reader. I would like to thank everyone in the Computer Graphics Lab and my friend Huichan Yao for everything they did to help. I would like to thank Ang Li for his assistance on the 3D spherical coordinate reformulation proof. I would also like to thank the anonymous reviewers in Graphics Interface 2014 for their meaningful comments. I would like to thank NSERC for their fund on this research.

Dedication

This is dedicated to my parents for their endless love. This is also dedicated to my boyfriend Ang. This thesis cannot be completed without his love, encouragement, and helpful discussions.

Table of Contents

List of Figures

Chapter 1

Introduction

In this thesis, I analyze the weight function used in implicit moving least squares schemes. Based on a test I ran on one of these schemes, I found that sometimes the method constructed ill-defined implicit functions. The main contribution of my work is to find conditions for when these implicit moving least squares schemes construct valid implicit functions. These schemes were devised as a way to interpolate and approximate non-manifold polyhedral data, although my analysis focuses on manifold curves and surfaces. I also looked at extending these methods to reduce the bulges while hilling holes.

1.1 Motivation

Polygonal meshes are one representation used in computer graphics. Numerous applications to generate and edit polygonal meshes have been developed in recent years. Some of these methods store polygonal mesh data in a "polygon soup", which is a group of unorganized triangles, with generally no explicit relationships between the triangles. Polygon soup uses an unorganized polygon mesh to save memory and read/write time. However, such meshes may have problems such as holes, gaps, T-junctions, self-intersections, and non-manifold structure. These defects make polygon soup unsuitable for a large variety of applications other than rendering, such as shape blending, deformation, constructive solid geometry operations, collision detection and so on. In contrast, implicit surfaces do not have these problems, and they have other desirable properties such as accuracy, conciseness, affine invariance, arbitrary topology, guaranteed continuity, and they allow efficient intersection computations. Its mesh-less and function based representation makes implicit

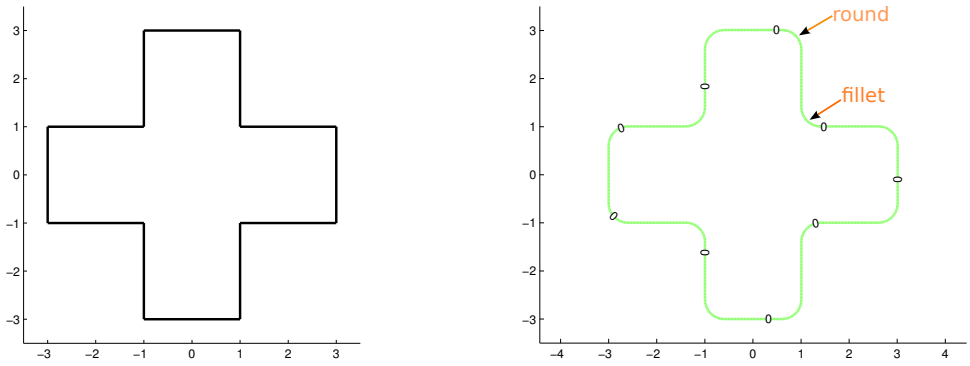
surface applicable to those polygon soup non-suitable applications. In addition, if we construct an implicit surface from polygonal soup, a clean and simplified polygonal model can be extracted from the reconstructed implicit surface to solve the above and other problems with polygon soup such as excessive detail or bad aspect-ratio polygons.

My work was driven by analyzing a method of Shen et al. [34]. Their method is called implicit moving least squares (IMLS) in this thesis. IMLS is a moving least squares technique to interpolate or approximate polygon soup, requiring integration over each polygon. In Shen's dissertation, he claimed that the method could fill holes and gaps but had no guarantee that the fill would satisfy any particular criteria. Further, Shen gave no clear example of holes or gaps that were filled with his method.

While objects are often modeled as polyhedral models, when manufactured, the edges between polygons need to be rounded (rounds), and the creases filled (fillets), as shown in Fig. 1.1. If Shen et al.'s method produces reasonable fills of holes, then we should be able to use their method for filleting and rounding. The idea is to take the polyhedral model, and shrink all the faces, leaving gaps along all the edges between polygons. We can then use Shen et al.'s method to interpolate the polygons, which will fill the gaps between the edges, giving us fillets and rounds. The main question is: do the fillets and rounds created by Shen et al.'s method have reasonable shape? To check whether his method gets ideal filleting and rounding results on polygon soup with holes or gaps, I did experiments on polygons and polyhedra with all the faces shrunk, creating holes and gaps. Fig. 1.2 shows a 2D example on a shrunken plus.

As seen in Fig. 1.2(c), Shen et al.'s method can fill gaps but they do not achieve an ideal result; in particular, their rounds "bulge". This problem is worse for larger holes. The method presented in my thesis modifies Shen et al.'s method by using an additional Gaussian weighting of the polygons to reduce the bulge problem. However, it suffers from sharp corners and edges, as shown in Fig. 1.2(d). On the other hand, when filling small holes, my method reduces the bulges that Shen et al.'s method has, as seen in Fig. 1.3 which shows 3D interpolation results from Shen et al.'s method in red, my method in green, and the approximation result from my method in bronze. I tried several other methods but none of them match the ideal result. Some of these methods have good results on 2D data, but fail on 3D.

I also noticed that Shen et al.'s method differs from standard moving least squares in that they square the weight function, but they give no explanation for this squaring. In addition, they were unable to derive a closed form of their integrals and used numerical methods to solve them. Park et al. [29] used Shen et al.'s method with a different weight function. Their weight function allowed them to derive an analytic solution to the polygon



(a) Original input.

(b) Fillets and rounds.

Figure 1.1: Ideal fillets and rounds example in 2D.

integrals.

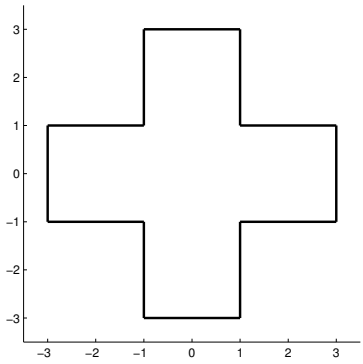
1.2 Contributions

In analyzing the weight functions of Shen et al. and of Park et al., I determined several things. First, I discovered that the squaring of the weight function in Shen et al.'s method is critical to its correctness of interpolating polygon soup, and in this thesis I show why. Second, for Shen et al.'s approximation scheme, the generalized weight function also needs to meet some necessary conditions to avoid degeneracies. Third, through the same analysis, I found that the weight function of Park et al. gives an incorrect result.

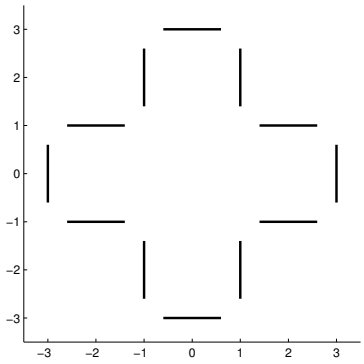
To summarize, the major technical contributions of this thesis include:

- I derive a lower-bound of the rate of decrease of the IDW (inverse distance weight) function. This lower-bound provides a criterion for choosing weight functions for MLS interpolation and approximation of closed polyhedron. In addition, for an IMLS function f, I show in which regions of space f is positive and in which regions f is negative.

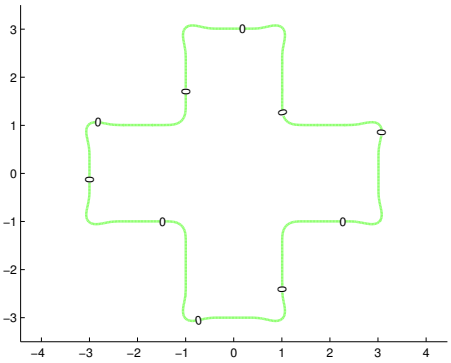
- I provide a generalized 2D polar coordinate formulation and 3D spherical coordinate



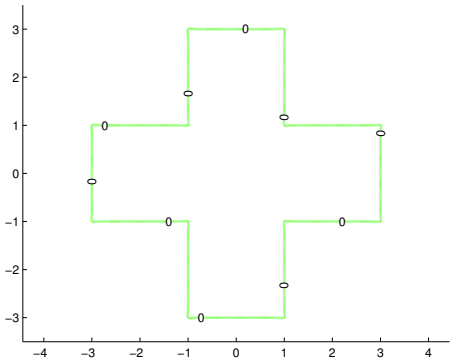
(a) Original input.



(b) Shrink all the faces.



(c) Shen et al.'s result.



(d) Result of my method.

Figure 1.2: Shen et al.'s fillets and rounds example in 2D.

formulation for IMLS. The interpolation solution proposed in Park et al. [29] can be seen as a special case of mine.

- To avoid bulges in Shen et al.'s filleting and rounding, I use a Gaussian distribution function as another weight function for polygons, which allows for controlling the sharpness when filling holes and gaps.

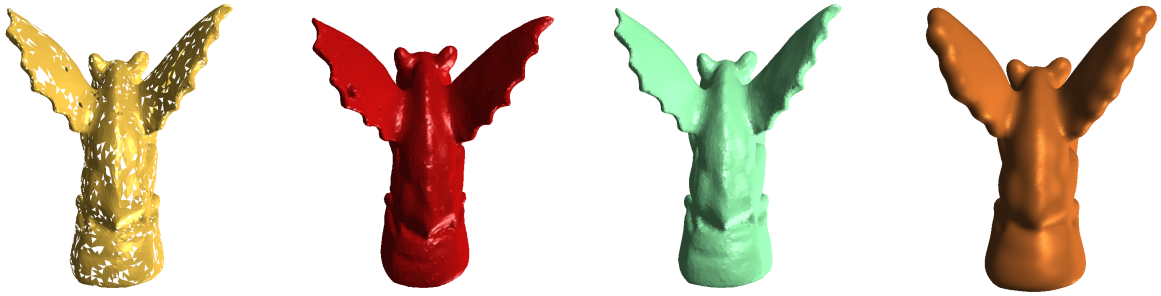


Figure 1.3: Illustration of the reconstruction from polygonal input data. Left: Input polygon soup model. Middle Left: Interpolation using Implicit Moving Least Squares [34]. Middle Right: Interpolation using proposed Generalized Implicit Moving Least Squares (GIMLS). Right: Approximation using GIMLS with $\epsilon = 0.02$ and $h = 0.08$. Note the bump on the left wing of the IMLS surface; the GIMLS surface does not have this artifact.

I also discuss using my method to approximate data in a manner similar to that of Shen et al. and to Park et al.

Chapter 2

Background

Scattered data approximation deals with the problem of reconstructing an unknown function from scattered data. Its many applications include terrain modeling, fluid-structure interaction, the numerical solution of partial differential equations, kernel learning, and parameter estimation, the most commonly used parameter estimation being the interpolation and approximation with implicit surfaces. My research is based on the interpolation and approximation with implicit surfaces as well. This chapter provides an introductory overview of implicit surface and scattered data approximation methods.

2.1 Implicit Surface and Parametric Surface

In computer graphics and geometry modeling, there are two common ways to specify a surface: implicit and parametric. Parametric representations usually define a surface as a set of points $\mathbf{p}(s,t)$, i.e.,

$$\mathbf{p}(s,t) = (x(s,t), y(s,t), z(s,t)).$$

Implicit representations typically define a surface as the zero contour of a function $F(\mathbf{p}) = 0$, i.e.,

$$F(\mathbf{p}) = F(x,y,z) = 0,$$

with the sign of F indicating whether points not on the surface are inside or outside the surface. Fig. 2.1 shows a simple implicit curve. We say that an implicit function is well defined if it is continuous, is negative in a bounded region, and is positive outside of a (possibly larger) bounded region.

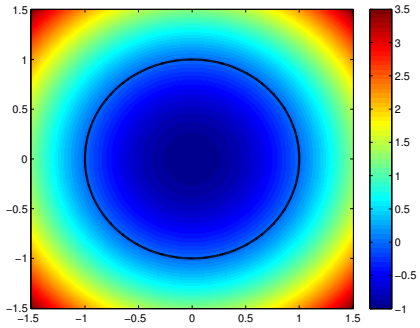


Figure 2.1: Implicit curve example: $f(\mathbf{p}) = \mathbf{p}_x^2 + \mathbf{p}_y^2 - 1$.

Both implicit and parametric surfaces are well developed in computer graphics. Traditionally, parametric representations have been favored because the parametric representation is simpler to render and is easy to use for many applications like the control of position or tangency or the computation of curvature or bounds. According to Rockwood [32], "parametric surfaces are generally easier to draw, tessellate, subdivide, and bound, or to perform any operation on that requires a knowledge of 'where' on the surface."

However, implicit surfaces are receiving increased attention due to their natural representation of solid objects and their innate blend properties. Bajaj [1] mentioned that "The ability to enclose volume and to represent blends of volumes endows implicit surfaces with inherent advantages in geometric design." In addition, skeletal implicit modeling techniques [9] make animation, shape change and deformation natural with implicit surface.

There are many other topics within the broad area of implicit surface, and the book by Bloomenthal and his co-authors [1] is a good reference.

2.2 Implicit Surface Reconstruction

In the past decades, reconstructing implicit surfaces from scattered data has been extensively explored [39]. The main stream of approaches includes signed distance estimation, Voronoi-based reconstruction, implicit surface fitting, and moving least squares surfaces [5]. However, little work has looked at interpolating polygonal data. My approach addresses the problem of implicit surface reconstruction from polygon soup, based on moving least squares.

Gois et al. [8] presented a method for reconstructing implicit surfaces from polygonal meshes based on Hermite-Birkhoff interpolation with radial basis functions. They had three independent sets of constraints: points, normals, and tangents. However, a lack of good criteria for automatic selection of these constraints made their method unstable. The requirement of solving large linear systems and susceptibility to noise are also problems with their method. There are also two hierarchical approximation methods dealing with polygonal data inspired by the multi-level partition of unity (MPU) idea proposed by Ohtake et al. [26] for point data sets. Kanai et al. [13] proposed a method to construct a hierarchy of a quadratic sparse low-degree implicit (SLIM) surfaces from a polygon mesh. As they build a set of implicit surfaces with different resolutions, their approach has large memory requirements. The quadratic polynomial implicit surface they used also makes the approach unstable when the polygonal mesh has a complicated shape. Li et al. [19] published a robust MPU method for triangle meshes. They used the dual graph of the triangle mesh to subdivide the triangle set into an Octree and they regarded each triangle as "a point with an error metric". This reduces the problem to that of a point cloud approximation MPU. However, their method applies only to triangle meshes and is hard to extend to more general polygonal data due to the design of their error metric. Xu et al. [42] recently published a paper dealing with polygon soup. They calculated signed distance fields for polygon soup meshes. To determine the sign they used a global analysis to determine the inside and outside instead of normals. They claimed that they could fill holes but showed no explicit results.

The most relevant research to this thesis is that of Shen et al. [34] and Park et al. [29]. The Shen et al. method is based on the MPU by Ohtake et al. [26]. IMLS and MPU are fundamentally alike but the integrated constraints of Shen et al. are different from the collections of point constraints of Ohtake et al.. Shen et al.'s method has problems with larger holes and gaps, and when used to approximate data, the resulting surface "shrinks" (i.e., the resulting surface is smaller than the data would suggest). The approximation scheme they proposed is inefficient as well. Kolluri et al. [16] revised Shen et al.'s IMLS method by using a Gaussian function as the weight function instead of Shen et al.'s inversed distance function. They proved that when applying their method on certain sampling conditions, the implicit surface generated is a good approximation of the signed distance function of the original surface. They also proved that their reconstructed surface is geometrically and topologically correct. In this thesis, I will analyze the weight function in IMLS of Shen et al. to give theoretical guarantees for their type of weight function. Shen et al. do not have closed form to their integral moving least squares equations and they use numerical method to approximate the result. Park et al. [29] claimed that they found an analytic solution for Shen et al. [34], but I will show that their method is in error, at least for

interpolating closed polyhedrons.

Another point based idea for filling holes related to my work is that of Öztireli et al. [28], who produced a new MLS based surface construction method based on a robust statistics technique to deal with holes and noise in a point data set. They have similar results to my Gaussian weighted implicit moving least squares method when filling holes. However, their iterative process to re-weight each implicit result is slow to converge.

Chapter 3

Moving Least Squares Basics

The fundamental method used in my research is moving least squares. It builds near-best approximations to functionals on $\mathbb{R}^d$ by using scattered-data information. In this chapter, standard least squares will be reviewed and compared to moving least squares.

3.1 Standard Least Squares

Standard least squares is an approximate solution for systems of linear equations when there are more equations than unknowns. "Least squares" means the solution minimizes the sum of the squares of the errors made in the results of every equation. This makes least squares applicable to curve fitting problems: the solution minimizes the sum of the squared residuals (ordinate differences) of the difference between an observed value and the fitted value.

The simplest form of least squares is to find the best fit straight line $y = ax + b$ for n points (data pairs) (x_i, y_i), $i \in \{1, \ldots, N\}$, where x_i is an independent variable and y_i is a dependent variable whose value is found by observation. It is unlikely that there is a perfect fit for two reasons. The first reason is experimental error; the second reason is that the underlying relationship of given pairs may not be linear. Thus the goal of least squares is to find a "best fit" to the data. The standard form of least squares fits an arbitrary degree polynomial to the data. For least squares applied to scattered data approximation, the problem is defined as follows.

Definition 3.1.1. For a set of points $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \in \mathbb{R}^{d \times N}$, a globally defined function $f(\mathbf{x})$ that approximates given scalar values f_i at points $\mathbf{p}_i$ in the least-squares

sense will be obtained by minimizing the error function

$$min \left\{ \sum_{i=1}^{N} [f(\mathbf{p}_i) - f_i]^2 : f \in \pi_M(R^d) \right\}, \tag{3.1}$$

where $f(\mathbf{p}_i)$ is a constraint on $\mathbf{p}_i$, and $\pi_M(R^d)$ is the set of polynomials f of degree at most M.

Let $\mathbf{b} = [b_1, \ldots, b_M] \in \mathbb{R}^M$ be a tuple of basis function and, let $\mathbf{c} = [c_1, \ldots, c_M] \in \mathbb{R}^M$ be the unknown coefficients of a polynomial f:

$$f = \sum_{j=1}^{M} c_j b_j. \tag{3.2}$$

The residual to be minimized with least squares is

$$R(\mathbf{c}) = \sum_{i=1}^{N} \left[f(\mathbf{p}_i) - \sum_{j=1}^{M} (c_j b_j(\mathbf{p}_i)) \right]^2. \tag{3.3}$$

The condition for $R(\mathbf{c})$ to be minimum is that

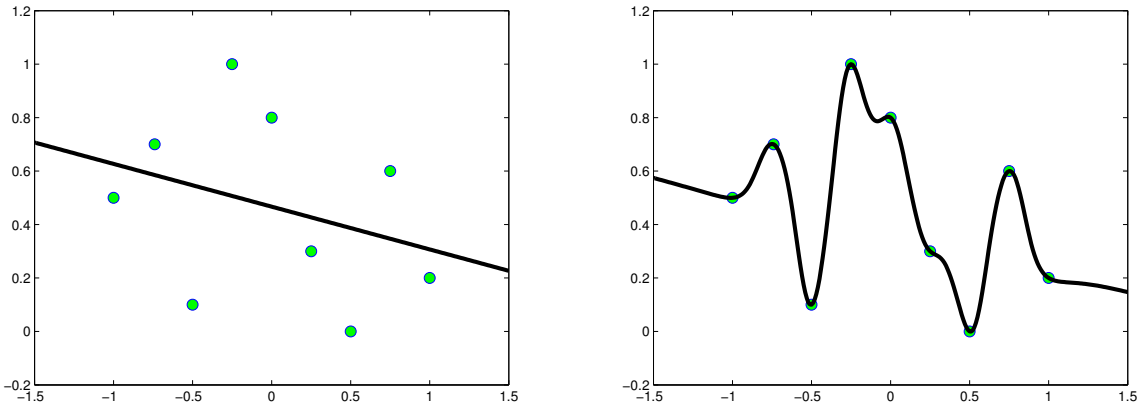
$$\frac{\partial R(\mathbf{c})}{\partial c_i} = 0$$

for $i = 1, \ldots, M$. For a polynomial of M degree fit, we need to solve a $M \times M$ linear system. Standard least squares approximates the data globally by balancing between all of the input data. For all the query points, the resulting approximant is constant.

Unfortunately, for complicated fitting problems, a satisfactory global fit requires a high degree polynomial. Apart from the difficulty of solving large linear system to specify such a polynomial, the resulting surface or curve will almost always be folded and mountainous. An approach with local polynomial reproduction named moving least squares will address these problems. Fig. 3.1 shows the plot of least squares approximation comparing to the plot of moving least squares interpolation. Both are with a linear basis function.

3.2 Moving Least Squares

The moving least squares method for smoothing and interpolating scattered data was proposed by Lancaster and Salkauskas [18] in 1981. The idea of moving least squares



(a) Least Squares approximation.

(b) Moving least squares approximation.

Figure 3.1: Least squares and moving least squares comparison. Moving least squares uses weight function in Eq. 3.13 and $\epsilon = 0$. Both methods are with a linear basis function.

is to solve for every point $\mathbf{x}$ a locally weighted least square problem. This appears to be expensive at first sight, but it will turn out to be efficient. The approximation is obtained by solving many small linear systems, instead of solving a single large linear system as done by least squares. Moreover, in many applications one is only interested in a few evaluations instead of the whole domain of the input data. For such applications, the moving least squares approximation is even more attractive. The influence of the data points is governed by a weight function w: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, which becomes smaller the further away its arguments are from each other. Ideally, w vanishes for arguments $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ with their distance $d(\mathbf{x}, \mathbf{y})$ greater than a given threshold. Such behavior can be modeled by using a translation-invariant weight function.

The standard moving least squares is defined as follows [39]:

Definition 3.2.1. For a set of points $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \in \mathbb{R}^{d \times N}$, and a query point $\mathbf{x} \in \mathbb{R}^d$, the value $s_{f,X}(\mathbf{x})$ of the moving least squares approximant is given by $s_{f,X}(\mathbf{x}) = p^*(\mathbf{x})$ where $p^*(x)$ is the solution to

$$\min \left\{ \sum_{i=1}^{N} \left[f(\mathbf{x}_i) - p(\mathbf{p}_i) \right]^2 w(\mathbf{x} - \mathbf{p}_i) : p \in \pi_M(R^d) \right\}, \tag{3.4}$$

where $f(\mathbf{p}_i)$ is the constraint on $\mathbf{p}_i$, and $\pi_M(\mathbb{R}^d)$ is the set of polynomials p of degree at most M.

According to Wendland [39], the polynomial p can be defined in terms of basis functions $\mathbf{b} = [b_1, \ldots, b_M] \in \mathbb{R}^M$ and adjusting the coefficients $\mathbf{c} = [c_1, \ldots, c_M] \in \mathbb{R}^M$ as

$$p = \sum_{j=1}^{M} c_j b_j. \tag{3.5}$$

This reduces the minimization problem to finding the optimal coefficient vector $\mathbf{c}$. Combining Eq. 3.4 and Eq. 3.5, we need to minimize the following:

$$R(\mathbf{c}) = \sum_{i=1}^{N} \left[f(\mathbf{p}_i) - \sum_{j=1}^{M} (c_j b_j(\mathbf{p}_i)) \right]^2 w(\mathbf{x} - \mathbf{p}_i). \tag{3.6}$$

Using the notation

$$\begin{aligned} \mathbf{B} &= [b^\intercal(\mathbf{p}_1), \ldots, b^\intercal(\mathbf{p}_N)]^\intercal \in R^{N \times M} \\ \mathbf{W} &= \operatorname{diag}\left(w(\mathbf{x} - \mathbf{p}_i) : i \in [0, N]\right) \\ \phi &= (f(\mathbf{p}_1), \ldots, f(\mathbf{p}_N))^\intercal \in R^N, \end{aligned}$$

the matrix formulation of Eq. 3.6 is

$$\begin{aligned} R(\mathbf{c}) &= \sum_{i=1}^{N} (\phi_i - \mathbf{B}_i \mathbf{c})^2 w(\mathbf{x} - \mathbf{p}_i) \\ &= (\phi - \mathbf{B}\mathbf{c})^\intercal \mathbf{W} (\phi - \mathbf{B}\mathbf{c}) \\ &= (\phi^\intercal \mathbf{W} \phi) - 2\mathbf{c}^\intercal \mathbf{B}^\intercal \mathbf{W} \phi + \mathbf{c}^\intercal \mathbf{B}^\intercal \mathbf{W} \mathbf{B} \mathbf{c}. \end{aligned}$$

Here $R(\mathbf{c})$ is a quadratic function in $\mathbf{c}$, and it can be shown that $\mathbf{B}^\intercal\mathbf{W}\mathbf{B}$ is positive semi-definite and that the minimization problem has a unique solution [39]. Because there is a unique solution we can use the necessary condition $\nabla R(\mathbf{c}) = 0$ to find this solution, i.e.,

$$\nabla R(\mathbf{c}) = -2\mathbf{B}^\intercal \mathbf{W} \phi + 2\mathbf{B}^\intercal \mathbf{W} \mathbf{B} \mathbf{c} = 0 \Leftrightarrow \mathbf{B}^\intercal \mathbf{W} \mathbf{B} \mathbf{c} = \mathbf{B}^\intercal \mathbf{W} \phi, \tag{3.7}$$

the solution of which is $\mathbf{c}^\intercal = (\mathbf{B}^\intercal \mathbf{W} \mathbf{B})^{-1} \mathbf{B}^\intercal \mathbf{W} \phi$.

Shen et al. [34] took a non-standard approach to MLS in their IMLS method. They changed the matrix form to

$$\begin{bmatrix} w(\mathbf{x} - \mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x} - \mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \mathbf{b}^\intercal(\mathbf{p}_1) \\ \vdots \\ \mathbf{b}^\intercal(\mathbf{p}_N) \end{bmatrix} \mathbf{c} = \begin{bmatrix} w(\mathbf{x} - \mathbf{p}_1) & & \\ & \ddots & \\ & & w(\mathbf{x} - \mathbf{p}_N) \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}, \tag{3.8}$$

which is

$$\mathbf{WBc} = \mathbf{W}\phi. \tag{3.9}$$

Compared to Eq. 3.7, Shen et al. eliminate $\mathbf{B}^\intercal$ directly. To solve Eq. 3.9, they multiplied Eq. 3.9 by $(\mathbf{WB})^\intercal$, giving

$$(\mathbf{WB})^\intercal\mathbf{WBc} = (\mathbf{WB})^\intercal\mathbf{W}\phi \tag{3.10}$$

$$\mathbf{B}^\intercal\mathbf{W}^2\mathbf{Bc} = \mathbf{B}^\intercal\mathbf{W}^2\phi. \tag{3.11}$$

The difference between the standard derivation and Shen's derivation is only in the weight function. Shen et al. gave no explicit reason why squaring the weight is necessary for surface reconstruction especially as they were free to choose the weight function. Following Shen et al. [34], constant basis functions are used for reconstruction, i.e., when $p(\mathbf{x}_i)$ is constant the implicit function that can be used with these methods is

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{N} w(\mathbf{x} - \mathbf{p}_i)\phi_i}{\sum_{i=1}^{N} w(\mathbf{x} - \mathbf{p}_i)}. \tag{3.12}$$

This implicit function is essentially weighting the constraints from the surfaces. Therefore, the weight function becomes an important factor in this construction. If $w(\mathbf{x} - \mathbf{p}_i)$ is the distance from the scattered point to the evaluation point, Eq. 3.12 will be the simplest form of inverse distance weighted reconstruction, called Sheppard's method [35, 39]. Shen et al. used the weight

$$w(r) = \frac{1}{r^2 + \epsilon^2}. \tag{3.13}$$

Since they squared the weight function, I will use the following weight function

$$w(r) = \frac{1}{(r^2 + \epsilon^2)^2} \tag{3.14}$$

in the standard MLS formulation. However, in Section 5.2, I will show that if Shen et al. did not square the weight function, the MLS solution would be degenerate.

When $\epsilon = 0$, the moving least-squares function will interpolate the sample points (constraint values).When ϵ is non-zero, the moving least-squares function only approximates the constraint values. By adjusting ϵ, a variety of approximating behaviors can be achieved. A larger ϵ tends to provide a smoother approximation while a smaller ϵ gives a closer approximation to the original points.

A generalized inverse distance function is

$$w(r) = \frac{1}{(r^2 + \epsilon^2)^d}. \tag{3.15}$$

where d is a positive integer. Greater values of d assign greater influence to values closest to the interpolated point. In Chapter 5, I will use this generalized weight function to prove that d has to meet some conditions to generate well-defined interpolation and approximation implicit surfaces.

Chapter 4

Implicit Moving Least Squares

In this chapter, I review implicit moving least squares and the analytic solution to implicit moving least squares (IMLS) proposed by Park et al. I also show that their solution has degeneracy problems.

4.1 Implicit Moving Least Squares

The moving least squares method for implicit surface reconstruction was originally applied to scattered point data approximation. Shen et al. [34] modified moving least squares to interpolate and approximate polygon soup. The major differences between Shen et al.'s implicit moving least square and standard moving least squares are:

- Shen et al.'s IMLS uses a shape function to avoid trivial results,
- Shen et al.'s IMLS integrates the constraints over each polygon to avoid oscillations in the resulting surface.

4.1.1 True-Normal Shape Function

If the data to interpolate are all 0, then there is a trivial solution of $c_j = 0$ for all j in Eq. 3.5. To avoid this trivial solution, we need points with positive/negative constraints (i.e., points not on the surface). Shen et al.'s IMLS used a shape function $S(\mathbf{x})$ to incorporate normal constraints,

$$S(\mathbf{x}) = \phi + (\mathbf{x} - \mathbf{p})^\intercal \mathbf{n}, \tag{4.1}$$

where $\mathbf{x}$ is the query point, $\mathbf{p}$ is the point on a polygon, $\mathbf{n}$ is the normal of the polygon, and ϕ is the implicit value on $\mathbf{x}$, which is normally 0. Shen et al. call $S(\mathbf{x})$ a *true-normal* shape function to contrast it with Turk and O'Brien's [38] pseudo-normal constraints which places zero constraints at points on the surface, positive constraints offset sightly inside the surface, and negative ones slightly outside. However, Turk and O'Brien's additional constraints influence the function's gradient crudely, while the true-normal shape function of Shen et al. force the surface to interpolate the normal. A detailed discussion of these two normal constraints can be found in Chapter 4 of Shen's dissertation [33] and in a technical report [14]. Martin et al. [21] used a similar shape function in their point-based moving least squares method.

4.1.2 Integrated Moving Least Squares

When sampling finite points on a polygon for interpolation, the resulting implicit surface has bumps and dimples as shown in Fig. 4.1(a). Increasing the sample density will make the problem less severe, but the problem is still not solved, shown in Fig. 4.1(b). To achieve good results, what we would like to do is to scatter an infinite number of points continuously across the surface of each polygon. To meet this requirement, Shen et al. integrates the constraints over each element (line segment in 2D or polygon in 3D). For a data set of K elements, let Ω_k, $k \in [1, \ldots, K]$ be the domain for each of K input elements. Eq. 3.12 becomes

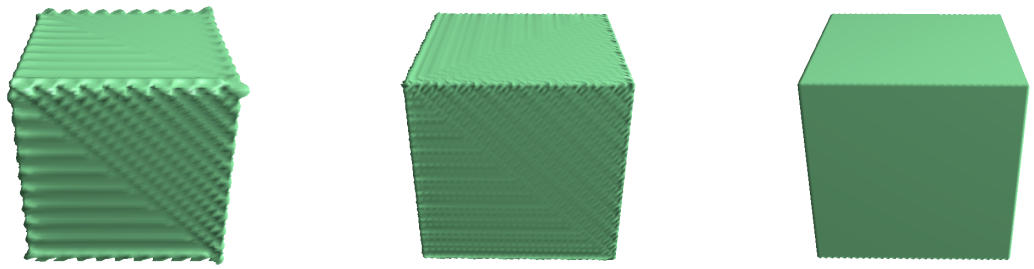
$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) S_k(\mathbf{x}) d\mathbf{p}_{\Omega_k}}{\sum_{k=1}^{K} \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega k}}, \tag{4.2}$$

where $\mathbf{p}_{\Omega_k}$ is a point on Ω_k, and $S_k(\mathbf{x}) = \phi + (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k}$ is defined in Section. 4.1.1. The comparison of a finitely sampled cube surface and integrated cube surface is shown in Fig. 4.1.

Shen et al. gave an analytic solution of their IMLS for 2D. In 3D, Eq. 4.2 expands to a double integral. But Shen et al. only provided inner layer (analytical line-integration) solutions, with the outer layer (final area integration) being calculated as a numerical integration of the analytical line-integration solutions for each polygon. Park et al. [29] proposed an analytic solution for both 2D and 3D of a variation of Shen et al.'s IMLS. They gave a general representation of a weight function in Eq. 3.14 with $\epsilon = 0$ such that

$$w_n(\mathbf{x} - \mathbf{p}_k) = \left(\frac{1}{\|\mathbf{x} - \mathbf{p}_k\|^2} \right)^n. \tag{4.3}$$

Their closed form solution is limited to $n = 1$ for 2D interpolation and $n = 3/2$ for 3D interpolation of Eq. 4.3, which is applied in Eq. 4.2. However, when I implemented Park



(a) 100 points on each face (b) 400 points on each face (c) Integrated over each face.

Figure 4.1: Comparison of finite sampled cube and integrated cube surface.

et al.'s scheme, I found that it produces degenerate results as shown in Fig. 4.2. While the resulting implicit function should only be zero at points on the input polygons, with their method the implicit values of all points outside of these shapes are zero. Their method has the same problem in 3D (that the implicit values of all points outside of the input closed polygonal are zero). In next section, I will discuss the 2D and 3D cases in Park et al.'s

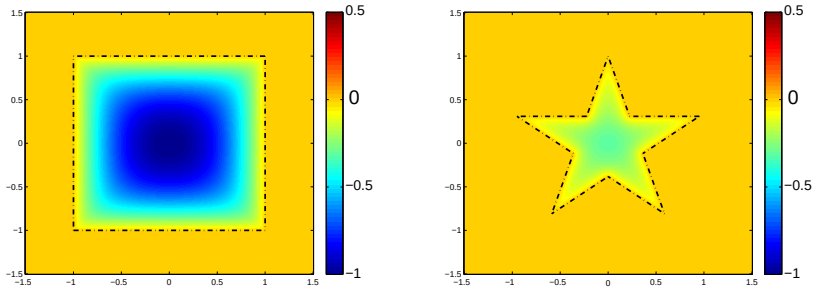


Figure 4.2: 2D example of implicit surface constructed by Park et al.'s method Implicit value is zero outside the square (left) and outside the star (right).

solution and show the reason for these degeneracies.

4.2 Analyzing 2D and 3D Interpolation of Park et al.'s Method

In this section, I will use Park et al.'s derivation and weight function to show why their method is degenerate in both 2D and 3D when $\epsilon = 0$ (i.e., when their method interpolates the data).

4.2.1 2D Interpolation

In this section, I present Park et al.'s derivation of their method, and in the next section, I will use their formulation to show that their method does not construct well-defined implicit functions when interpolating data. In 2D, Park el al. found a closed form for $n = 1$ in the weight function in Eq. 4.3. The normal function $S_k(\mathbf{x})$ in Eq. 4.2 equals the distance from $\mathbf{x}$ to the tangent line of line segment L_k in the input polygonal curve. So $S_k(\mathbf{x})$ is constant within the k-th segment and it can be moved outside the integral. Thus in 2D, for an input polygonal curve defined by the set of line segments $\mathbf{L} = \{L_k\}, k = 1 \dots K$, when $n = 1$ in the weight function in Eq. 4.3, Eq. 4.2 becomes

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} S_k(\mathbf{x}) \int_{L_k} \frac{1}{\|\mathbf{x}-\mathbf{p}_{L_k}\|^2} d\mathbf{p}_{L_k}}{\sum_{k=1}^{K} \int_{L_k} \frac{1}{\|\mathbf{x}-\mathbf{p}_{L_k}\|^2} d\mathbf{p}_{L_k}} = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k} \tag{4.4}$$

where

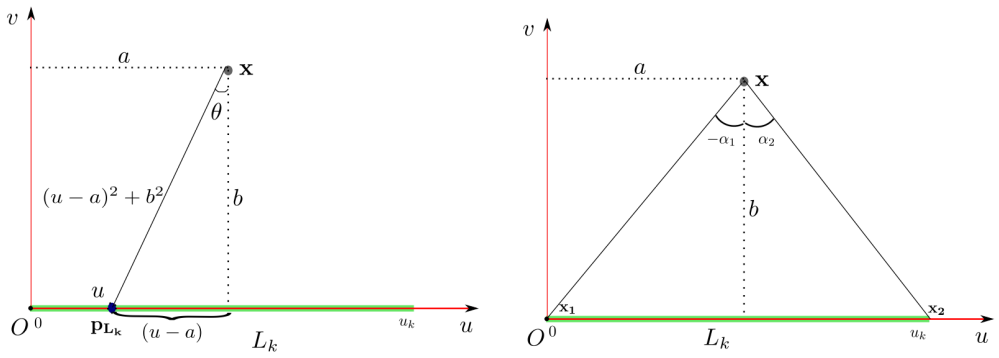
$$A_k = \int_{L_k} \frac{1}{\|\mathbf{x} - \mathbf{p}_{L_k}\|^2} d\mathbf{p}_{L_k} \tag{4.5}$$

$$a_k = S_k(\mathbf{x}) A_k. \tag{4.6}$$

The problem of solving Eq. 4.2 simplifies to finding A_k and a_k. To find a closed form, Park et al. re-expressed A_k on a u, v-space where the u-axis is along the line L_k with positive direction the same as the clockwise direction of the polygon, and the v-axis is along the normal direction to L_k. The most anti-clockwise ending point is set as the origin, as shown in Fig. 4.3, where the green line is the input line segment. Then

$$A_k = \int_{\mathbf{x}_1}^{\mathbf{x}_2} \frac{1}{\|\mathbf{x} - \mathbf{p}_{\mathbf{L_k}}\|^2} d\mathbf{p}_{\mathbf{L_k}} \tag{4.7}$$

$$= \frac{1}{b^2} \int_0^{u_k} \frac{1}{(u-a)^2/b^2 + 1} du, \tag{4.8}$$



(a) (b)

Figure 4.3: Diagrammatic sketch of the transformed u, v space

where a, b are the coordinates of the query point $\mathbf{x}$ in the u, v space, as shown in Fig. 4.3(a). $\mathbf{x}_1$ and $\mathbf{x}_2$ are the starting and ending points of line segment L_k, and they are mapped to 0 and u_k in u, v space, shown in Fig. 4.3(b). Let θ be the angle from the normal of L_k to vector $(\mathbf{x} - \mathbf{p}_{L_k})$ shown in Fig. 4.3(a), then

$$\tan\theta = (u-a)/b. \tag{4.9}$$

Differentiating both sides of Eq. 4.9, we obtain

$$d\theta \sec^2\theta = du/b. \tag{4.10}$$

Using Eq. 4.9 and Eq. 4.10 in Eq. 4.8, we have

$$\begin{aligned} A_k &= \frac{1}{b^2}\int_{-\alpha_1}^{\alpha_2} \frac{b\sec^2\theta}{\tan^2\theta+1} d\theta \\ &= \frac{1}{b}\int_{-\alpha_1}^{\alpha_2} d\theta = \frac{(\alpha_2+\alpha_1)}{b}, \end{aligned}$$

where

$$\alpha_1 = \arctan(a/b), \quad \alpha_2 = \arctan(\frac{u_k - a}{b}). \tag{4.11}$$

Since $(\mathbf{x} - \mathbf{p}_{L_k})^\intercal \mathbf{n}_{L_k}$ is the distance from $\mathbf{x}$ to the tangent line of L_k, $S_k(\mathbf{x}) = b_k + \phi$. ϕ is zero as it is the constraint value of points on the surface. So $S_k(\mathbf{x}) = b_k$. Let α' be the

angle at point $\mathbf{x}$ of the triangle constructed using $\mathbf{x}$ and the two end points of L_k. From Fig. 4.3(b),

$$\alpha'_k = \alpha_1 + \alpha_2, \tag{4.12}$$

then

$$\begin{aligned} A_k &= (\alpha_1 + \alpha_2)/b_k = \alpha'_k/b_k \\ a_k &= A_k S_k(\mathbf{x}) = (\alpha'_k/b_k)b_k = \alpha'_k. \end{aligned}$$

Thus, the implicit function becomes

$$f(x) = \frac{\sum_{k=1}^{K} \alpha'_k}{\sum_{k=1}^{K} \alpha'_k/b_k},$$

which matches the result of Park et al..

4.2.2 2D Degeneracies in Park et al.'s Method

Fig. 4.4 shows why Park et al.'s method has zero implicit values for query points on the exterior of the input square. The corresponding angles α_1 and α_2 in Eq. 4.11 in the above 2D analytic solution for the square in Fig. 4.2 are marked in Fig. 4.4 for each line segment. For the evaluation on each line segment, the plus or minus sign of each angle is determined by the new coordinate of a and b of query point $\mathbf{p}_1$ on the new u, v-axis. I marked the sign of each a and b for each line segment, and also the sign of each α'_k. It can be seen that

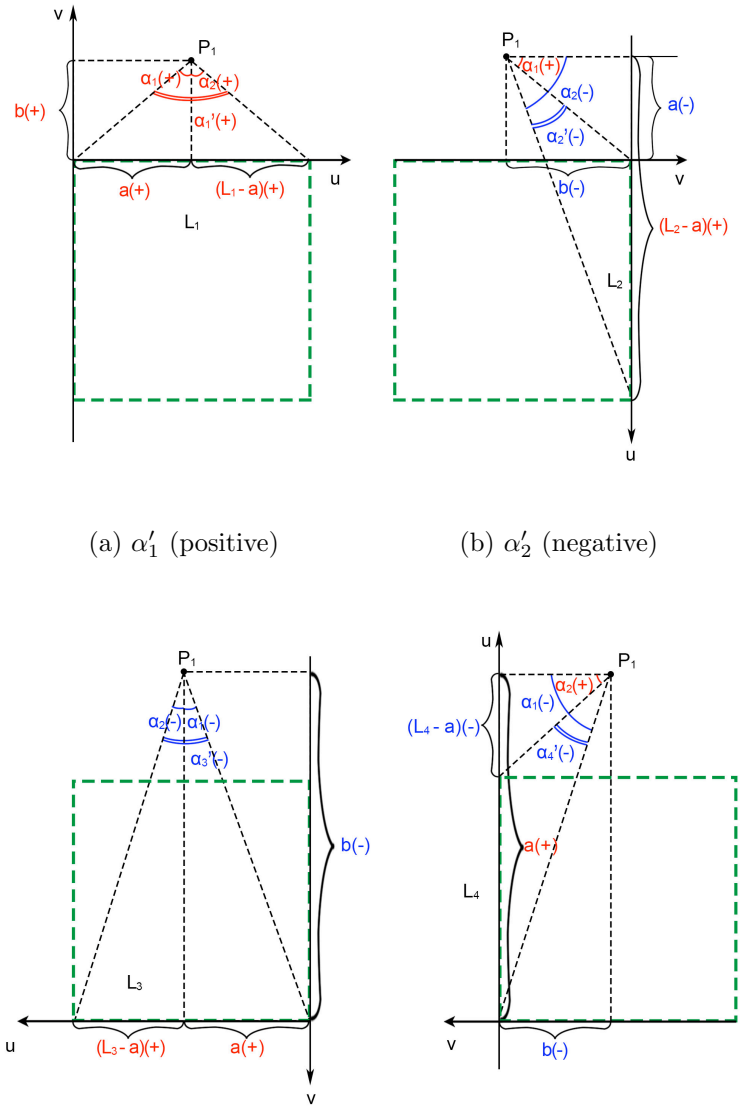
$$\|\alpha'_1\| = \|\alpha'_2\| + \|\alpha'_3\| + \|\alpha'_4\|$$

and only α'_1 is positive, while $\alpha'_2, \alpha'_3, \alpha'_4$ are negative. This results in $f(\mathbf{x}) = 0$ when the query point is on the exterior of square. This confirms that $f(\mathbf{x}) = 0$ for the square in the experiment shown in Fig. 4.2.

To generalize the proof of degeneracies to any closed polygons, I project any point on the polygon $\mathbf{L}$ to a unit circle centered at query point $\mathbf{x}$. According to Eq. 4.12,

$$|\alpha'_k| = |\alpha_1 + \alpha_2|.$$

Without loss of generality, I define anti-clockwise to be the positive direction of the polygon and each line L_k is oriented in its positive direction, starting from $\mathbf{p}_k^s$ and ending at $\mathbf{p}_k^e$. I project any point $\mathbf{p}$ on the polygon to point $P(\mathbf{p})$ lying on the unit circle centered at



(a) α_1' (positive) (b) α_2' (negative)

(c) α_3' (negative) (d) α_4' (negative)

Figure 4.4: 2D case illustration showing that Park et al.'s method is 0 on a point exterior to the polygon. Green dotted lines are input data and P_1 is the query point. Negative values are colored blue and positive values are colored red. The sum of the four α's are 0.

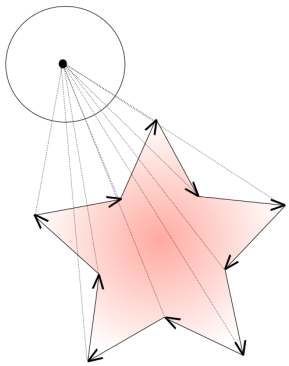


Figure 4.5: Polygon to Unit circle centered at query point $\mathbf{x}$

$\mathbf{x}$. The projection $P(L_k)$ of line segment L_k is an arc starting from $P(\mathbf{p}_k^s)$ and ending at $P(\mathbf{p}_k^e)$. Then α'_k equals the integral along the circle such that

$$\alpha'_k = \int_{P(L_k)} d\mathbf{p}.$$

Since the projection $P(\mathbf{L})$ of the a closed polygon $\mathbf{L}$ is continuous and ending at the starting point, we can conclude that

$$\sum_{k=1}^{K} \alpha'_k = \int_{P(\mathbf{L})} d\mathbf{p} = 0$$

and $f(\mathbf{x}) = 0$ for any $\mathbf{x}$ on the exterior of the polygon. Fig. 4.5 shows an example for a star polygon. In Section 5.2.2, I will show that this integral is zero for any closed polygon only when $n = 1$ in Eq. 4.3.

4.2.3 3D Interpolation

For 3D interpolation, Park el al. derived a closed form for $n = 3/2$ in the weight function in Eq. 4.3. For an input polyhedral surface defined by the set of polygons $\mathbf{\Omega} = \{\Omega_k\}, k = 1 \dots N$, when $n = 3/2$ in the weight function Eq. 4.3, Eq. 4.2 becomes

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} S_k(\mathbf{x}) \int_{\Omega_k} \frac{1}{\|\mathbf{x}-\mathbf{p}_{\Omega_k}\|^3} d\mathbf{p}_{\Omega_k}}{\sum_{k=1}^{K} \int_{\Omega_k} \frac{1}{\|\mathbf{x}-\mathbf{p}_{\Omega_k}\|^3} d\mathbf{p}_{\Omega_k}} = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k} \tag{4.13}$$

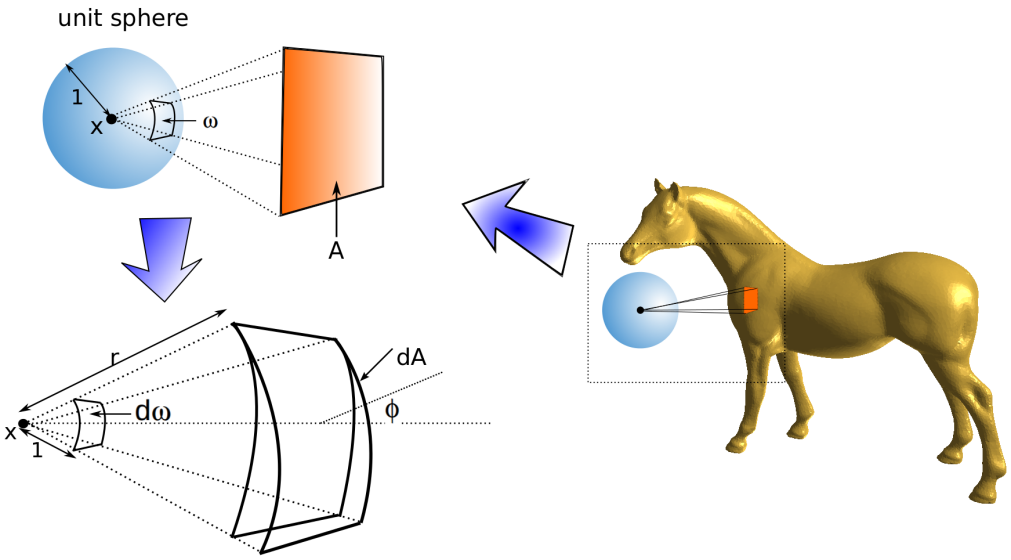


Figure 4.6: Calculation of Solid Angle. The left two illustrations are revised from illustrations in Long [20]

where

$$A_k = \int_{L_k} \frac{1}{\|\mathbf{x} - \mathbf{p}_{L_k}\|^3} d\mathbf{p}_{L_k} \tag{4.14}$$

$$a_k = S_k(\mathbf{x}) A_k. \tag{4.15}$$

Park et al. [29] solved for A_k, a_k using solid angles. In Fig. 4.6, the solid angle subtended by area A at point $\mathbf{x}$ is measured by the area ω on the surface of the unit sphere centered at $\mathbf{x}$. An infinitesimal area dA is from A, $d\omega$ is the solid angle subtended by dA at a point $\mathbf{x}$. ϕ is the angle between the normal to the surface and the line connecting the surface and point $\mathbf{x}$.

Treating A as an input polygon Ω_k, the solid angle $d\omega_k$ subtended by $d\mathbf{p}_{\Omega_k}$ is

$$d\omega_k = \frac{\cos\phi}{r^2} d\mathbf{p}_{\Omega_k} \Leftrightarrow \omega_k = \int_{\Omega_k} \frac{\cos\phi}{r^2} d\mathbf{p}_{\Omega_k} \tag{4.16}$$

where Ω_k here stands for the kth input polygon (see Long [20] for detail). As

$$\cos\phi = \mathbf{n}_{\Omega_k} \cdot (\mathbf{x} - \mathbf{p}_{\Omega_k}) / \|\mathbf{x} - \mathbf{p}_{\Omega_k}\| \tag{4.17}$$

and

$$r = \|\mathbf{x} - \mathbf{p}_{\Omega_k}\|, \tag{4.18}$$

ω_k for the input polygon Ω_k is now

$$\omega_k = \int_{\Omega_k} \frac{\mathbf{n}_{\mathbf{\Omega_k}} \cdot (\mathbf{x} - \mathbf{p}_{\Omega_k})}{\|\mathbf{x} - \mathbf{p}_{\Omega_k}\|^3} d\mathbf{p}_{\Omega_k}.$$

Since $S_k(\mathbf{x}) = \mathbf{n}_{\mathbf{\Omega_k}} \cdot (\mathbf{x} - \mathbf{p}_{\Omega_k})$, the solutions of the implicit surface are given by $a_k = \omega_k$ and $A_k = \omega_k / S_k(\mathbf{x})$ and so

$$f(x) = \frac{\sum_{k=1}^{K} \omega_k}{\sum_{k=1}^{K} \omega_k / S_k(\mathbf{x})}.$$

4.2.4 3D Degeneracies in Park et al.'s Method

In 3D, we can map the input polygons to a unit sphere centered at the query point $\mathbf{x}$ and prove in a similar way as 2D interpolation that $f(\mathbf{x}) = 0$ for $\mathbf{x}$ in the exterior of the polyhedron. Park et al. re-expressed f as

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} \omega_k}{\sum_{k=1}^{K} \omega_k / S_k(\mathbf{x})}, \tag{4.19}$$

where ω_k is the solid angle subtended by Ω_k in the polyhedron $\mathbf{\Omega}$. The area of ω_k is cut out by lines drawn from query point $\mathbf{x}$ to every point on the periphery of Ω_k. An example of the solid angles on the unit sphere projected from the polyhedron is in Fig. 4.7. From Eq. 4.17 and Eq. 4.16, we have

$$\omega_k = \int_{\Omega_k} \frac{\cos\phi}{r^2} d\mathbf{p}_{\Omega_k} \quad \text{and} \tag{4.20}$$

$$\cos\phi = \mathbf{n}_{\Omega_k} \cdot (\mathbf{x} - \mathbf{p}_{\Omega_k}) / \|\mathbf{x} - \mathbf{p}_{\Omega_k}\|. \tag{4.21}$$

Note that the sign of ω_k is the sign of $\cos\phi$. Normally for each polygon on the polyhedron, the vertices are in anti-clockwise order and the polygon's normal points outward. We can project any point $\mathbf{p}$ on the polyhedron to the point $P(\mathbf{p})$ lying on the unit sphere centered at $\mathbf{x}$. The projection $P(\Omega_k)$ of polygon Ω_k is a patch on the sphere contains all the projected points $\mathbf{p}_{\Omega_k}$, and ω_k equals the integral over the patch such that

$$\omega_k = \iint_{P(\Omega_k)} d\mathbf{p}. \tag{4.22}$$

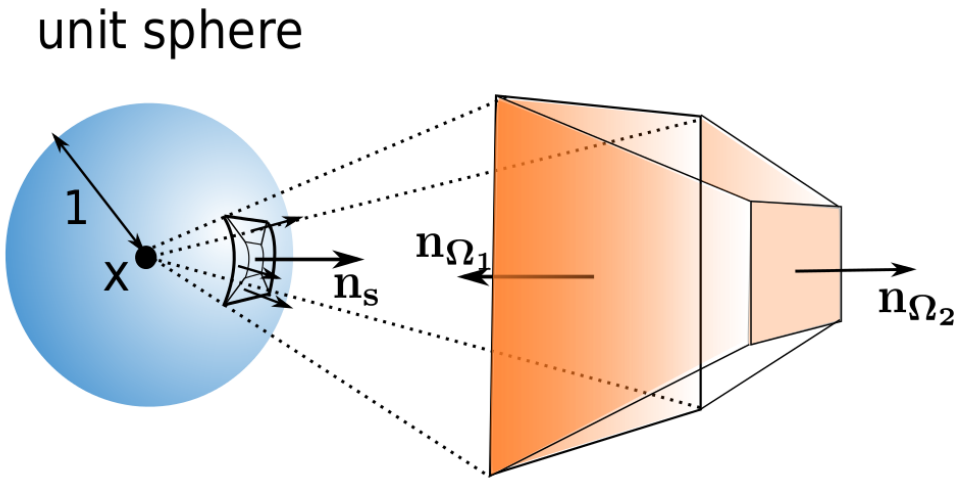


Figure 4.7: Example of polyhedron projected to unit sphere.

If the projected vertices on $P(\Omega_k)$ are in anti-clockwise order, $\omega_k > 0$; if they are in clockwise order, $\omega_k < 0$; and if the vertices project to a line or a point on the sphere, then $\omega_k = 0$. The projection $P(\mathbf{\Omega})$ of a closed polygon $\mathbf{\Omega}$ is closed in the projection space, hence

$$\sum_{k=1}^{K} \omega_k = \iint_{P(\mathbf{\Omega})} d\mathbf{p} = 0 \ . \tag{4.23}$$

Therefore, $f(\mathbf{x}) = 0$ for any $\mathbf{x}$ on the exterior of the polygon.

From the above analysis, I noticed that carefully selecting a good weight is of central importance to the implicit moving least squares problem. A bad choice of the weight function may cause serious degeneracy problems. This observation motivated my work to take a more theoretical analysis of the weight functions.

Chapter 5

Analysis of the IMLS Weight Function

From the previous chapter, we know that a good choice of the weight function is essential to the correctness of IMLS interpolation results. For IMLS approximation, Shen et al. used a weight function

$$w(\mathbf{x}-\mathbf{p}) = \left(\frac{1}{\|\mathbf{x}-\mathbf{p}\|^2+\epsilon^2}\right)^2 \tag{5.1}$$

with $\epsilon \neq 0$. When I implemented Shen et al.'s approximation weight function as

$$w(\mathbf{x}-\mathbf{p}) = \frac{1}{\|\mathbf{x}-\mathbf{p}\|^2+\epsilon^2}, \tag{5.2}$$

I found the result is not a well-defined implicit function; an example is shown in Fig. 5.1. These degeneracy problems with Park et al. scheme raise the question of when and if the implicit functions constructed by Shen et al. and Park et al. well defined. In this chapter, I will derive conditions for these schemes to be well defined.

From my experiments with Park et al.'s scheme, it is clear that the weight function must meet some conditions for f to be well-defined. In particular, I will further analyze the weight function

$$w_n(\mathbf{x}-\mathbf{p}) = \left(\frac{1}{\|\mathbf{x}-\mathbf{p}\|^2+\epsilon^2}\right)^n \tag{5.3}$$

for both IMLS interpolation and approximation, and prove necessary conditions on the choice weight functions to avoid degeneracy problems in IMLS interpolation and approximation.

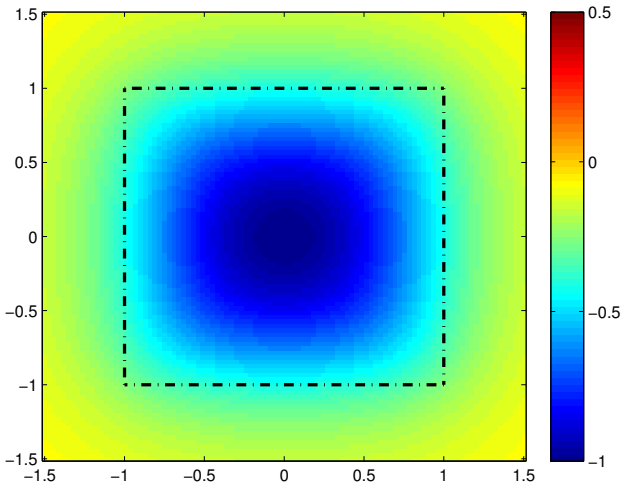


Figure 5.1: Approximating cube with $n = 1$ and $\epsilon = 0.3$.

To start with, I will reformulate the IMLS implicit function by using 2D polar coordinates and 3D spherical coordinates. Inspired by the new forms that are integrated over polar directions in 2D and spherical directions in 3D on each segments, I cast rays from the query point to intersect each ray with the polygonal curve/polyhedral surface, and use these intersections to compute the sign of the integrand in the direction of the ray, and integrate over all the rays. I will then determine the sign of implicit curve or surface functions by analyzing the integrands of the new integrations over rays. For my analysis, I only consider closed polygonal curves and polyhedral surfaces that do not intersect with themselves and I prove the 2D and 3D cases separately.

5.1 Technical Basics

In this section, I will present two lemmas that will be used in the proofs of both 2D and 3D theorems. The t in the following will be used as the dimension (2D or 3D) when we use these lemmas in later sections.

Lemma 5.1.1. Let $\mathbf{R} = \{r_1, \ldots, r_N\} \in \mathbb{R}^N$, and $r_i < r_j$ for $i < j$. For N is even, $t > 0$

and $\epsilon, n \in \mathbb{R}$, define

$$I = \sum_{i=1}^{N} (-1)^{i+1} \frac{r_i^t}{(r_i^2 + \epsilon^2)^n} \tag{5.4}$$

When $\epsilon = 0$:

- If $n < t/2$, then $I < 0$;
- If $n = t/2$, then $I = 0$;
- If $n > t/2$, then $I > 0$.

When $\epsilon \neq 0$:

- If $n \leq t/2$, then $I < 0$;
- If $n > t/2$ and $r_N < \sqrt{t\epsilon^2/(2n-t)}$, then $I < 0$;
- If $n > t/2$ and $r_1 > \sqrt{t\epsilon^2/(2n-t)}$, then $I > 0$.

Proof. Eq. 5.4 can be expanded to the sum of $N/2$ pairs as

$$I = \sum_{i=1}^{N/2} \frac{r_{2i-1}^t}{(r_{2i-1}^2 + \epsilon^2)^n} - \frac{r_{2i}^t}{(r_{2i}^2 + \epsilon^2)^n}. \tag{5.5}$$

When $\epsilon = 0$,

$$I = \sum_{i=1}^{N/2} \frac{1}{r_{2i-1}^{2n-t}} - \frac{1}{r_{2i}^{2n-t}}.$$

We know $r_{2i-1} < r_{2i}$ for any $i \in [1, \ldots, N/2]$, then

- when $n < t/2$, $\frac{1}{r_{k_{2i-1}}^{2n-t}} - \frac{1}{r_{k_{2i}}^{2n-t}} < 0$, thus $I < 0$;
- when $n = t/2$, $\frac{1}{r_{k_{2i-1}}^{2n-t}} - \frac{1}{r_{k_{2i}}^{2n-t}} = 1 - 1 = 0$, thus $I = 0$;
- when $n > t/2$, $\frac{1}{r_{k_{2i-1}}^{2n-t}} - \frac{1}{r_{k_{2i}}^{2n-t}} > 0$, thus $I > 0$.

When $\epsilon \neq 0$, let

$$h(r) = \frac{r^t}{(r^2+\epsilon^2)^n} \tag{5.6}$$

and then

$$h'(r) = \frac{tr^{t-1}(r^2+\epsilon^2)^n - 2nr^{t+1}(r^2+\epsilon^2)^{n-1}}{(r^2+\epsilon^2)^{2n}}. \tag{5.7}$$

When $n \leq t/2$, $h'(r) > 0$ and $h(r)$ is monotonically increasing for all $r \in \mathbf{R}$. Therefore, every pair in Eq. 5.5 is positive and $I > 0$.

When $n > t/2$, if

$$r > \sqrt{\frac{t\epsilon^2}{2n-t}},$$

$h'(r) > 0$ and $h(r)$ is monotonically increasing. As r_1 is the minimum in $\mathbf{R}$, if $r_1 > \sqrt{\frac{t\epsilon^2}{2n-t}}$ then $h(r)$ is monotonically increasing for all $r \in \mathbf{R}$. Therefore, every pair in Eq. 5.5 is positive and $I > 0$.

When $n > t/2$, if

$$r < \sqrt{\frac{t\epsilon^2}{2n-t}},$$

$h'(r) < 0$ and $h(r)$ is monotonically decreasing. As r_N is the maximum in $\mathbf{R}$, if $r_N < \sqrt{\frac{t\epsilon^2}{2n-t}}$ then $h(r)$ is monotonically decreasing for all $r \in \mathbf{R}$. Therefore, every pair in Eq. 5.5 is negative and $I < 0$. □

Lemma 5.1.2. Let $\mathbf{R} = \{r_1, \ldots, r_N\} \in \mathbb{R}^N$, and $r_i < r_j$ for $i < j$. For N odd, $t > 0$ and $\epsilon, n \in \mathbb{R}$, define

$$I = \sum_{i=1}^{N} (-1)^i \frac{r_i^t}{(r_i^2+\epsilon^2)^n} \tag{5.8}$$

When $\epsilon = 0$, $I < 0$. When $\epsilon \neq 0$:

- if $n \leq t/2$, then $I < 0$;
- if $n > t/2$ and $r_N \leq \sqrt{t\epsilon^2/(2n-t)}$, then $I < 0$;
- if $n > t/2$ and $r_1 \geq \sqrt{t\epsilon^2/(2n-t)}$, then $I < 0$.

Proof. Eq. 5.8 can be transformed equivalent to two different formulations:

$$I = -\frac{r_1^t}{(r_1^2+\epsilon^2)^n} + I_1 \quad \text{where} \quad I_1 = \sum_{i=1}^{(N-1)/2} \frac{r_{2i}^t}{(r_{2i}^2+\epsilon^2)^n} - \frac{r_{2i+1}^t}{(r_{2i+1}^2+\epsilon^2)^n} \tag{5.9}$$

and

$$I = -\frac{r_N^t}{(r_N^2+\epsilon^2)^n} - I_2 \quad \text{where} \quad I_2 = \sum_{i=1}^{(N-1)/2} \frac{r_{2i-1}^t}{(r_{2i-1}^2+\epsilon^2)^n} - \frac{r_{2i}^t}{(r_{2i}^2+\epsilon^2)^n}. \tag{5.10}$$

Eq. 5.10 is equivalent to

$$I_2 = \sum_{i=1}^{N-1} (-1)^{i+1} \frac{r_i^t}{(r_i^2+\epsilon^2)^n} \ . \tag{5.11}$$

For Eq. 5.9, let $r'_i = r_{i+1}$, then we have

$$I_1 = \sum_{i=1}^{(N-1)/2} \frac{(r'_{2i-1})^t}{({r'_{2i-1}}^2+\epsilon^2)^n} - \frac{{r'_{2i}}^t}{({r'_{2i}}^2+\epsilon^2)^n} = \sum_{i=1}^{N-1} (-1)^{i+1} \frac{{r'_i}^t}{({r'_i}^2+\epsilon^2)^n} \ . \tag{5.12}$$

Note that both I_1 and I_2 are the summation of $N-1$ (even) of elements. When $\epsilon = 0$, by Lemma 5.1.1, we have

- when $n < t/2$, $I_1 < 0$, so $I = -\frac{r_1^t}{(r_1^2+\epsilon^2)^n} + I_1 < 0$;
- when $n = t/2$, $I_1 = 0$, so $I = -\frac{r_1^t}{(r_1^2+\epsilon^2)^n} < 0$;
- when $n > t/2$, $I_2 > 0$, so $I = -\frac{r_N^t}{(r_N^2+\epsilon^2)^n} - I_2 < 0$.

Therefore, $I < 0$ when $\epsilon = 0$.

If $\epsilon \neq 0$, similarly according to Lemma 5.1.1,

- when $n \leq t/2$, $I_1 < 0$, so $I = -\frac{r_1^t}{(r_1^2+\epsilon^2)^n} + I_1 < 0$;
- when $n > t/2$ and $r_N \leq \sqrt{t\epsilon^2/(2n-t)}$, $I_1 < 0$, so $I = -\frac{r_1^t}{(r_1^2+\epsilon^2)^n} + I_1 < 0$;
- when $n > t/2$ and $r_1 \geq \sqrt{t\epsilon^2/(2n-t)}$, $I_2 > 0$, so $I = -\frac{r_N^t}{(r_N^2+\epsilon^2)^n} - I_2 < 0$.

□

5.2 Necessary Conditions on the Weight Function for IMLS in 2D

In this section, I will provide a generic implicit moving least squares formulation that is equivalent to but different from Eq. 4.2 by using 2D polar coordinates. With the polar coordinate formulation, I will show necessary conditions on the weight function for 2D IMLS interpolation and approximation to construct well-defined implicit functions.

5.2.1 Polar Coordinate Reformulation of IMLS in 2D

In 2D, with an input polygonal curve $\mathbf{L} = \{L_k\}, k = 1, \ldots, K$ and weight function in Eq. 5.3, the original implicit function in Eq. 4.2 is

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k}, \tag{5.13}$$

where

$$A_k = \int_{L_k} w_n(\mathbf{x} - \mathbf{p}_{L_k}) \mathbf{d}\mathbf{p}_{L_k} \quad \text{and} \quad a_k = A_k S_k(\mathbf{x}). \tag{5.14}$$

The normal function $S_k(\mathbf{x})$ is Shen et al.'s shape function (Section 4.1.1) with $\phi = 0$:

$$S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{L_k})^\intercal \mathbf{n}_{L_k}. \tag{5.15}$$

I transform all the points into polar coordinates (r, θ) where the origin is the query point $\mathbf{x}$, and θ ranges from 0 to 2π. $\|\mathbf{x} - \mathbf{p}_{L_k}\|$ for each line of the polygon L_k is given by

$$\|\mathbf{x} - \mathbf{p}_{L_k}\| = r_k(\theta) = b_k \sec|\theta - \varphi|, \tag{5.16}$$

where b_k and φ are the distance and polar angle of the intersection point between L_k and its perpendicular line from $\mathbf{x}$. $\mathbf{n}_k$ is the unit normal vector to L_k. Fig. 5.2 shows the transformed polar coordinates.

From Eq. 5.16, we know

$$\frac{dr_k}{d\theta} = b_k \sec|\theta - \varphi| \tan|\theta - \varphi| = r_k(\theta) \tan|\theta - \varphi|$$

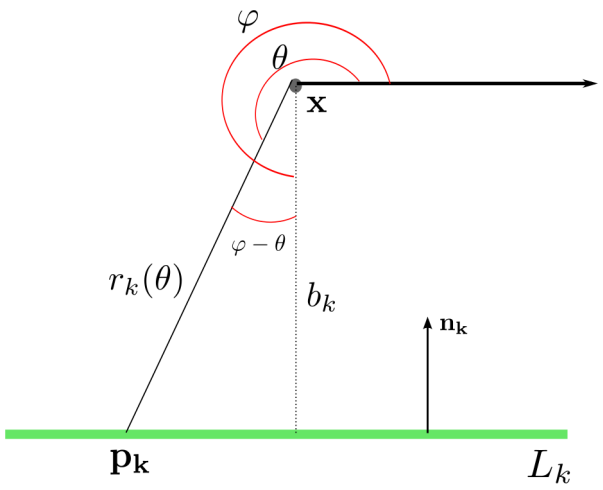


Figure 5.2: 2D polar coordinate notation. Green line is the input line segment and $\mathbf{x}$ is the query point.

and

$$S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{L_k})^\intercal \mathbf{n}_{L_k} = b_k.$$

Therefore, from Eq. 5.14,

$$a_k = S_k(\mathbf{x}) \int_{L_k} w_n(\mathbf{x} - \mathbf{p}_{L_k}) d\mathbf{p}_{L_k} \tag{5.17}$$

$$= b_k \int_{L_k} w_n(r_k) d\mathbf{p}_{L_k}. \tag{5.18}$$

By using the arc length formula for polar coordinates,

$$\begin{aligned} a_k &= b_k \int_{L_k} w_n(r_k)\sqrt{r_k^2 + \left(\frac{dr_k}{d\theta}\right)^2}\, d\theta \\ &= b_k \int_{L_k} w_n(r_k)\sqrt{r_k^2 + r_k^2 \tan^2|\theta - \varphi|}\, d\theta \\ &= b_k \int_{L_k} w_n(r_k)\sqrt{r_k^2 \sec^2|\theta - \varphi|}\, d\theta \\ &= b_k \int_{L_k} w_n(r_k)\sqrt{r_k^2 (\frac{r_k}{b_k})^2} \\ &= \operatorname{sgn}(b_k) \int_{L_k} r_k^2 w_n(r_k)\, d\theta, \end{aligned} \tag{5.19}$$

where $\operatorname{sgn}(\cdot)$ is the sign function. I will use Eq. 5.19 in the next section to give necessary conditions on the weight function.

5.2.2 Proof of the Weight Function for IMLS in 2D

In this section, I will present proofs of the necessary conditions on the weight function in Eq. 5.3 for 2D IMLS interpolation (with $\epsilon = 0$) and 2D IMLS approximation (with $\epsilon \neq 0$).

IMLS Interpolation in 2D

Theorem 5.2.1. In 2D, for the implicit moving least squares method interpolating a manifold polygon defined by the set of line segments $\mathbf{L} = \{L_k\}, k = 1, \ldots, K$, $\mathbf{p}_{L_k}$ being a point on line segment L_k, and $\mathbf{x}$ being the point of evaluation, let the implicit function be

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k} \tag{5.20}$$

where $A_k = \int_{L_k} w_n(\mathbf{x} - \mathbf{p}_{L_k}) d\mathbf{p}_{L_k}$, $a_k = A_k S_k(\mathbf{x})$, with weight function $w_n(\mathbf{x} - \mathbf{p}_{L_k}) = \frac{1}{\|\mathbf{x} - \mathbf{p}_{L_k}\|^{2n}}$ for an arbitrary $n \in \mathbb{R}$, and normal function $S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{L_k})^\intercal \mathbf{n}_{L_k}$. For any $\mathbf{x}$:

- if $\mathbf{x}$ is in the interior of $\mathbf{L}$, then $f(\mathbf{x}) < 0$;
- if $\mathbf{x}$ is in the exterior of $\mathbf{L}$,

– if $n = 1$, $f(\mathbf{x}) = 0$,
– if $n > 1$, $f(\mathbf{x}) > 0$,
– if $n < 1$, $f(\mathbf{x}) < 0$.

Proof. Since

$$A_k = \int_{L_k} w_n(\mathbf{x} - \mathbf{p}_{L_k}) \mathbf{d} p_{L_k}, \tag{5.21}$$

is always positive, the sign of the implicit function $f(\mathbf{x})$ in Eq. 5.20 is determined by the numerator $\sum_{k=1}^{K} a_k$. Therefore, in the following, I analyze $\sum_{k=1}^{K} a_k$ to determine the sign of the implicit function at points in the interior and exterior of $\mathbf{L}$.

From Eq. 5.19, we know

$$\sum_{k=1}^{K} a_k = \sum_{k=1}^{K} \text{sgn}(b_k) \int_{L_k} r_k^2 w_n(r_k) d\theta \tag{5.22}$$

$$= \sum_{k=1}^{K} \int_{L_k} \text{sgn}(b_k) r_k^2 w_n(r_k) d\theta \tag{5.23}$$

Eq. 5.23 integrates over each line segment and sums up all the integrations. A ray cast from $\mathbf{x}$ to intersects $\mathbf{L}$ in several places. While rotating the ray 360° around $\mathbf{x}$, the intersections cover all the points on $\mathbf{L}$. This is illustrated in Fig. 5.3.

Therefore, for any θ, $r_k^2 w(r_k)$ can be computed by casting a ray $\vec{r}$ from $\mathbf{x}$ in direction θ, and intersecting $\vec{r}$ with $\mathbf{L}$. Let ℓ_i be the ith intersection as we step along the ray from $\mathbf{x}$, where the value of ℓ_i is the index of the line segment intersected. The set of intersections for a ray in direction θ from $\mathbf{x}$ can be represent as

$$\mathbf{R}(\theta) = \{\ell_i\}, i = 1, 2, \ldots, N,$$

where N depends on θ. Therefore, the summation of all a_k can be represented by

$$\sum_{k=1}^{K} a_k = \sum_{k=1}^{K} \text{sgn}(b_k) \int_{L_k} \frac{1}{r_k^{2n-2}} d\theta \tag{5.24}$$

$$= \int \sum_{\ell \in \mathbf{R}(\theta)} \text{sgn}(b_\ell) \frac{1}{r_\ell^{2n-2}} d\theta \tag{5.25}$$

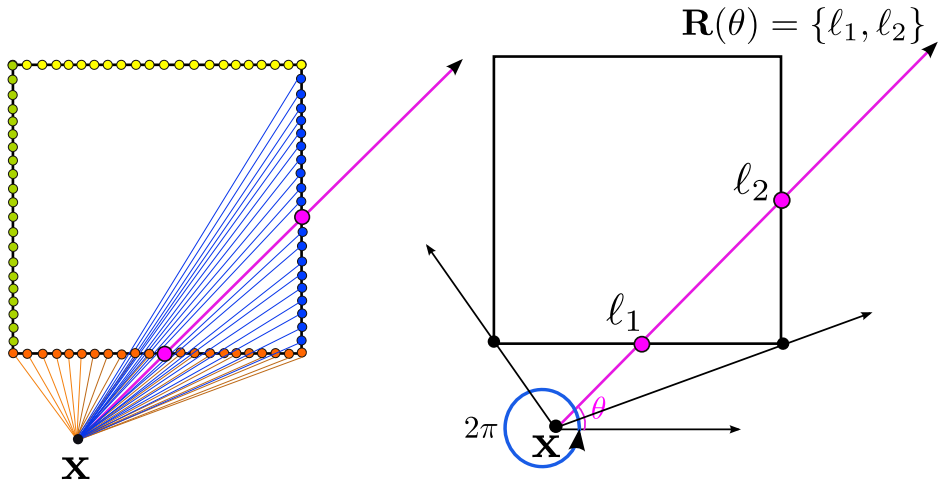


Figure 5.3: Ray casting observation.

Since we step along the ray from $\mathbf{x}$, the elements of $\mathbf{R}(\theta)$ are in ascending order in their distance from $\mathbf{x}$, i.e., $r_{\ell_i}(\theta) < r_{\ell_j}(\theta)$ for $i < j$, as shown in Fig. 5.4.

From Eq. 5.25, the sign of f can be determined as follows:

1. For each θ, the sign of the integrand

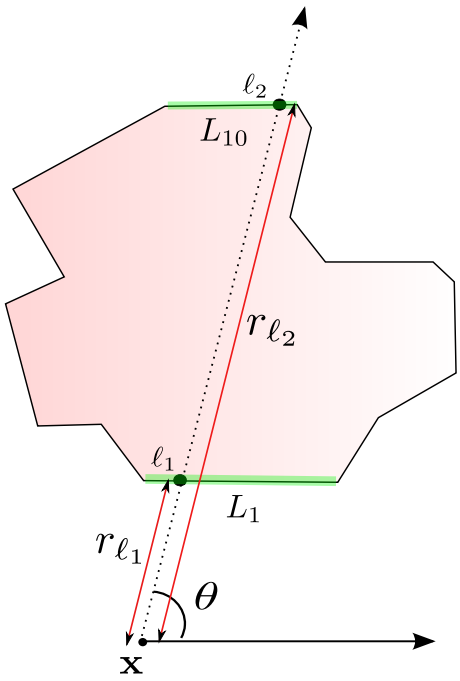
$$I(\theta) = \sum_{\ell \in \mathbf{R}(\theta)} \operatorname{sgn}(b_\ell) \frac{1}{r_\ell^{2n-2}} \tag{5.26}$$

   can be computed as a sum of positive and negative values that can be paired based on N.

2. For each θ, if $I(\theta)$ is of one sign, then the integral is of that sign.

3. If the sign of $\sum_{k=1}^{K} a_k$ is of one sign, then $f(\mathbf{x})$ in Eq. 5.20 is of that sign.

There are a finite number of intersections located exactly on the shared end points of two line segments. If the two line segments are on the same side of the ray, the intersection is ignored. When the ray coincides with a line segment, the implicit value contribution of the line segment is always zero. Discarding these intersections in $\mathbf{R}(\theta)$ will not influence



$$\mathbf{R}(\theta) = \{k_1, k_2\} = \{1, 10\}$$

Figure 5.4: Visual illustration of intersections for the ray cast from the point $\mathbf{x}$ to the polygon in direction θ. $\mathbf{R}(\theta) = \{\ell_1, \ell_2\}$ where $\ell_1 = 1$ and $\ell_2 = 10$. r_{ℓ_1} and r_{ℓ_2} are distances from $\mathbf{x}$ to intersections on 1st and 10th line segments.

the integral of Eq. 5.25. Hence, from standard ray casting [36], we know that if $\mathbf{x}$ is in the exterior of $\mathbf{L}$, N is even, and if the origin is in the interior of $\mathbf{L}$, N is odd.

From the Jordan curve theorem, we know that space is divided by the polygon into two components, the bounded interior and the unbounded exterior. Thus every line segment indexed by $\mathbf{R}(\theta)$ separates the ray into two components, with the ray always alternating between the exterior and interior of the polygon.

When $\mathbf{x}$ is in the exterior of $\mathbf{L}$, we know that $\text{sgn}(b_{\ell_j}) = (-1)^{j+1}$ where $\ell_j \in \mathbf{R}(\theta)$. Therefore, we can rewrite the integrand in Eq. 5.26 as

$$I(\theta) = \sum_{j=1}^{N} (-1)^{j+1} \frac{1}{r_{\ell_j}^{2n-2}} \tag{5.27}$$

where $\ell_j \in \mathbf{R}(\theta)$ and N is even. Hence, from Lemma 5.1.1 with $t = 2$ and $\epsilon = 0$, we get:

- If $n < 1$ then $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$;
- If $n = 1$ then $I(\theta) = 0$ for each θ, thus $f(\mathbf{x}) = 0$;
- If $n > 1$ then $I(\theta) > 0$ for each θ, thus $f(\mathbf{x}) > 0$.

An example is shown in Fig. 5.5.

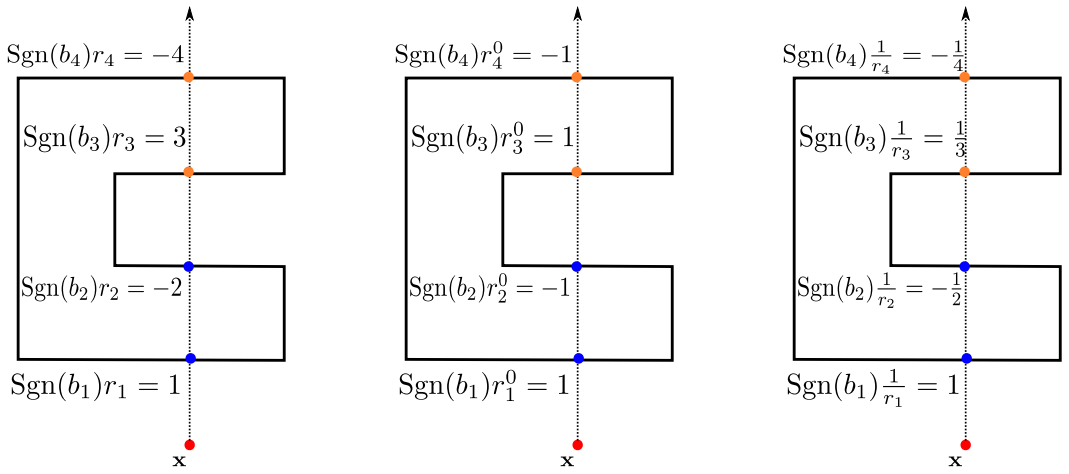
For the case where $\mathbf{x}$ is in the interior of $\mathbf{L}$, $\text{sgn}(b_{\ell_j}) = (-1)^j$, we can rewrite Eq. 5.25 as

$$I(\theta) = \sum_{j=1}^{N} (-1)^{j} \frac{1}{r_{\ell_j}^{2n-2}}, \tag{5.28}$$

where $\ell_j \in \mathbf{R}(\theta)$ and N is odd. Hence, from Lemma 5.1.2 with $t = 2$ and $\epsilon = 0$, we get $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$. Therefore, $f(\mathbf{x}) < 0$ for any $\mathbf{x}$ in the interior of $\mathbf{L}$. □

Corollary 5.2.2. To interpolate using IMLS with the normal function in Eq. 4.1 in 2D, and have the property that the implicit function has positive value outside the polygonal curve, zero on the polygonal curve and negative value inside the polygonal curve, the weight function in Eq. 5.3 should have $n > 1$.

Thus, we see that the squaring of the weight function in Shen et al.'s method was critical for its correctness.



(a) $n = 1/2$ in Eq. 5.3 (b) $n = 1$ in Eq. 5.3 (c) $n = 3/2$ in Eq. 5.3

Figure 5.5: Example of different exterior results with three weight functions. $\mathbf{x}$ is the query point, $r_1 = 1, r_2 = 2, r_3 = 3$, and $r_4 = 4$. Intersections are grouped into blue and orange.

IMLS Approximation in 2D

In the proof of interpolation, we only need to consider inside and outside cases, since when $\mathbf{x}$ is on the polygonal curve, the weight function for interpolation dominates and $f(\mathbf{x}) = 0$. However, in approximation, while $\mathbf{x}$ is on the polygonal curve, $f(\mathbf{x}) \neq 0$ and more effort is needed to determine the sign of $f(\mathbf{x})$.

Theorem 5.2.3. In 2D, for the implicit moving least squares method approximating a manifold polygonal curve defined by the set of line segments $\mathbf{L} = \{L_k\}, k = 1, \ldots, K$, $\mathbf{p}_{L_k}$ being a point on line segment L_k, and $\mathbf{x}$ being the point of evaluation, let the implicit function be

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k}, \tag{5.29}$$

where $A_k = \int_{L_k} w(\mathbf{x} - \mathbf{p}_{L_k}) d\mathbf{p}_{L_k}$, $a_k = A_k S_k(\mathbf{x})$, with weight function $w(\mathbf{x} - \mathbf{p}_{L_k}) = \left(\frac{1}{\|\mathbf{x}-\mathbf{p}_{L_k}\|^2+\epsilon^2}\right)^n$ for an arbitrary $n \in \mathbb{R}$, and normal function $S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{L_k})^\intercal \mathbf{n}_{L_k}$. Then

- when $n \leq 1$, for any $\mathbf{x}$, $f(\mathbf{x}) < 0$;
- when $n > 1$, for any $\mathbf{x}$, let $r_{min}(\mathbf{x})$ and $r_{max}(\mathbf{x})$ be the minimum and maximum distance from $\mathbf{x}$ to $\mathbf{L}$:
  - if $r_{max}(\mathbf{x}) < \sqrt{\epsilon^2/(n-1)}$ then $f(\mathbf{x}) < 0$.
  - if $\mathbf{x}$ is in the exterior of $\mathbf{L}$ and $r_{min}(\mathbf{x}) > \sqrt{\epsilon^2/(n-1)}$, then $f(\mathbf{x}) > 0$;
  - if $\mathbf{x}$ is in the interior of $\mathbf{L}$, and $r_{min}(\mathbf{x}) \geq \sqrt{\epsilon^2/(n-1)}$, then $f(\mathbf{x}) < 0$;

Proof. From the proof of Theorem 5.2.1, we know the sign of the implicit function $f(\mathbf{x})$ in Eq. 5.29 is determined by the numerator $\sum_{k=1}^{K} a_k$. Therefore, in the following, I analyze $\sum_{k=1}^{K} a_k$ to determine the sign of the implicit function.

Using the same method to cast a ray $\vec{r}$ from $\mathbf{x}$ in direction θ, and following the same notation as in the proof of Theorem 5.2.1, we can reformulate Eq. 5.19 as

$$\sum_{k=1}^{K} a_k = \sum_{k=1}^{K} \operatorname{sgn}(b_k) \int_{L_k} \frac{r_k^2}{(r_k^2+\epsilon^2)^n} d\theta \tag{5.30}$$

$$= \int \sum_{\ell \in \mathbf{R}(\theta)} \operatorname{sgn}(b_\ell) \frac{r_\ell^2}{(r_\ell^2+\epsilon^2)^n} d\theta. \tag{5.31}$$

As we step along the ray from $\mathbf{x}$, the set $\mathbf{R}(\theta)$ is in ascending order in the distance from $\mathbf{x}$, i.e., $r_{\ell_i}(\theta) < r_{\ell_j}(\theta)$ for $i < j$.

From Eq. 5.31, the sign of f can be determined as follows:

1. For each θ, the sign of the integrand

$$I(\theta) = \sum_{\ell \in \mathbf{R}(\theta)} \text{sgn}(b_\ell) \frac{r_\ell^2}{(r_\ell^2 + \epsilon^2)^n} \tag{5.32}$$

   can be computed as a sum of positive and negative values that can be paired based on N.

2. For each θ, if $I(\theta)$ is of one sign then the integral is of that sign, thus $\sum_{k=1}^{K} a_k$ is of that sign.

3. If the sign of $\sum_{k=1}^{K} a_k$ is of one sign, then $f(\mathbf{x})$ in Eq. 5.31 is of that sign.

Again discarding non-constructive intersections in $\mathbf{R}(\theta)$, we get that if $\mathbf{x}$ is in the exterior of $\mathbf{L}$, then N is even, and if $\mathbf{x}$ is in the interior of $\mathbf{L}$, then N is odd. Additionally, the ray is always alternating between the exterior and interior of $\mathbf{L}$.

When $\mathbf{x}$ is in the exterior of $\mathbf{L}$, we know that $\text{sgn}(b_{\ell_j}) = (-1)^{j+1}$ where $k \in \mathbf{R}(\theta)$. Therefore, we can rewrite the integrand in Eq. 5.32 as

$$I(\theta) = \sum_{j=1}^{N} (-1)^{j+1} \frac{r_{\ell_j}^2}{(r_{\ell_j}^2 + \epsilon^2)^n} \tag{5.33}$$

where $\ell_j \in \mathbf{R}(\theta)$ and N is even. Hence, from Lemma 5.1.1 with $\epsilon \neq 0$ and $t = 2$,

- If $n \leq 1$ then $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$;
- If $n > 1$ and $r_{\ell_N}(\theta) < \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) < 0$. Let $r_{max} = \max\{r_{\ell_N}(\theta)\}$ for all θ. If $r_{max} < \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$.
- If $n > 1$ and $r_{\ell_1}(\theta) > \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) > 0$. Let $r_{min} = \min\{r_{\ell_1}(\theta)\}$ for all θ. If $r_{min} > \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) > 0$ for each θ, thus $f(\mathbf{x}) > 0$.

When $\mathbf{x}$ is in the interior of $\mathbf{L}$, $\text{sgn}(b_{\ell_j}) = (-1)^j$, we can rewrite the integrand in Eq. 5.32 as in

$$I(\theta) = \sum_{j=1}^{N} (-1)^j \frac{r_{\ell_j}^2}{(r_{\ell_j}^2 + \epsilon^2)^n} \tag{5.34}$$

where $\ell_j \in \mathbf{R}(\theta)$ and N is odd. Hence, from Lemma 5.1.2 with $\epsilon \neq 0$ and $t = 2$,

- if $n \leq 1$ then $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$.
- If $n > 1$ and $r_{\ell_N}(\theta) \leq \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) < 0$. We know $r_{max} = \max\{r_{\ell_N}(\theta)\}$ for all θ. If $r_{max} \leq \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$.
- If $n > 1$ and $r_{\ell_1}(\theta) \geq \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) < 0$. We know $r_{min} = \min\{r_{\ell_1}(\theta)\}$ for all θ. If $r_{min} \geq \sqrt{\epsilon^2/(n-1)}$ then $I(\theta) < 0$ for each θ, thus $f(\mathbf{x}) < 0$.

For $\mathbf{x}$ on $\mathbf{L}$, $\mathbf{x}$ is on a particular line segment $L_{k'}$. The intersection of the ray $\vec{r}$ with $L_{k'}$ (which is actually $\mathbf{x}$) has no contribution to $\sum_{k=1}^{K} a_k$ because $b_{k'} = 0$ and $a_{k'} = 0$. Therefore, for a ray cast from $\mathbf{x}$ on $L_{k'}$, the first intersection on $L_{k'}$ is ignored. When the ray is in the direction of the outside (inside) of $\mathbf{L}$, it is the same case as when $\mathbf{x}$ is in the exterior (interior) of $\mathbf{L}$. Therefore, for $\mathbf{x}$ is on $\mathbf{L}$, the sign $f(\mathbf{x})$ is determined only in those cases when the interior and exterior have the same result, i.e., when $n \leq 1$, $f(\mathbf{x}) < 0$; when $n > 1$ and $r_{max} < \sqrt{\epsilon^2/(n-1)}$, $f(\mathbf{x}) < 0$. □

What Theorem 5.2.3 tells us is that for $n > 1$, there is an unbounded region outside of $\mathbf{L}$ where f is strictly positive and when certain conditions are met, there is a bounded region inside $\mathbf{L}$ where f is strictly negative as illustrated in Fig. 5.6. This leads to the following corollary.

Corollary 5.2.4. When using IMLS to approximate data ($\epsilon > 0$), if $n > 1$ and if there exists a point $\mathbf{x}$ inside of $\mathbf{L}$ such that $r_{min} \geq \sqrt{\epsilon^2/(n-1)}$ or $r_{max} \leq \sqrt{\epsilon^2/(n-1)}$, then the resulting implicit function f is well defined.

From Theorem 5.2.3, we can show the following.

Corollary 5.2.5. When using IMLS to approximate data ($\epsilon > 0$), if $n > 1$ and the input data is convex, the resulting implicit function f is well defined. In addition the interior of $\mathbf{L}$ is on the interior of f.

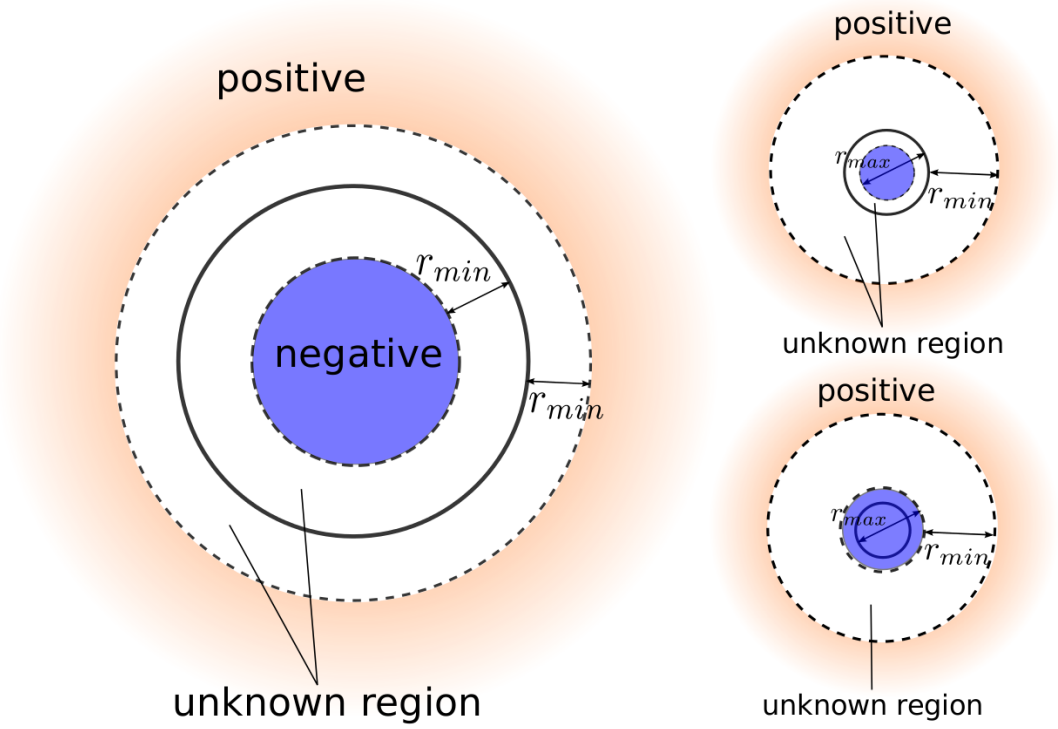


Figure 5.6: Positive and negative zones for IMLS approximation when $n > 1$. The solid circles are the input data. The minimum and maximum distances from the dashed circles to the solid circles are marked as r_{min} and r_{max} and $r_{min} = r_{max} = \sqrt{\epsilon^2/(n-1)}$.

From Theorem 5.2.3, we know that f is positive in an unbounded region a distance of $\sqrt{\epsilon^2/(n-1)}$ from $\mathbf{L}$. If the conditions of Corollary 5.2.4 are met, then we know that there is at least one point $\mathbf{x}$ on the interior $\mathbf{L}$ where $f(\mathbf{p}) < 0$ and thus f is well defined. If the conditions of Corollary 5.2.4 are not met, then f might still be well defined. For example, in Fig. 5.7, the size of the square is $0.58 * 0.58$, and I use $\epsilon = 0.3$ and $n = 2$ in the weight function in Eq. 5.3. Thus the threshold distance $\sqrt{\epsilon^2/(n-1)}$ in this example is 0.3. For any point in the interior of the square, $r_{min} \leq 0.29$ and $r_{max} \geq 0.29 * \sqrt{2}$, the conditions of Corollary 5.2.4 are not met. But since the conditions of Corollary 5.3.5 are met, then f is well defined. If the conditions of both corollaries are not met, then f might not be well defined, although I have not been able to find an example where this is the case.

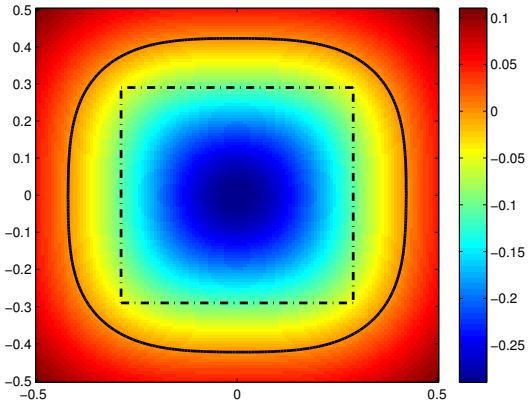


Figure 5.7: An example where the conditions in Corollary 5.2.4 are not met but a well-defined approximation curve is generated. Approximating with $\epsilon = 0.3$ and n = 2. The solid curve is the approximation and dashed square is the input data. The size of the square is $0.58 * 0.58$.

5.3 Necessary Conditions on the Weight Function for IMLS in 3D

In this section, I will provide a generic implicit moving least squares formulation that is equivalent to but different from Eq. 4.2 by using 3D spherical coordinates. With the spherical coordinate formulation, I will show necessary conditions on weight function for 3D IMLS interpolation and approximation to avoid degeneracy problems.

5.3.1 Spherical Coordinate Formulation of IMLS in 3D

In 3D, with an input polyhedral surface $\mathbf{\Omega} = \{\Omega_k\}$, $k = 1, \ldots, K$ and the weight function in Eq. 5.3, the implicit function in Eq. 4.2 is

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{k} A_k}, \tag{5.35}$$

where

$$A_k = \iint_{\Omega_k} w_n(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} \quad \text{and} \quad a_k = A_k S_k(\mathbf{x}). \tag{5.36}$$

let the normal function $S_k(\mathbf{x})$ be Shen et al.'s shape function (Section 4.1.1) with $\phi = 0$:

$$S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k}. \tag{5.37}$$

Similar to the 2D case, in 3D I transform points into spherical coordinates (r, θ, φ) where the origin is the query point $\mathbf{x}$, θ ranges from 0 to π, and φ ranges from 0 to 2π. Let $\mathbf{r}_k = \mathbf{x} - \mathbf{p}_{\Omega_k}$ for Ω_k and $r_k = \|\mathbf{r}_k\|$ be the distance from the query point $\mathbf{x}$ to points on Ω_k, then

$$S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k} = \mathbf{r}_k^\intercal \mathbf{n}_{\Omega_k} = b_k \tag{5.38}$$

is the signed distance from $\mathbf{x}$ to Ω_k. From the definition of a solid angle, Eq. 4.16, Eq. 4.17 and Eq. 4.18,

$$d\omega = \frac{|\mathbf{r}_k^\intercal \mathbf{n}_{\Omega_k}|}{r_k^3} d\mathbf{p}_{\Omega_k}, \tag{5.39}$$

where $\mathbf{n}_{\Omega_k}$ is the unit normal vector to Ω_k. Since ω is an area on the unit sphere, we also have

$$d\omega = \sin\theta \; d\theta d\varphi. \tag{5.40}$$

Thus, by substituting Eq. 5.38, Eq. 5.40 and Eq. 5.39 into Eq. 5.36, we get

$$\begin{aligned} a_k &= S_k(\mathbf{x}) \iint_{\Omega_k} w_n(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} \\ &= \iint_{\Omega_k} w_n(r_k) \mathbf{r}_k^\intercal \mathbf{n}_{\Omega_k} d\mathbf{p}_{\Omega_k} \\ &= \operatorname{sgn}(b_k) \iint_{\Omega_k} w_n(r_k) r_k^3 d\omega \\ &= \operatorname{sgn}(b_k) \iint_{\Omega_k} w_n(r_k) r_k^3 \sin\theta \; d\theta d\varphi \end{aligned} \tag{5.41}$$

where $\operatorname{sgn}(\cdot)$ is the sign function. I will use Eq. 5.41 in the next section to give necessary conditions on the weight function.

5.3.2 Proof of the Weight Function for IMLS in 3D

In this section, I will use a similar approach as in 2D to give a bound for the weight function in Eq. 5.3 to avoid degeneracies in IMLS interpolation and approximation in 3D.

IMLS Interpolation in 3D

Theorem 5.3.1. In 3D, for the implicit moving least squares method interpolating a manifold polyhedral surface defined by a set of polygons $\mathbf{\Omega} = \{\Omega_k\}, k = 1, \ldots, N$, $\mathbf{p}_{\Omega_k}$ being a point on polygon Ω_k, and $\mathbf{x}$ being the evaluated point, let the implicit function be

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k}, \tag{5.42}$$

where $A_k = \iint_{\Omega_k} w_n(\mathbf{x}-\mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}$, $a_k = A_k S_k(\mathbf{x})$, with weight function $w_n(\mathbf{x}-\mathbf{p}) = \frac{1}{\|\mathbf{x}-\mathbf{p}\|^{2n}}$ for an arbitrary $n \in \mathbb{R}$ and shape function $S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k}$. For any $\mathbf{x}$:

- if $\mathbf{x}$ is in the interior of $\mathbf{\Omega}$, then $f(\mathbf{x}) < 0$;
- if $\mathbf{x}$ is in the exterior of $\mathbf{\Omega}$,
  - if $n > 3/2$, $f(\mathbf{x}) > 0$,
  - if $n = 3/2$, $f(\mathbf{x}) = 0$,
  - if $n < 3/2$, $f(\mathbf{x}) < 0$.

Proof. Since

$$A_k = \iint_{\Omega_k} w_n(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} \tag{5.43}$$

is always positive, the sign of the implicit function $f(\mathbf{x})$ in Eq. 5.42 is determined by the numerator $\sum_{k=1}^{K} a_k$. Therefore, in the following, I analyze $\sum_{k=1}^{K} a_k$ to determine the sign of the implicit function at points in the interior and exterior of $\mathbf{\Omega}$

In Eq. 5.41,

$$a_k = \operatorname{sgn}(b_k) \iint_{\Omega_k} w_n(r_k) r_k^3 \sin\theta \; d\theta d\varphi. \tag{5.44}$$

For any (θ, φ), $w_n(r_k) r_k^3$ can be computed by casting a ray $\vec{r}$ from $\mathbf{x}$ in direction (θ, φ), and intersecting $\vec{r}$ with $\mathbf{\Omega}$. Let ℓ_i be the ith intersection of $\vec{r}$ with $\mathbf{\Omega}$ as we step along $\vec{r}$ from $\mathbf{x}$, where the value of ℓ_i is the index of the polygon intersected by $\vec{r}$. The set of intersections for a ray in direction (θ, φ) from $\mathbf{x}$ can be represent as

$$\mathbf{R}(\theta, \varphi) = \{\ell_i\}, i = 1, 2, \ldots, N,$$

where N depends on (θ, φ). Therefore, the summation of the a_k's is

$$\sum_{k=1}^{K} a_k = \sum_{k=1}^{K} \operatorname{sgn}(b_k) \iint \frac{1}{r_k^{2n-3}} \sin\theta \; d\theta d\varphi \tag{5.45}$$

$$= \iint \sin\theta \sum_{\ell \in \mathbf{R}(\theta,\varphi)} \operatorname{sgn}(b_\ell) \frac{1}{r_\ell^{2n-3}} \; d\theta d\varphi, \tag{5.46}$$

Since we step along the ray from $\mathbf{x}$, the set of $\mathbf{R}$ is in ascending order in their distance to $\mathbf{x}$, i.e., $r_{\ell_i}(\theta, \varphi) < r_{\ell_j}(\theta, \varphi)$ for $i < j$.

From Eq. 5.46, the sign of f can be determined as follows:

1. Since $\theta \in [0, \pi]$ and $\sin\theta \geq 0$, thus the sign of the integrand in Eq. 5.46 can be determined from
$$I(\theta, \varphi) = \sum_{\ell \in \mathbf{R}(\theta,\varphi)} \operatorname{sgn}(b_{\ell_j}) \frac{1}{r_\ell^{2n-3}}. \tag{5.47}$$
For each (θ, φ), $I(\theta, \varphi)$ can be computed as a sum of positive and negative values, which can be paired based on N.

2. For each pair (θ, φ), if the integrand is of one sign then the integral is of that sign, thus $\sum_{k=1}^{K} a_k$ is of that sign.

3. If the sign of $\sum_{k=1}^{K} a_k$ is of one sign, then $f(\mathbf{x})$ in Eq. 5.42 is of that sign.

There are a finite number of intersections located exactly on the shared vertices of several polygons. At any such vertex, if the ray does not transit between interior and exterior at this vertex, then the intersection is discarded. When the ray is in the plane of a polygon, the implicit value contribution of the polygon is always zero. Discarding these intersections in $\mathbf{R}(\theta, \varphi)$ will not influence the integral of Eq. 5.46. Hence, from standard ray casting [36], we know that if $\mathbf{x}$ is in the exterior of $\mathbf{\Omega}$, N is even, and if the origin is in the interior of $\mathbf{\Omega}$, N is odd.

By the Jordan-Brouwer Theorem, we know that space is divided by the surface into two component, the bounded interior and the unbounded exterior. Thus, every polygon indexed by $\mathbf{R}(\theta, \varphi)$ separates the ray into two components, and the ray always alternating between the exterior and interior of the space.

When $\mathbf{x}$ is in the exterior of $\mathbf{\Omega}$, we know that $\text{sgn}(b_{\ell_j}) = (-1)^{j+1}$ where $\ell_j \in (\theta, \varphi)$. Therefore, we rewrite $I(\theta, \varphi)$ in Eq. 5.47 as

$$I(\theta, \varphi) = \sum_{j=1}^{N} (-1)^{j+1} \frac{1}{r_{\ell_j}^{2n-3}}. \tag{5.48}$$

where $\ell_j \in \mathbf{R}(\theta, \varphi)$ and N is even. Hence, as in Lemma 5.1.1 with $t = 3$ and $\epsilon = 0$, we get:

- If $n < 3/2$ then $I(\theta, \varphi) < 0$ for each combination of (θ, φ), thus $f(\mathbf{x}) < 0$;
- If $n = 3/2$ then $I(\theta, \varphi) = 0$ for each combination of (θ, φ), thus $f(\mathbf{x}) = 0$;
- If $n > 3/2$ then $I(\theta, \varphi) > 0$ for each combination of (θ, φ), thus $f(\mathbf{x}) > 0$;

When $\mathbf{x}$ is in the interior of $\mathbf{\Omega}$, $\text{sgn}(b_{\ell_j}) = (-1)^j$, we can rewrite Eq. 5.47 as

$$I(\theta, \varphi) = \sum_{j=1}^{N} (-1)^{j} \frac{1}{r_{\ell_j}^{2n-3}}. \tag{5.49}$$

where $\ell_j \in \mathbf{R}(\theta, \varphi)$ and N is odd. Hence, as in Lemma 5.1.2 with $t = 3$ and $\epsilon = 0$, we get $I(\theta, \varphi) < 0$ for any combination of (θ, φ), thus $f(\mathbf{x}) < 0$. Therefore, $f(\mathbf{x}) < 0$ for any $\mathbf{x}$ in the interior of $\mathbf{\Omega}$. □

Corollary 5.3.2. To interpolate using IMLS with the normal function in Eq. 4.1 in 3D, and have the property that the implicit function has positive value outside the polyhedral surface, zero on the polyhedral surface and negative value inside the polyhedral surface, the weight function in Eq. 5.3 should have $n > 3/2$.

IMLS Approximation in 3D

In the proof of interpolation, we only need to consider the inside and outside cases, because when $\mathbf{x}$ is on the polyhedral surface, weight function for interpolation dominates and $f(\mathbf{x}) = 0$. However, in approximation, when $\mathbf{x}$ is on the polyhedral surface, $f(\mathbf{x}) \neq 0$ and more effort is required to determine the sign of $f(\mathbf{x})$.

Theorem 5.3.3. In 3D, for the implicit moving least squares method approximating a polyhedron $\mathbf{\Omega} = \{\Omega_k\}, k = 1, \ldots, K$, $\mathbf{p}_{\Omega_k}$ being a point on polygon Ω_k, and $\mathbf{x}$ being the point of evaluation, let the implicit function be

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k}{\sum_{k=1}^{K} A_k}, \tag{5.50}$$

where $A_k = \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}$, $a_k = A_k S_k(\mathbf{x})$, with weight function $w(\mathbf{x} - \mathbf{p}_{\Omega_k}) = \left(\frac{1}{\|\mathbf{x}-\mathbf{p}_{\Omega_k}\|^2+\epsilon^2}\right)^n$ for an arbitrary $n \in \mathbb{R}$, and normal function $S_k(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k}$. Then

- when $n \leq 3/2$, for any $\mathbf{x}$, $f(\mathbf{x}) < 0$;
- when $n > 3/2$, for any $\mathbf{x}$, let $r_{min}(\mathbf{x}) = \min(r_i)$ and $r_{max}(\mathbf{x}) = \max(r_i), i \in [1, \ldots, N]$:
  - if $r_{max}(\mathbf{x}) < \sqrt{3\epsilon^2/(2n-3)}$, then $f(\mathbf{x}) < 0$ then $f(\mathbf{x}) < 0$;
  - if $\mathbf{x}$ is in the exterior of $\mathbf{\Omega}$ and $r_{min}(\mathbf{x}) > \sqrt{3\epsilon^2/(2n-3)}$, then $f(\mathbf{x}) > 0$;
  - if $\mathbf{x}$ is in the interior of $\mathbf{\Omega}$, and $r_{min}(\mathbf{x}) \geq \sqrt{3\epsilon^2/(2n-3)}$ then $f(\mathbf{x}) < 0$.

Proof. From the proof of Theorem 5.3.1, we know the sign of the implicit function $f(\mathbf{x})$ in Eq. 5.50 is determined by the numerator $\sum_{k=1}^{K} a_k$. Therefore, in the following, I analyze $\sum_{k=1}^{K} a_k$ to determine the sign of the implicit function at points in the interior and exterior of $\mathbf{\Omega}$.

Using the same method to cast a ray $\vec{r}$ from $\mathbf{x}$ in direction (θ, φ), and following the same notation in the proof of Theorem 5.3.1, we can reformulate Eq. 5.41 as:

$$\sum_{k=1}^{K} a_k = \sum_{k=1}^{K} \text{sgn}(b_{k_j}) \iint \frac{r_k^3}{(r_k^2 + \epsilon^2)^n} \sin\theta \; d\theta \; d\varphi \tag{5.51}$$

$$= \iint \sin\theta \sum_{\ell \in \mathbf{R}(\theta)} \text{sgn}(b_{\ell_j}) \frac{r_\ell^3}{(r_\ell^2 + \epsilon^2)^n} \; d\theta \; d\varphi \tag{5.52}$$

where

$$\mathbf{R}(\theta, \varphi) = \{\ell_i\}, i = 1, \ldots, N$$

and N depends on (θ, φ). As we step along the ray from $\mathbf{x}$, the set $\mathbf{R}(\theta, \varphi)$ is in ascending order in the distance from $\mathbf{x}$, i.e., $r_{\ell_i}(\theta, \varphi) < r_{\ell_j}(\theta, \varphi)$ for $i < j$, as shown in Fig. 5.4.

From Eq. 5.52, the sign of f can be determined as follows:

1. Since $\theta \in [0, \pi]$ and $\sin\theta \geq 0$, the sign of the integrand in Eq. 5.52 can be determined from

$$I(\theta, \varphi) = \sum_{k \in \mathbf{R}(\theta)} \text{sgn}(b_{k_j}) \frac{r_k^3}{(r_k^2 + \epsilon^2)^n}. \tag{5.53}$$

   For each combination (θ, φ), $I(\theta, \varphi)$ can be computed as a sum of positive and negative values, which can be paired based on N.

2. For each combination (θ, φ), if the integrand is of one sign then the integral is of that sign, thus $\sum_{k=1}^{K} a_k$ is of that sign.

3. If the sign of $\sum_{k=1}^{K} a_k$ is of one sign, then $f(\mathbf{x})$ in Eq. 5.50 is of that sign.

Discarding non-constructive intersections in $\mathbf{R}(\theta, \varphi)$, we get that if $\mathbf{x}$ is in the exterior of $\mathbf{\Omega}$, then N is even, and if $\mathbf{x}$ is in the interior of $\mathbf{\Omega}$, then N is odd. Additionally, the ray is always alternating between the exterior and interior of $\mathbf{\Omega}$.

When $\mathbf{x}$ is in the exterior, we know that $\text{sgn}(b_{k_j}) = (-1)^{j+1}$ where $k_j \in \mathbf{R}(\theta, \varphi)$. Therefore, we can rewrite Eq. 5.53 as

$$I(\theta, \varphi) = \sum_{j=1}^{N} (-1)^{j+1} \frac{r_{\ell_j}^3}{(r_{\ell_j}^2 + \epsilon^2)^n}, \tag{5.54}$$

where $\ell_j \in \mathbf{R}(\theta, \varphi)$ and N is even. Hence, from Lemma. 5.1.1 with $t = 3$ and $\epsilon \neq 0$,

- If $n \leq 3/2$ then $I(\theta, \varphi) < 0$ for each θ, thus $f(\mathbf{x}) < 0$;
- If $n > 3/2$ and $r_{\ell_N}(\theta, \varphi) < \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) < 0$. Let $r_{max} = \max\{r_{\ell_N}(\theta, \varphi)\}$ for all combination (θ, φ). If $r_{max} < \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) < 0$ for each (θ, φ), thus $f(\mathbf{x}) < 0$.
- If $n > 3/2$ and $r_{k_1}(\theta, \varphi) > \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) > 0$. Let $r_{min} = \min\{r_{\ell_1}(\theta, \varphi)\}$ for all combination (θ, φ). If $r_{min} > \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) > 0$ for each (θ, φ), thus $f(\mathbf{x}) > 0$.

When $\mathbf{x}$ is in the interior of $\mathbf{\Omega}$, $\text{sgn}(b_{\ell_j}) = (-1)^j$ where $\ell_j \in \mathbf{R}(\theta, \varphi)$.Therefore, we can rewrite Eq. 5.53 as

$$I(\theta, \varphi) = \sum_{j=1}^{N} (-1)^{j} \frac{r_{\ell_j}^3}{(r_{\ell_j}^2 + \epsilon^2)^n}. \tag{5.55}$$

where $\ell \in \mathbf{R}(\theta, \varphi)$ and N is odd. Hence, from Lemma. 5.1.2 with $t = 3$ and $\epsilon \neq 0$,

- If $n \leq 3/2$ then $I(\theta, \varphi) < 0$ for each θ, thus $f(\mathbf{x}) < 0$;
- If $n > 3/2$ and $r_{\ell_N}(\theta, \varphi) \leq \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) < 0$. We know $r_{max} = \max\{r_{\ell_N}(\theta, \varphi)\}$ for all combination (θ, φ). If $r_{max} \leq \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) < 0$ for each (θ, φ), thus $f(\mathbf{x}) < 0$.

- If $n > 3/2$ and $r_{\ell_1}(\theta, \varphi) \geq \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) < 0$. We know $r_{min} = \min\{r_{\ell_1}(\theta, \varphi)\}$ for all combination (θ, φ). If $r_{min} \geq \sqrt{3\epsilon^2/(2n-3)}$ then $I(\theta, \varphi) > 0$ for each (θ, φ), thus $f(\mathbf{x}) < 0$.

If $\mathbf{x}$ is on the input polyhedron then $\mathbf{x}$ is on a particular polygon $\Omega_{k'}$. The intersection of $\vec{r}$ with $\Omega_{k'}$ has no contribution to $\sum_{k=1}^{K} a_k$ because $b_{k'} = 0$ and $a_{k'} = 0$. Therefore, for every ray cast from $\mathbf{x}$ to $\Omega_{k'}$, the first intersection on $\Omega_{k'}$ is ignored. When the direction of the ray is towards the outside (inside) of $\mathbf{\Omega}$, it is the same case as when $\mathbf{x}$ is in the exterior (interior) of $\mathbf{\Omega}$. Therefore, for $\mathbf{x}$ on $\mathbf{\Omega}$, the sign of $f(\mathbf{x})$ is determined only in those cases when the interior and exterior have the same sign, i.e., when $n \leq 3/2$, $f(\mathbf{x}) < 0$; when $n > 3/2$ and $r_{max} < \sqrt{3\epsilon^2/(2n-3)}$, $f(\mathbf{x}) < 0$. □

Like in the 2D case, what Theorem 5.3.3 tells us is that for $n > 3/2$, there is an unbounded region outside of $\mathbf{\Omega}$ where f is strictly positive and when certain conditions are met, there is a bounded region inside $\mathbf{\Omega}$ where f is strictly negative. This leads to the following corollary.

Corollary 5.3.4. When using IMLS to approximate data ($\epsilon > 0$), if $n > 3/2$ and if there exists a point $\mathbf{x}$ inside of $\mathbf{\Omega}$ such that $r_{min} \geq \sqrt{3\epsilon^2/(2n-3)}$ or $r_{max} \leq \sqrt{3\epsilon^2/(2n-3)}$, then the resulting implicit function f is well defined.

Corollary 5.3.5. When using IMLS to approximate data ($\epsilon > 0$), if $n > 3/2$ and the input data is convex, the resulting implicit function f is well defined. In addition, the interior of $\mathbf{\Omega}$ is in the interior of f.

As in 2D, Theorem 5.3.3 tells us when f is well defined. Whether polyhedrons exist where f is not well defined when $n > 3/2$ is an open question.

Chapter 6

Gaussian Weighted IMLS Method from Polygon Soup

In this section, I derive a generalized implicit moving least squares method to interpolate or approximate polygon soup. My method uses a Gaussian weighting of polygons to reduce the bulges seen in the IMLS scheme of Shen et al. [34] (Fig. 1.3). According to the analysis in Section 5.2, the weight function of Shen et al. is well defined. Their method create reasonable curves and surfaces for both 2D and 3D cases when interpolating polygon soup with only a few small holes and gaps. However, for larger holes, the results produced by IMLS become unsatisfactory as can be seen from the middle column of Fig. 6.1, where 2D curves (middle) generated by IMLS suffer from serious bulge problems. The 2D curves (right) generated by my method do not have this bulging problem. The bulges appearing on the IMLS surfaces are a result of the negative contribution coming from opposing polygons further away that are non-negligible when there are bigger holes. This bulge occurs because the weight function used in IMLS is not decreasing steep enough as the distance from the query point $\mathbf{x}$ to the sample points $\mathbf{p}$ on the polygons increases. Straightforward solutions to this problem include either making the IMLS weight function $n > 2$ in Eq. 4.3 or forcing the weight to be non-zero only within a bounded interval. Unfortunately, both of these approaches increase the difficulty in solving the integration problem in Eq. 3.12. Using the original weight function in IMLS, Shen et al. [34] provided a closed form solution for a one-dimensional integral (i.e., constraints over edges) and an advanced quadrature method to gain good approximations for the two-dimensional integral. In the following, in an effort to reduce the bulges, I use Shen et al.'s weight function (Eq. 3.14) while providing a more generalized formulation to the surface construction problem that not only weights the points but also considers the distance from the query point to each polygon.

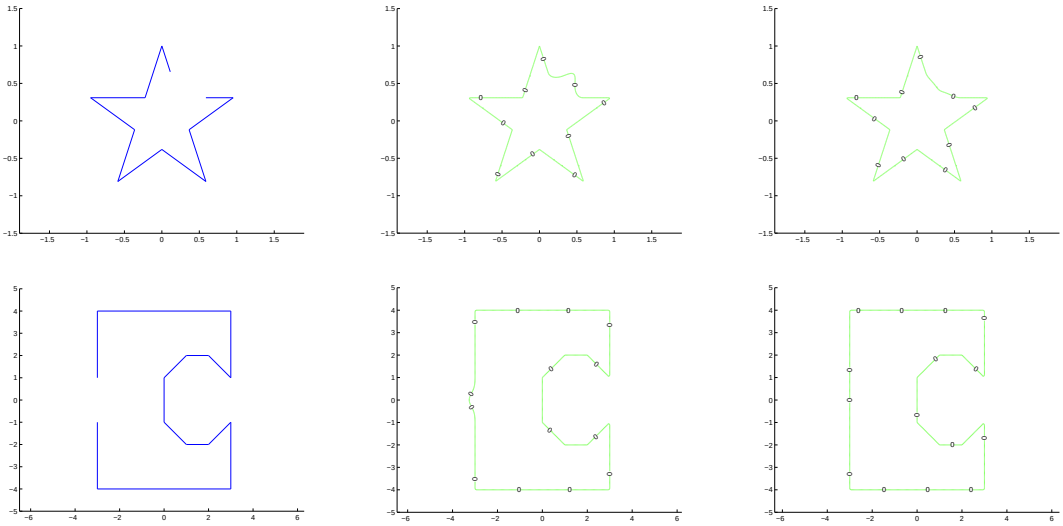


Figure 6.1: 2D interpolation results from IMLS and GIMLS from input data with gaps. Left column: Original Shapes in 2D; Middle column: IMLS results; Right column: the results of my GIMLS method with $\sigma = 0.2$. Zero contours are extracted by Matlab.

I define the generalized implicit moving least squares (GIMLS) as

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) S_k(\mathbf{x}) d\mathbf{p}_{\Omega_k}}{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}} , \tag{6.1}$$

$$w(\mathbf{x} - \mathbf{p}_{\Omega_k}) = \frac{1}{(\|\mathbf{x} - \mathbf{p}_{\Omega_k}\|^2 + \epsilon^2)^2}, \tag{6.2}$$

where $G_{\Omega_k}(\mathbf{x})$ describes the contribution from polygon Ω_k to the evaluated point $\mathbf{x}$ and $S_k(\mathbf{x}) = \phi + (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k}$.

In my generalized formulation, the point-based weight function w is used to ensure that details on each polygon are captured, and the polygon-based weight function G is used to emphasize the contribution from each polygon when it is close to the query point. I define the polygon-based weight to be Gaussian to make it controllable, i.e.,

$$G_{\Omega_k}(\mathbf{x}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{D^2(\Omega_k, \mathbf{x})}{2\sigma^2}}, \tag{6.3}$$

where $D(\Omega, \mathbf{x})$ is the distance from $\mathbf{x}$ to polygon Ω and σ is the variance of the Gaussian distribution that determines how quickly the contribution grows as the query point approaches the polygon.

6.1 Interpolation

The contribution from farther polygons is decreased when the variance parameter σ becomes larger. For interpolation, the σ parameter can be fixed. In my experiments, the value of σ in $[0.02, 0.04]$ produces satisfactory results. Throughout all of my experiments in this chapter, I set $\sigma = 0.03$. As σ goes to ∞, my method approaches IMLS and holes are filled with surfaces that bulge. Note that σ is scale dependent. If σ is too small relative to the size of the hole or gap, there will be numerical issues since the value of f will be closer to 0 then floating point number can represent. Otherwise, as we decrease σ, the filled corners or edges get sharper.

It is clear that my method in Eq. 6.1 is a re-weighting of IMLS in Eq. 4.2. The re-weighting function is controllable to make my method a generalized IMLS method. The 3D examples in Fig. 6.2[1] show that my method not only captures features of the polygons but also make each polygon contribute in a more reasonable way.

6.2 Approximation

In this section, I discuss using my GIMLS method to approximate data.

6.2.1 Self-intersection Problem

Neither IMLS nor my approach can handle the self-intersection problem while interpolating the data. For surface approximation, both IMLS and my method incorporate the weight function with a small parameter ϵ in Eq. 4.3 to approximate the polygon soup. While IMLS removes self-intersections, in my experiments shown in Fig. 6.3(c), using a fixed Gaussian weight for each polygon for GIMLS does not resolve the self-intersection problem. However, in the following I will give an example where GIMLS removes self-intersections

[1]Bloomenthal polygonization is used for visualization in Fig. 6.2, and although we use high polygonization resolution, small aliasing defects along sharp features can also be seen. This problem can be solved by numerous post-processing mesh optimization techniques proposed in the past decade [12, 15, 27].

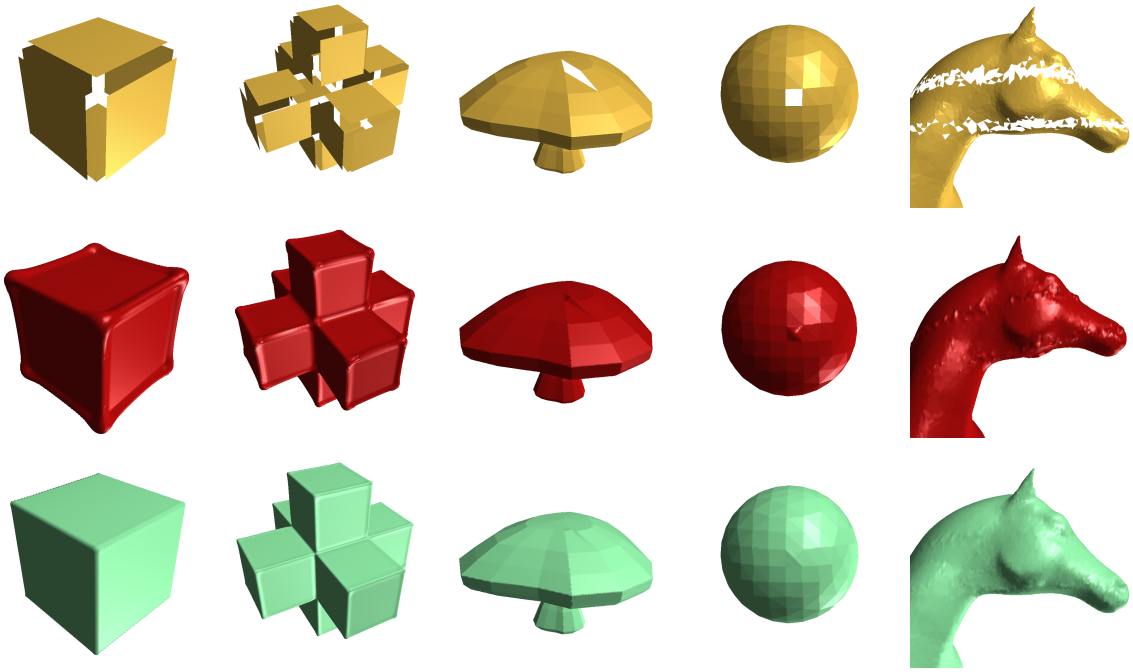
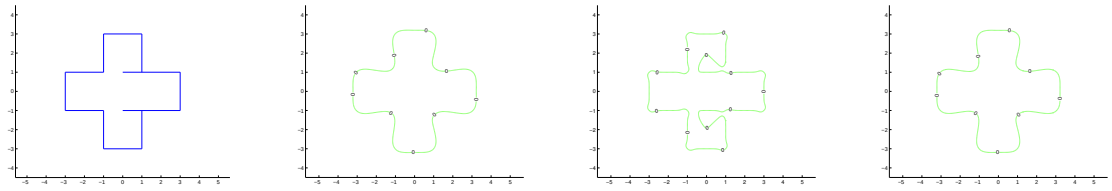


Figure 6.2: 3D interpolation reconstruction comparison between IMLS and GIMLS. First Row: original polygon data with holes or gaps; Second Row: IMLS; Third Row: my method.

when approximating by using an adaptive Gaussian variance σ. In particular, I define the polygon-based weight in approximation as

$$G_{\Omega_k}(\mathbf{x}) = \frac{1}{(\sigma + \lambda\epsilon)\sqrt{2\pi}} e^{-\frac{D^2(\Omega_k, \mathbf{x})}{2(\sigma+\lambda\epsilon)^2}}, \tag{6.4}$$

where I experimentally found that $\lambda = 8$ works well, and I used the same value for $\sigma = 0.03$ that I use for interpolation, and I set ϵ to a small, positive value to approximate the polygon soup. As ϵ increases, the Gaussian weight function becomes flatter which gradually reduces the contribution from closest polygons and prevents them from dominating the implicit function values. Results for my method are shown in Fig. 6.3(d). The underlying idea of my GIMLS is that when we interpolate the data, the implicit function has to emphasize local regions, and when we approximate the data, the implicit function need not to be as exact as in interpolation and thus the contribution to the query point should be more diverse to be robust and to eliminate outliers.



(a) (b) (c) (d)

Figure 6.3: Surface reconstruction with self-intersections. (a) Original data (b) IMLS approximation with $\epsilon = 0.8$ (c) GIMLS approximation with fixed σ (d) GIMLS approximation with adaptive $\sigma = 0.2 + 8\epsilon$.

6.2.2 Behavior of the Approximation

When $\epsilon \neq 0$ in the weight function in Eq. 4.3, the IMLS function approximates the constraint values. However, setting ϵ to a non-zero value suffers from two problems. The first is that as ϵ grows larger, the approximating surface has the tendency to move away from the input data. The second problem is that Shen et al. wanted all the data to be on the interior of the approximating surface. However, Shen's scheme cannot ensure this. Shen et al. used an ϵ first, sampled the average implicit value over the surface of the input polygons, and then extracted a surface at that iso-value to solve the first problem.

Park et al. [29] showed why Shen et al.'s method cannot include all the object's original vertices. They proved that near concavities, approximation surfaces based on Shen et al.'s original work fail to include the original model fully. Park et al. also proposed an alternative heuristic to approximate the data. They first construct the interpolating implicit surface and then they select a positive value iso-surface for their approximation. This method guarantees that all the vertices and polygons are inside the surface envelop of the approximation. However, IMLS with this approximation scheme is unable to heal self-intersections in the polygon soup and forms a poor approximation to the data as it is "strictly larger" than the desired surface.

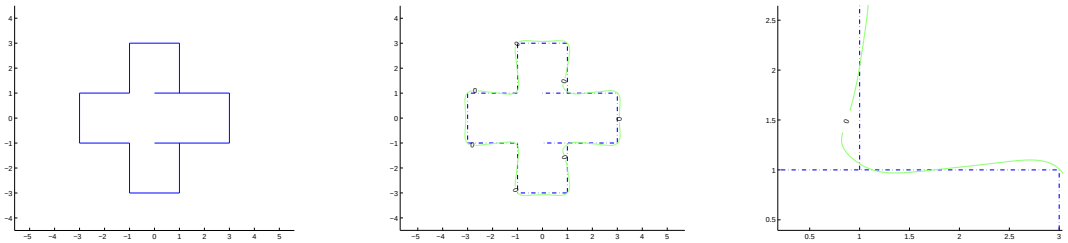
We can use either approach with my method (and indeed, Shen et al.'s heuristic could be used with Park et al.'s method, and Park et al.'s heuristic could be used with Shen et al.'s method).

Regardless, Fig. 6.4(b) shows an example of my method approximating data $\epsilon = 0.5$. Fig. 6.4(c) is an enlargement of Fig. 6.4(b). It is clear that some line segments and vertices

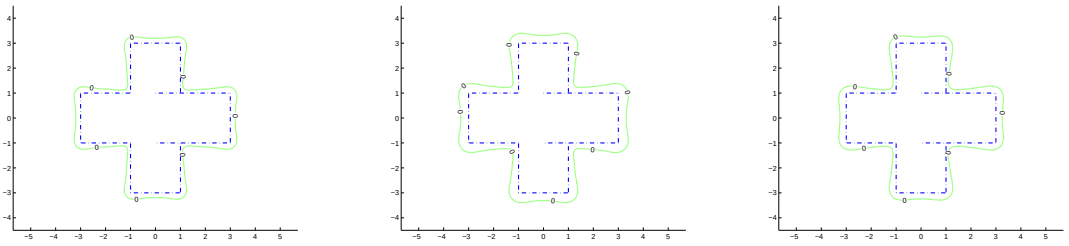
are not on the interior of the generated implicit surface. Using the same implicit surface, we can extract a different iso-surface as

$$f'(\mathbf{x}) = f(\mathbf{x}) - h, \tag{6.5}$$

where $f(x)$ is from Eq. 4.2 and h is the particular iso-surface. Fig. 6.4(d) is the result with $\epsilon = 0.35, h = 0.15$, Fig. 6.4(e)e has $\epsilon = 0.35, h = 0.25$, and Fig. 6.4(f) has $\epsilon = 0.5, h = 0.15$. Some 3D approximation examples are shown in Fig. 6.5.



(a) (b) (c)



(d) (e) (f)

Figure 6.4: Results of GIMLS approximation with different parameters. (a): original self-intersection input; (b): GIMLS approximation with $\epsilon = 0.5, h = 0$; (c): Enlarged view of the left part of (b); (d): GIMLS approximation with $\epsilon = 0.35, h = 0.15$; (e): GIMLS approximation with $\epsilon = 0.35, h = 0.25$; (f): GIMLS approximation with $\epsilon = 0.5, h = 0.15$.

Figure 6.5: 3D approximating reconstruction. First Row: interpolation results; Second Row: approximation results with $\epsilon = 0.02$. Last Row: approximation results with $\epsilon = 0.02$ and $h = 0.08$

.

6.2.3 Correctness of GIMLS

In Chapter 5, I showed when the implicit function constructed by IMLS was well-defined. We can ask a similar question about GIMLS. In essence, adding the Gaussian weight function to IMLS is similar to increasing n in Eq. 5.3. From the proof in Chapter 5, we see that GIMLS is well-defined in all cases that IMLS is well-defined. Further some cases that do not work for IMLS might work for GIMLS. For example, when interpolating with $n = 1$ in Eq. 5.3, IMLS gives all zero values in the exterior, but GIMLS gives well-defined implicit functions.

6.3 Numerical Integration

To implement my method, I had to compute the same integrals that Shen et al. did. In implementing Shen et al.'s numerical solution to these integrals, I encountered numerical issues. In this section, I give an alternative method for numerically evaluating these integrals that is numerically stable.

Shen [33] presented a closed form solution for one-dimensional integrals over line segments in Eq. 5.21. Unfortunately, there is no analytic solutions of the integral MLS for polygon soups in 3D. And while Park et al. claim to have a solution, their solution is incorrect as shown in Section 4.2.2. In 3D, the implicit function in Eq. 4.2 was simplified by Shen to

$$f(\mathbf{x}) = \frac{Num_0(\mathbf{x}) + \mathbf{x}^\intercal \cdot Num_x(\mathbf{x})}{Den(\mathbf{x})}. \tag{6.6}$$

where

$$Den(\mathbf{x}) = \sum_{k=1}^{K} \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}, \tag{6.7}$$

$$Num_0(\mathbf{x}) = \sum_{k=1}^{K} \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) \phi_0 d\mathbf{p}_{\Omega_k}, \tag{6.8}$$

$$Num_x(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{n}_{\Omega_k} \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} \tag{6.9}$$

where ϕ_0 and $\mathbf{n}_{\Omega_k}$ is derived from normal function defined in Eq. 4.1, for the kth polygon:

$$\begin{aligned} S_k(\mathbf{x}) &= \phi + (\mathbf{x} - \mathbf{p}_{\Omega_k})^\intercal \mathbf{n}_{\Omega_k} \\ &= (\phi - \mathbf{n}_{\Omega_k}{}^\intercal \mathbf{p}_{\Omega_k}) + \mathbf{n}_{\Omega_k}{}^\intercal \mathbf{x} \\ &= \phi_0 + \mathbf{n}_{\Omega_k}{}^\intercal \mathbf{x}. \end{aligned}$$

The numerical integration problem for this implicit function is to solve

$$Num_1(\mathbf{x}) = \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} \tag{6.10}$$

$$\text{and} \quad Num_2(\mathbf{x}) = \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) \phi_0 d\mathbf{p}_{\Omega_k}. \tag{6.11}$$

By parameterizing each triangular region with its three vertices, in Chapter 5.4 of Shen's dissertation, he derived a closed form solution for the one-dimensional integrals to calculate the inner layer of the above two-dimensional integrals, which is I_{i0} and I_{i1} in Shen's dissertation Eq. 5.70 and Eq. 5.71. However, the outer layers in I_{i0} and I_{i1} are not solved with analytic solutions, as shown in Eq. 5.93 in Shen's dissertation. Because the weight function is a radial basis function, the integrands have near-singularity problems where the query point approaches the current evaluated polygon. The shape of an inner layer integral is shown in Fig. 6.6. Due to the near-singularities property, standard quadrature

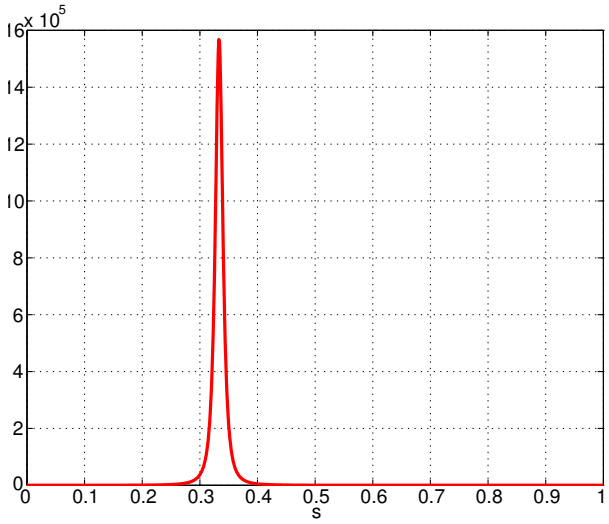


Figure 6.6: Shape of the inner layer solution for the two dimensional integration over 3D polygons

approximation methods like the trapezoid rule are not applicative; they perform poorly for the same reason that sampling finite point constraints does: unless the distance between quadrature points is significantly less than ϵ the resulting surface will have dimples and bumps.

Shen proposed a special quadrature scheme in Chapter 5.4.3 in his dissertation by transforming the integrands with singularities into flat ones using a change of coordinates. Directly transforming the complicated solution of the inner layer integration is infeasible, and Shen approximates it by a function that can capture the near-singularities and also has a closed-form integral. Shen choose to approximate the inner integrals shown in Fig. 6.6 with

$$ga(u) = \frac{1}{c_2 u^2 + c_1 u + c_0} \ , \tag{6.12}$$

where c_0, c_1 and c_2 are unknown. To solve for c_0, c_1 and c_2, Shen sampled three points u_-, u_m and u_+. The point u_m is the near-singularity point with the maximum function value of the integrand. u_- and u_+ are two points near u_m with small offsets. However, if u_- and u_+ are far from each other, the near-singularity is not caught; if they are too close to each other, the linear system to determine c_0, c_1 and c_2 in Eq. 6.13 is ill-conditioned:

$$\begin{bmatrix} s_-^2 & s_- & 1 \\ s_m^2 & s_m & 1 \\ s_+^2 & s_+ & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{i0}(s_-)} \\ \frac{1}{I_{i0}(s_m)} \\ \frac{1}{I_{i0}(s_+)} \end{bmatrix} \tag{6.13}$$

After implementing this method, I found that was unstable. I substituted a simpler method for Shen's special quadrature scheme to avoid the above approximation step. It requires three steps:

1. Find the closest point from the evaluation point to each polygon mesh and make it the approximation of the singular point.
2. From the singular point to each boundary, sample points on an exponentially decreasing interval.
3. Use the unevenly sampled points to do trapezoidal approximation.

This gives similar results as Shen et al.'s method and also avoids the instability of Eq. 6.13.

6.4 Efficient Evaluation with kD-trees

Naive implementation of the implicit function in my method would require computing the sums and integrals in Eq. 6.1 on every input polygon. For large polygon soup, this naive implementation is infeasible. Shen et al. [34] revised the hierarchical evaluation scheme applied in the partition-of-unity method proposed by Ohtake et al. [26] to solve this problem. The primary cost of evaluating Eq. 6.1 is to calculate the integrals of weight functions over each polygon. If current polygon is near the query point, the weight function changes rapidly. However, the weight function changes slowly for far away polygons. Shen et al. approximated groups of the slowly changing far integrals by first summing them and then multiplying by their average weight.

Since GIMLS is a generalization of IMLS, the kD-tree hierarchical scheme used in IMLS is also applicable to my method. By checking distances from the query point to a bounding

box of the kD-tree nodes, groups of far-away polygons are approximated as a unit. The integration in Eq. 6.1 of far-away groups is

$$\begin{aligned}\int_{\Omega_{far}} w^2(x - p_{\Omega_{far}})dp_{\Omega_{far}} &\approx \int_{\Omega_{far}} w^2(x - p_{centre})dp \\ &= w^2(x - p_{centre})Area_{far},\end{aligned}$$

where p_{centre} is the center of the bounding box far-away group of polygons. In Eq. 6.1, the result of this integration is also multiplied by a Gaussian weight function.

Another acceleration approach is to remove polygon surfaces that are farther away from the query point because the Gaussian weight assigned to each polygon is exponentially decaying when this distance increases. More formally, a polygon Ω_k that satisfies

$$\frac{d(\mathbf{x}, \Omega_{nearest})}{d(\mathbf{x}, \Omega_k)} < \zeta \tag{6.14}$$

is discarded in computing the implicit surface function, where $\Omega_{nearest}$ is the nearest polygon to the query point $\mathbf{x}$, d the distance function and ζ a threshold.

Care must be taken when approximating f, however, since we are interested in $\mathbf{x}$ such that $f(\mathbf{x}) = 0$. Potentially, discarding small terms could drastically change the zero set. More formally, define $\hat{f}(\mathbf{x})$ as an approximation to $f(\mathbf{x})$. To prove that $\hat{f}(\mathbf{x})$ is geometrically close to $f(\mathbf{x})$, a fundamental condition is that when $f(\mathbf{x}) = 0$, there must exists $\hat{\mathbf{x}}$, $\hat{f}(\mathbf{x}) = 0$ and $\|\mathbf{x} - \hat{\mathbf{x}}\| < \varepsilon$. However, while necessary, this condition only guarantees the implicit value closeness of the approximated implicit surface $\hat{f}(\mathbf{x})$, and the topological closeness of $\hat{f}(\mathbf{x})$ is not ensured. Regardless, neither Shen et al. nor I proved that our approximations have this property. For my straightforward acceleration scheme, there are several situations it might fail. The first is when there are a large number of polygons considered as far-away; their individual contributions could be small but their summation could be large and should not be neglected; the second is when the gradient of the implicit value on the evaluated point is small, in which case removing far-away polygons could lead to significant changes to the zero set.

6.5 GIMLS with a Distance Function

While GIMLS reduces the bulges seen in Shen et al.'s method when filling holes, it does a poor job of filleting and rounding. In this section, I discuss some ideas I tried to improve the fillets and rounds, although none gave satisfactory results. As shown in Fig. 6.2, GIMLS

fills gaps with sharp edges or corners, especially in the example of the cube and the 3D cross. Shen et al.'s method fills gaps with bulges. However, neither is sufficient to fill the gaps or holes with rounded edges and corners.

The sharp corner computed by GIMLS is due to the property of the shape function in Section. 4.1.1. The shape function calculates the distance from query points to the tangent plane of input data. An unsigned distance function that takes the nearest distance from query points to the polygons instead of tangent plane should reduce the sharpness. There are many different ways to add such a unsigned distance function.

Among my experiments, the following method performed better than others on 2-dimensional cases. This distance function is the minimum distance from $\mathbf{x}$ to the segment Ω_k, represented as $Dist(\mathbf{x}, \Omega_k)$. The implicit function is

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x})(S_k(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} + \lambda Dist(\mathbf{x}, \Omega_k))}{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}}. \tag{6.15}$$

The 2-dimensional results with two different λ are show in Fig. 6.7, where we see that the sharp corners are rounded as we desire.

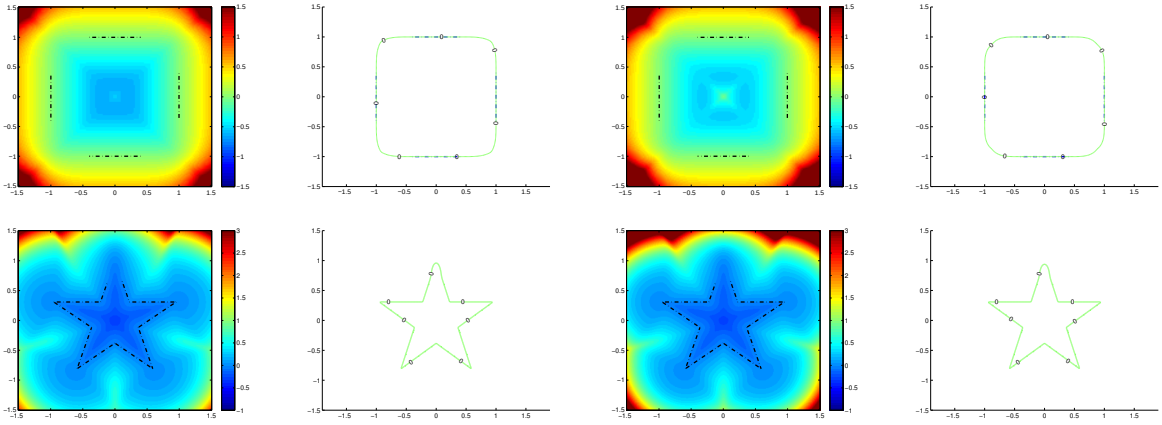


Figure 6.7: 2D examples for GIMLS combined with distance function. Left two:$\lambda = 1/3$; Right two: $\lambda = 2/3$

However, the 3-dimensional results were unsatisfactory. As shown in Fig. 6.8, the sharp edges and corners are not smoothed. I tried two other ways to embed distance functions, but their results were even worse. The two other methods I tried were:

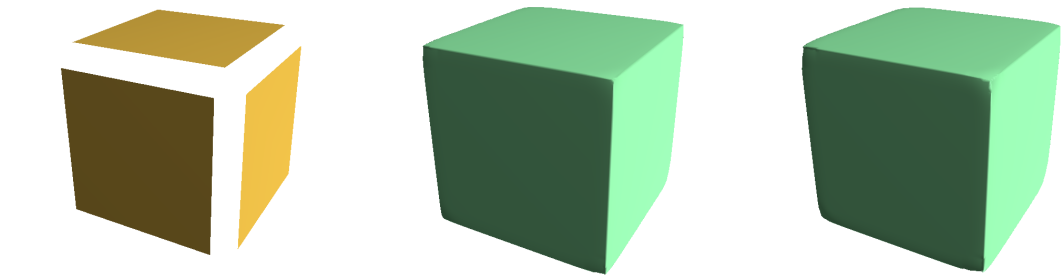


Figure 6.8: 3D examples for GIMLS combined with distance function. Left: Original Data; middle: $\lambda = 1/3$; right: $\lambda = 2/3$.

1. Embed the distance function on the final implicit surface:

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) S_k(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}}{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}} + \lambda Dist(\mathbf{x}, \Omega_k)$$

2. Weight the distance function with Gaussian and radial basis functions:

$$f(\mathbf{x}) = \frac{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k} (S_k(\mathbf{x}) + \lambda Dist(\mathbf{x}, \Omega_k))}{\sum_{k=1}^{K} G_{\Omega_k}(\mathbf{x}) \int_{\Omega_k} w(\mathbf{x} - \mathbf{p}_{\Omega_k}) d\mathbf{p}_{\Omega_k}}$$

However, according to my experiments, simply incorporating the above-mentioned distance function does not result in rounded edges in the re-constructed surfaces. The distance function exploration is a topic for future work.

Chapter 7

Conclusion and Future Work

I analyzed the moving least squares method of Shen et al. [34] and discussed why they squared the original weight function. I also pointed out that the analytic method of Park et al. [29] is incorrect because of their adjustment to the weight function. In addition, I presented a rigorous proof of necessary conditions on the weight functions of IMLS both in 2D and 3D. Based on the understanding of moving least squares and implicit surface reconstruction, I provided a new generalized method that can interpolate polygon soup with holes and gaps better than previous interpolation methods based on polygonal data.

I only proved the necessary conditions on the weight functions for IMLS in 2D and 3D. However, by extending the solid angle into N-dimension, we should be able to generalize the proof to work in N-dimensions. This is a topic of future research.

My proof for choosing weight functions of IMLS requires the polygonal curve and polyhedral surface to be manifold. When polygon soup contains holes and self-intersections, my proof is not applicable. Another avenue for future research is to extend my proofs to polygon soup. This will likely be non-trivial since polygon soup is usually not closed, and thus it is hard to distinguish the inside from the outside.

Shen et al. and I have two methods to accelerate the implicit function evaluating process: one is approximating far-away segments by their central points and the other is discarding far-away segments directly. However, neither Shen et al. nor I gave a theoretical proof to show that the two methods will generate a surface that is geometrically and topologically close to the original surface. This is another topic that requires further exploration.

While objects are often modeled as polyhedral models, when manufactured, the edges between polygons need to be rounded, and the creases filled. My GIMLS method decreases

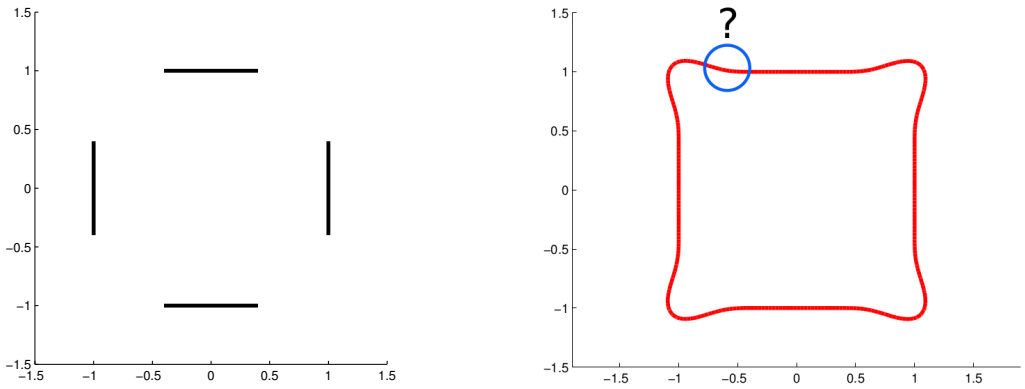


Figure 7.1: Continuity.

the bulges in the previous IMLS method but the ideal fillets and rounds algorithm for filling holes and gaps are still an open problem.

Another possible idea to reduce bulges is to use the geodesic distance instead of Euclidean distance in the weight function. This will make input data geometrically far away contribute less. However, for polygon soup, the geodesic distance is hard to compute.

When interpolating with IMLS or GIMLS in 3D, if there is a polygon that does not have neighbors on two consecutive edges, then at the common vertex, the surface is not curvature continuous. However, we do not know if it is tangent plane continuous. More investigation needs to determine the continuity. A similar question arises in 2D, where we would like at least tangent continuity between input line segments and the constructed hole filling implicit curves; see Fig. 7.1

For $n = 1$ and $\epsilon = 0$, Park et al.'s method builds an implicit function that is 0 at every point in the exterior of the polygon, shown in Fig. 4.2. While this makes the method unsuitable for data interpolation and approximation, note that this is a function that has compact support. Potentially, such a function is useful for other applications.

Like most of functional interpolation and approximation methods, the quality of my method degrades when the sampling rates are too low (i.e., when the holes and gaps are too big) or the parameters are extreme values. In particular, if the polygons are all missing in a certain direction (like the bottom or top of an object for which there is no normal information to use to fill the holes), my method will produce unsatisfactory results. Further research is need on this problem. For approximation, I found experimentally that my hybrid method can solve most of the problems with polygon soup, but I have not yet

derived a rigorous theory to support this.

References

[1] Chandrajit Bajaj. *Introduction to implicit surfaces*. Morgan Kaufmann, 1997.

[2] Chandrajit L Bajaj, Jindon Chen, and Guoliang Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics (TOG)*, 14(2):103–133, 1995.

[3] Robert E Barnhill. Representation and approximation of surfaces. *Mathematical software*, 3:69–120, 1977.

[4] James F Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982.

[5] William Chang. *Surface reconstruction from points*. Department of Computer Science and Engineering, University of California, San Diego, 2008.

[6] William S Cleveland and Clive Loader. Smoothing by local regression: Principles and methods. In *Statistical theory and computational aspects of smoothing*, pages 10–49. Springer, 1996.

[7] Tamal K Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2007.

[8] João Paulo Gois, Diogo Fernando Trevisan, Harlen Costa Batagelo, and Ives Macêdo. Generalized hermitian radial basis functions implicits from polygonal mesh constraints. *The Visual Computer*, pages 1–11, 2013.

[9] Abel Gomes, Irina Voiculescu, Joaquim Jorge, Brian Wyvill, and Callum Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms: Mathematics, Data Structures and Algorithms*. Springer, 2009.

[10] William J Gordon and James A Wixom. Shepards method of metric interpolation to bivariate and multivariate interpolation. *Mathematics of Computation*, 32(141):253–264, 1978.

[11] CRAIG Gotsman and DANIEL Keren. Tight fitting of convex polyhedral shapes. *International Journal of Shape Modeling*, 4(3/4):111–126, 1998.

[12] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 339–346. ACM, 2002.

[13] Takashi Kanai, Yutaka Ohtake, and Kiwamu Kase. Hierarchical error-driven approximation of implicit surfaces from polygonal meshes. In *Symposium on geometry processing*, pages 21–30. Citeseer, 2006.

[14] Chun Liu Khodakhast Bibak, Hamideh Vosoughpour, Grace Yao, Zainab AlMeraj, Alex Pytel, William Cowan, and Stephen Mann. Implicit surfaces seminar, spring 2012. 2013.

[15] Leif P Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66. ACM, 2001.

[16] Ravikrishna Kolluri. Provably good moving least squares. *ACM Transactions on Algorithms (TALG)*, 4(2):18, 2008.

[17] Peter Lancaster. Moving weighted least-squares methods. *Polynomial and Spline Approximation*, 49:103–120, 1979.

[18] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155):141–158, 1981.

[19] Weitao Li, Yuanfeng Zhou, Caiming Zhang, and Xuemei Li. Robust multi-level partition of unity implicits from triangular meshes. *Computer Animation and Virtual Worlds*, 2013.

[20] W. F. Long. Construction of a solid angle. 1992.

[21] Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. *Unified simulation of elastic rods, shells, and solids*, volume 29. ACM, 2010.

[22] Dermot H McLain. Drawing contours from arbitrary data points. *The Computer Journal*, 17(4):318–324, 1974.

[23] Jai Menon and Baining Guo. *Free-form Modeling with Low Degree Algebraic Patches in Bilateral Brep and CSG Schemes*. IBM TJ Watson Research Center, 1995.

[24] Jai Menon and Baining Guo. *A framework for sculptured solids in exact CSG representation*. IBM TJ Watson Research Center, 1996.

[25] Jai Prakash Menon. Constructive shell representations for freeform surfaces and solids. *Computer Graphics and Applications, IEEE*, 14(2):24–36, 1994.

[26] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM SIGGRAPH 2005 Courses*, page 173. ACM, 2005.

[27] Yutaka Ohtake and Alexander G Belyaev. Dual/primal mesh optimization for polygonized implicit surfaces. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 171–178. ACM, 2002.

[28] A Cengiz Öztireli, Gaël Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, volume 28, pages 493–501. Wiley Online Library, 2009.

[29] Taejung Park, Sung-Ho Lee, and Chang-Hun Kim. Analytic solutions of integral moving least squares for polygon soups. *Visualization and Computer Graphics, IEEE Transactions on*, 18(10):1638–1649, 2012.

[30] Alexander Pasko, Valery Adzhiev, Benjamin Schmitt, and Christophe Schlick. Constructive hypervolume modeling. *Graphical models*, 63(6):413–442, 2001.

[31] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[32] Alyn P Rockwood. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics (TOG)*, 8(4):279–297, 1989.

[33] Chen Shen. *Building Interpolating and Approximating Implicit Surfaces Using Moving Least Squares*. PhD thesis, EECS Department, University of California, Berkeley, Jan 2007.

[34] Chen Shen, James F O'Brien, and Jonathan R Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 896–904. ACM, 2004.

[35] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.

[36] Moshe Shimrat. Alogrithm 112: position of point relative to polygon. *Communications of the ACM*, 5(8):434, 1962.

[37] Gabriel Taubin. An improved algorithm for algebraic curve and surface fitting. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 658–665. IEEE, 1993.

[38] Greg Turk and James F O'brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses*, page 13. ACM, 2005.

[39] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge University Press Cambridge, 2005.

[40] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, volume 18, pages 149–158. Wiley Online Library, 1999.

[41] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Soft objects. In *Advanced Computer Graphics*, pages 113–128. Springer, 1986.

[42] Hongyi Xu and Jernej Barbič. Signed distance fields for polygon soup meshes. *Graphics Interface 2014*, 2014.

[43] Oscar Zariski. *Algebraic surfaces*, volume 61. Springer, 1995.