Tese apresentada para obtenção do título de Doutor em Matemática pelo
Instituto Nacional de Matemática Pura e Aplicada

# On Sketches for Modeling

por

Emilio Ashton Vital Brazil

Orientador: Luiz Henrique de Figueiredo
Co-orientador: Mario Costa Sousa

Rio de Janeiro
22 de Fevereiro de 2011

# On Sketches for Modeling

Emilio Ashton Vital Brazil

Advisor: Luiz Henrique de Figueiredo
Co-advisor: Mario Costa Sousa

February 22, 2011

# Agradecimentos

A Cristina, Dinah e Gelia, pelo amor e paciência dedicados a mim. A Seu Genaro que primeiro me apresentou a matemática e depois incentivou e deu os ensinamentos para chegar ao IMPA. A Danilo Felizardo Barbosa e Alan Almeida que junto com Seu Genaro me mostraram a beleza da matemática. A meu irmão Ashton, meus sogros Josias e Valdelina e Dona Arlete que sempre me apoiaram. A Arthur Rodrigues, Laura Barreira e Judy Galper que acolheram à minha família e a mim em todos esses anos que passamos no Rio de Janeiro. A Raimunda e Gordon Jackson e Patricia Medici, que foram as nossas referências de família e nos deram muito apoio enquanto estávamos no Canadá.

Aos professores, Luiz Henrique de Figueiredo, que me orientou não só nesse trabalho como em diversas outras situações; Mario Costa Sousa que além de me orientar me mostrou as belezas do Canadá e assim como Luiz Henrique sempre me apoiou; Luiz Velho que foi um grande colaborador e sempre me deu excelentes conselhos e dicas de pesquisa; Paulo Cezar Carvalho pelas ótimas aulas e conversas sobre diversos temas em Computação Gráfica e afins e pela possibilidade de convívio com o grupo da Fundação Getulio Vargas.

Aos meus colegas do VISGRAF, que muito me ajudaram, especialmente, Ives Macêdo, Marcelo Cicconet e Thiago Pereira pelas colaborações e Francisco Ganacim pelas frutíferas discussões sobre este trabalho.

A todos eficientes e prestativos funcionários do IMPA, em especial Djalma Lúcio e Dion Villar que estiveram mais próximos porém sem esquecer dos demais.

Ao Doutor Fritz Kahn por sua experiência e compaixão. E a todos os excelentes amigos que fiz no IMPA.

# Resumo

Este trabalho consiste em um estudo sobre o uso de desenho a mão livre como entrada de sistemas de modelagem, i.e., Sketch-based modeling (SBM). Especificamente, focamos este trabalho nas representações matemática para os sistemas SBM. A representação matemática do modelo desempenha um papel central neste problema sendo desenvolvida especialmente para uso em aplicações.

Desenvolvemos um campo de deformação desenhado a mão livre que satisfaz as restrições exigidas pela aplicação em deformação de imagem RGBN. Este problema "como deformar uma imagem usando esboços" é reformulado para "como modelar um campo de deformação usando linhas". Desenvolvemos então ferramentas para definir campos de deformação usando funções de base radial de Hermite-Birkhoff. Além disso, apresentamos uma aplicação que usa essas ferramentas para deformar imagens RGBN.

A utilização de superfícies implícitas, que fornecem uma representação compacta, flexível e matematicamente precisa, pode beneficiar sistemas SBM para protótipos. Desenvolvemos um conjunto de operadores de modelagem adequados para criar amostras de Hermite (pontos e normais). Tais amostras são interpoladas por funções de base radial de Hermite (HRBF) definindo assim uma superfície implícita. Os operadores são implementados em um aplicativo de modelagem de superfície que tem como entrada desenhos a mão livre, modelando com sucesso superfícies implícita variacionais HRBF, interpolando as curvas delineadas e preservando assim a forma desenhada.

Introduzimos uma nova representação matemática para o problema de modelagem baseada em desenho a mão livre, a qual abstrai as qualidades principais desejadas para muitas aplicações nas quais o desenho é utilizado como entrada para a modelagem. Além disso, desenvolvemos um sistema SBM com base na representação matemática, que concretamente implementa uma *pipeline* adaptada à utilização da teoria desenvolvida.

**Palavras-Chave**: modelagem a mão livre; representação de superfícies; campo de deformação; deformação de imagens; superfícies implícitas.

# Abstract

We present a study on sketch as input for modeling systems (i.e., sketch-based modeling, or SBM), focusing on mathematical representations for sketch-based modeling system. The representation of the model plays a fundamental role in this problem, requiring the underlying representations to be specially tailored for use in SBM application.

We develop a sketch warping field which satisfies the hard constrains applied to RGBN images deformation. This problem of "how to deform an image using sketches" is reformulated as "how to model a warping field". Therefore we developed tools for sketch-based modeling to define warping fields using Hermite–Birkhoff Radial Basis Functions. In addition to this, we also developed an application using these tools for warping RGBN images.

Prototype free-form SBM systems can also benefit from using implicit surfaces, which provide a compact, flexible, mathematically precise representation. We develop a set of modeling operators suited to create Hermite samples (points and normals), which are interpolated by a Hermite Radial Basis Function, defining an implicit surface. These operators are implemented in a sketch-based surface modeling application, which successfully models Variational HRBF Implicit surfaces by interpolating the sketched curves (thus preserving the intended shape to be modeled).

We introduce a novel mathematical representation for sketch-based modeling, abstracting main qualities desired for many SBM applications. We also developed a SBM system based on the proposed mathematical representation, effectively implementing a pipeline tailored to use the developed theoretical framework.

**Keywords**: sketch-based modeling; surfaces representation; warping field; image warping; implicit surface.

# Contents

# List of Figures

# Chapter 1

# Introduction

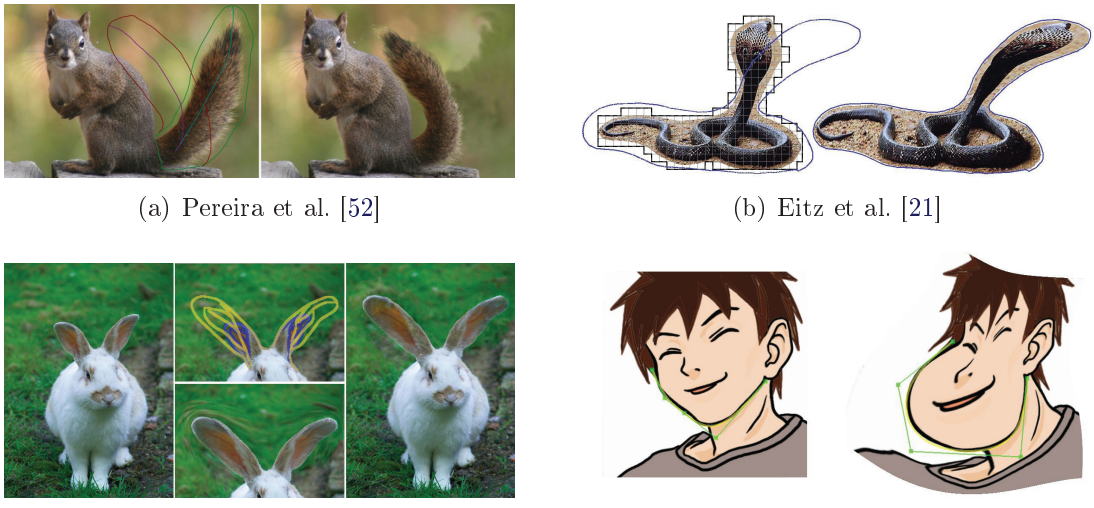
Sketch-based modeling (SBM) is a solid research area with many exciting problems on different domains such as: computer vision, human-computer interaction, and artificial intelligence (Olsen et al. [49]). Sketches are the most direct way to communicate shapes; humans are able to associate complex shapes with few curves (Figure 1.1, left). However sketches do not have all shape information and this information is often inexact thus ambiguities are natural: "My drawing was not a picture of a hat. It was a picture of a boa constrictor digesting an elephant." wrote de Saint-Exupéry [17, The little prince.] (Figure 1.1, right). On the other hand, to create, edit or visualize shapes using computers we need precise information such as a function formula or a triangle list. The problem of how modeling using sketches can be formulated: how to fill the missing information about the model, when sketches are used to define it. There are many approaches for this problem and each of them uses one or more specific knowledge areas. We focus this work on the study of mathematical representations for models of sketch-based modeling systems.


**Figure 1.1:** Sketches and shapes: it is innate for a human to identify the left drawing as a mug despite much missing information. On the other hand the sketch on the right can have many interpretations.

The main purpose of this work is to study sketches as input for modeling systems, focusing on mathematical representations. The mathematical representation of the model plays a central role in this problem, and therefore the representation should be developed specially for use in SBM applications. There are common requirements in many SBM applications that can be abstracted to guide the definition of specific representation for specific domains.

## 1.1 Related Work

In Chapter 2 we present the problem on how to deform image with control, reformulated as: how create a warping field using sketches. Our approach was successfully implemented to deform RGBN images in Pereira et al. [52]. There are other three recent works that present similar ideas: Eitz et al. [21], Fang and Hart [22], and Weng et al. [71] (Figure 1.2). Eitz et al. [21] present a method to find the matching between two given closed sketched curves. They use a quad mesh over the original image and deform it using a energy minimization to find the matching. After that, they apply a texture mapping with bilinear interpolation and then they compose the deformed image with the final target image using Poisson cloning. This approach seems to result in smooth warping fields. However, it provides little control and smoothness when trying to define a region of interest (ROI) for deformation. In our sketch-based interface anything outside the ROI defined by the user is left unchanged. Our system builds a warping field which is $C^1$ even on the desired part of the border of the ROI (Figure 1.2(a)). Fang and Hart [22] propose an warping image system by re-synthesizing texture from the source image. The main goal is to preserve the source image detail and orientation around a new feature curve location. The user sketches curves over the original image and deforms these curves to communicate the desired image warping. To achieve good results using this approach the user needs to handle many curves that can be awkward. On the contrary our system defines the warping field with few curves. Another problem is that the resulting field is $C^0$ on the sketches. Weng et al. [71] extended the work of Igarashi et al. [31] to handle cubic B-splines and all implementation is done in CPU reaching an interactive system. The user sketches two set of curves: the source and target, the system use these sets to define a field that satisfy these curve constrains. These fields are global and do not allow to create a ROI without creating discontinuities, thus in order to control the deformation the user needs to draw many lines.



(a) Pereira et al. [52]

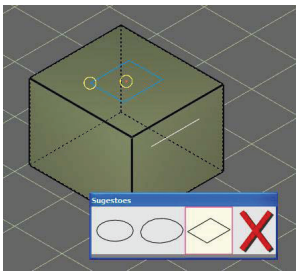
(b) Eitz et al. [21]

(c) Fang and Hart [22]

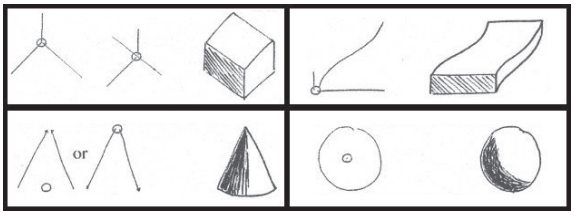
(d) Weng et al. [71]

**Figure 1.2:** Image deformation systems that use sketches.

In Chapter 3 we implement a SBM system of implicit surfaces. An important class of sketch-based interface and modeling (SBIM) systems is known as *constructive* SBIM systems, which directly map a set of 2D sketched input strokes to a 3D model without any previous knowledge about the model's geometry or topology (Olsen et al. [49]). Constructive SBIM systems can be categorized by the two fundamental types of geometry being reconstructed: *linear* (i.e., lines, planes and polyhedra) or *free-form*. Linear SBM systems are typically oriented towards CAD and architecture applications. Two notable works that can be categorized as linear are Zeleznik et al. [75] and Jorge et al. [33]. Free-form SBM systems are used in applications requiring modeling of more organic, natural structures. We can cite the seminal work of Igarashi et al. [30] on the now classical *Teddy* system and Nealen et al. [48] (Figure 1.3). In Chapters 3 and 4 we develop tools for Free-form SBM systems.

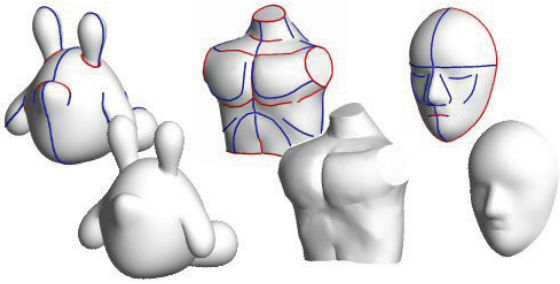


(a) Jorge et al. [33]




(b) Zeleznik et al. [75]


(c) Igarashi et al. [30]



(d) Nealen et al. [48]

**Figure 1.3:** Examples of constructive SBIM systems: linear (top row), Free-form (bottom row).

One important goal and challenge of constructive SBIM systems is to preserve the original modeling *intent* expressed by the user's 2D input sketch (i.e., "What You Sketch Is What You Get"). These systems aim to provide robust mechanisms to efficiently and effectively *interpret* the user's input strokes and to *map* them to quality representations of both the geometry and topology of the intended 3D object. Aiming to preserve the original modeling intent is particularly important during the initial stages of conceptual, free-form prototype modeling, where the main goal is to construct the overall shape of the object for later refinement and augmentation.

Different representation structures have been proposed for free-form SBIM constructive systems, including triangle meshes (Igarashi et al. [30], Karpenko and

Hughes [36], Cordier and Seo [14], Nealen et al. [48]) parametric surfaces (Cherlin et al. [13]) and implicit surfaces (Karpenko et al. [37], Araújo and Jorge [4], Alexe et al. [2], Schmidt et al. [57], Tai et al. [61], Bernhardt et al. [7]). In particular, prototype free-form SBIM systems can benefit from using implicit surfaces, which provide a compact, flexible, mathematically precise representation.

## 1.2 Contributions

First, in Chapter 2 the problem of how to deform an image using sketches is proposed as how to modeling a warping field. In this chapter we develop sketch tools to define warping fields using the Hermite–Birkhoff Radial Basis Functions (HBRBF), these functions are defined by two types of samples: points, and points and normals. In addition we present an application that uses these tools to warp RGBN images.

In Chapter 3 the objects to be modeled are implicit surfaces. We develop a set of modeling operators that are used to create samples which define a Hermite Radial Basis Functions (HRBF). HRBF representation use point and normals as samples. The operators are implemented in a sketch-based surface modeling application, this modeler successfully models Variational HRBF Implicit surfaces by interpolating the sketched curves, thus preserving the intended shape to be modeled.

While in Chapters 2 and 3 we develop sketch tools and applications guided by the representation, in Chapter 4 we invert this kind of approach. We conceive the representation to be tailored to our application. We introduce a novel mathematical representation for sketch-based modeling problem, this representation abstracts the principal qualities desired for many applications in which sketch is used as input for modeling. Besides that, we develop a SBM system based on the mathematical representation, this system uses a pipeline based on the theoretical framework developed.

The main contributions of this work are:

i) **A sketch-based warping field with region of interest control.**
A challenge in image warping using sketches is how to create a field defined with few strokes. In Chapter 2 we present an approach to create samples using sketches that defines a warping field which is $C^1$ and affect only where is defined by the user.

ii) **A sketch language using lines for implicit modeling.**
Prototype free-form SBM systems can benefit from using implicit surfaces, which provide a compact, flexible, mathematically precise representation. In Chapter 3 we present a collection of SBM operators inspired by traditional illustration techniques.

iii) **A surface representation tailored to sketch-based modeling.**
In Chapter 4 we propose a general mathematical representation for surfaces, conceived for specific sketch-based surface modeling processes.

iv) **A pipeline for Sketch-Based Modeling.**
To implement a modeler that uses the concepts of the surface representation developed in Chapter 4 we design a pipeline. This pipeline is tailored for SBM systems and is focused to control different levels of modeling edition.

v) **The creation of three applications of Sketch-Based Modeling.**
In Chapters 2, 3, and 4 we present concrete implementations of the presented ideas.

Parts of this work have been published:

- Pereira et al. [52]; *Sketch-based warping of RGBN images*, in Graphical Models (Chapter 2).
- Vital Brazil et al. [68]; *Sketching Variational Hermite-RBF Implicits*, in Proceedings of SBIM'10: 7th Eurographics Workshop on Sketch-Based Interfaces and Modeling (Chapter 3).
- Vital Brazil et al. [69]; *Shape and Tone Depiction for Implicit Surfaces*, in Computers & Graphics (Chapter 3).
- Vital Brazil et al. [67]; *A Few Good Samples: Shape & Tone Depiction for Hermite RBF Implicits*, in Proceedings of NPAR'10: 8th Intl. Symposium on Non-Photorealistic Animation and Rendering (Chapter 3).

# Chapter 2

# Sketching Warping Fields

In this chapter we propose a method to create samples for defining displacement fields in $\mathbb{R}^2$ using sketches; we apply this method to create fields for deforming images.

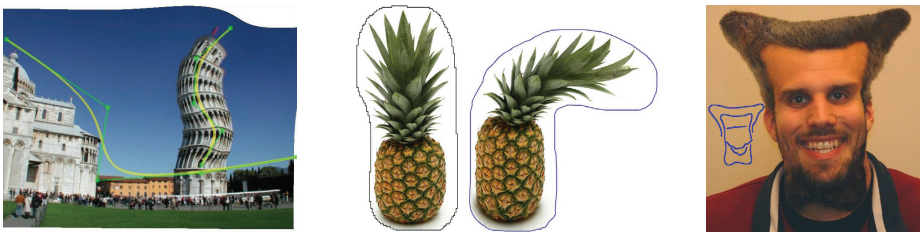
The problem of controlling image deformation has attracted the attention of researchers for a long time and there are many different solutions. Some approaches use control points or straight lines, notably the recent work by Igarashi et al. [31] and Schaefer et al. [56] (Figure 2.1). However, these elements are usually inconvenient for specifying curved restrictions, requiring too many points from the user to achieve a deformation that seems local. Sketching is a more natural way to communicate deformations in images, since artists typically want to define complex shapes intuitively and quickly rather than having to specify geometry precisely [75].

(a) Igarashi et al. [31] (b) Schaefer et al. [56]

**Figure 2.1:** Controlling image deformation using points and straight lines.

Weng et al. [71] use only source sketches and adjust a B-spline for control point manipulation. Our approach is most similar to methods that use sketches for defining both source and destination points, such as Fang and Hart [22] and Eitz et al. [21] (Figure 2.2). Igarashi et al. [31] calculate a deformation on a mesh and linearly interpolate the interior of the triangles. This results in a discontinuous Jacobian, constant inside each triangle. Fang and Hart [22] interpolate the field from the sketches using a Laplace equation, resulting in discontinuous derivatives on the sketches themselves. In most cases the effect desired is to deform a single object in the image or only a part of one object, as in Figure 2.1(b) the Pisa Tower. To obtain this local control the usual solution is increase the number of restrictions, particu-

larly in [56, 71, 22] the user creates extra curves, and in some cases the control is not achieved as we can note in the clouds in Figure 2.1(b) and the sky in Figure 2.2(a). Another desirable property is the field to be $C^1$ this is important for two reasons: to avoid creating seams and to be applied for warping normal as presented by Pereira et al. [52].



(a) Weng et al. [71] (b) Eitz et al. [21] (c) Fang and Hart [22]

**Figure 2.2:** Controlling image deformation using sketches.

The problem to be discussed in this chapter is how to use sketches to create a warping field with the following properties: (1) to be defined with few strokes; (2) to be $C^1$ on selected boundaries; (3) if a point is out of the region of interest it is not affected by the field. Part of this chapter was published in Pereira et al. [52], here we extended the description of how create samples using sketches (Section 2.2.3).

Our main purpose modeling these fields is to apply them to warping RGBN images controlling the region of interest (ROI). Our first attempt at creating displacement fields was to use a simple version of moving least squares (MLS) [56] to approximate the vectors samples. However, as discussed in Section 2.2.3 this approach does not guarantee the continuity in the boundaries and does not work well with sparse samples (Figure 2.11). For these two reasons, we developed a method to create samples inside the regions; this strategy reduces the problems but does not avoid them. Since to correctly warp normals of a RGBN images the field must be almost $C^1$ we changed our approach by adapting the methods to reconstruct surfaces and to model surfaces presented by Macêdo et al. [43, 44] and Vital Brazil et al. [68, 67]. Our new method creates warping fields using Hermit–Birkhoff Radial Basis Functions (HBRBF) [52] that interpolate the samples. The resulting fields are smooth inside and on the fixed boundary of ROI.

It is beyond the scope of this work to study deeply the methods to create the field (HBRBF or MLS); our main objective here is to discuss how to use sketches to create samples that define warping fields (Section 2.2). However, we need to discuss which properties the specific method requires. In Section 2.3 we discuss these requirements and we give a brief explanation of the HBRBF method, which was our final choice. In Section 2.4 we show and discuss some results of applying this method to define warping fields in RGBN images.

## 2.1 Sketch as Input

All sketch-based system starts by acquiring the points that will define the sketches and these points can come via a tablet, a mouse, trackpad (e.g., iPad), among others. In this work we do not care how these points are obtained, we just consider as raw data a sequence of 2D points. In almost all cases the raw input comes with noise and needs to be filtered. Our noise filtering process is done in two steps. First, we super-sample the input curve with a maximum distance of $\frac{1}{2}$ pixel between two consecutive points. Then we run 5 times the *reduce-resolution* algorithm 2.1 described in Samavati and Bartels [55].

**Input**: Sequence of points $F = F_1, \ldots, F_n$
**Output**: Sequence of points $C = C_1, \ldots, C_k$
$C_1 = F_1$;
$C_2 = -\frac{1}{2}F_1 + F_2 + \frac{3}{4}F_3 - \frac{1}{4}F_4$;
$j = 3$;
**for** $i = 3$ **to** $i \leq n - 5$ **step** 2 **do**
  $C_j = -\frac{1}{4}F_i + \frac{3}{4}F_{i+1} + \frac{3}{4}F_{i+2} - \frac{1}{4}F_{i+3}$;
  $j = j + 1$
**end**
$C_j = -\frac{1}{4}F_{n-3} + \frac{3}{4}F_{n-2} + F_{n-1} - \frac{1}{2}F_n$;
$C_{j+1} = F_n$;

**Algorithm 2.1**: Reduce-resolution (Samavati and Bartels [55]).

Where $n \geq 5$ and if $F$ is an open path then $k = \lfloor \frac{n}{2} \rfloor + 1$ else $k = \lfloor \frac{n}{2} \rfloor$. This filter provides both good overall approximation of the intended stroke path and distribution of points along the curve (Figure 2.3). From now on (all chapters), when we say user input or sketch we are considering the final filtered curve in spite of the raw data.

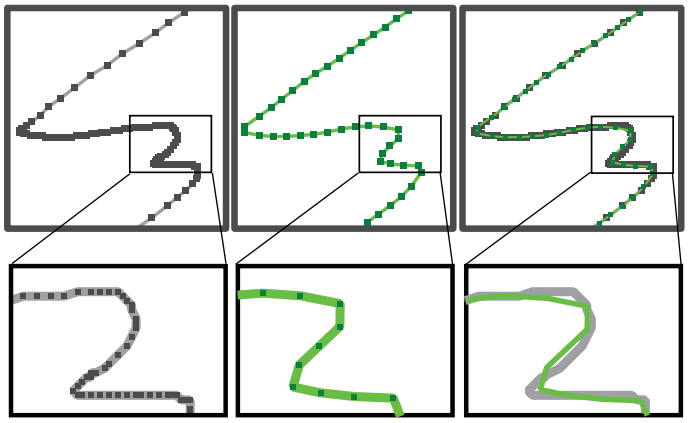


**Figure 2.3:** Input stroke: raw input (left), after filtering (middle) and overlapping for comparison (right).

## 2.2 Creating Samples

In our approach to create samples, the user defines two regions using sketches: the source region $\Omega$ and the destination region $\Lambda$, then we create vector samples that start in $\Lambda$ and end in $\Omega$. The region $\Omega \cup \Lambda$ is the region of interest (ROI), the warped object is generated by mapping $\Lambda$ to $\Omega$ and then calculating the properties (color or color and normal) in $\Omega$ using this map. We define this displacement field that maps $\Lambda$ on $\Omega$ using the vector samples.

To create samples we need the source region $\Omega$ and the destination region $\Lambda$. as mentioned before our samples map $\Lambda$ to $\Omega$; the names *source* and *destination* are chosen because the new properties of $\Lambda$ are calculated using the values in $\Omega$. Therefore, our first step is to create these regions (Section 2.2.1) using two sketching curves. However, if the user gives more information about the intended deformation, the applications can produce better results. We allow the user to sketch two internal curves to communicate better the intended deformation (Section 2.2.2). After all, we match points on curves in $\Omega$ with points on curves in $\Lambda$ to create the samples (Section 2.2.3).

### 2.2.1 Creating Regions

The user begins by drawing the boundary $\Omega^b$, thus defining the source region $\Omega$. Next, she or he sketches a curve $L$ specifying the deformation, i.e., this stroke is processed for defining the destination region $\Lambda$. If $L$ is closed, we get a well-defined destination region, i.e., the boundary of the region $\Lambda$ is $L$. However, most often, the curve $L$ is open, defining a new object silhouette. When $L$ is open, we need to build the $\Lambda$ region combining $\Omega^b$ and $L$. To achieve that $L$, must start and end on $\Omega^b$ (Figure 2.4(a)), inducing a partition of $\Omega^b$ in two curve segments $\Omega^b_1$ and $\Omega^b_2$ (Figure 2.4(b)). The border of region $\Lambda$ is defined by joining $L$ and one of $\Omega^b_i$ (Figure 2.4c).

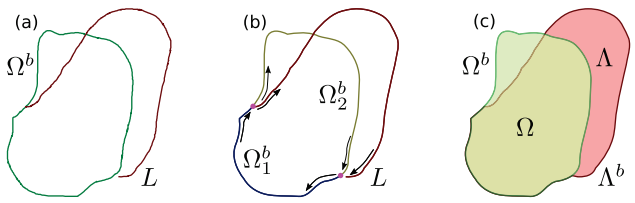



**Figure 2.4:** Processing of the input curves when $L$ is open: (a) user input; (b) We project the ends of $L$ on $\Omega^b$ and break it in two parts $\Omega^b_1$ and $\Omega^b_2$; (c) we connect $L$ with $\Omega^b_1$ resulting in the region $\Lambda$.

When $L$ is open a fixed boundary is created because $\Omega_b$ and $\Lambda_b$ are built using the same curve segment; specifically in Figure 2.4 the fixed boundary of $\Omega$ is $\Omega^b_1$. On this segment the field must be 0. When connecting $L$ to the correct $\Omega^b_i$, care must be taken to guarantee proper orientation. Indeed we always close the $\Omega^b$ and force its orientation to be clockwise; however the curve $L$ has free orientation, i.e., the user chooses its orientation. $L$ is connected to the $\Omega^b_i$ that results in an oriented

boundary $\Lambda^b$ of the target region $\Lambda$, particularly in Figure 2.4 $L$ is connected to $\Omega_1^b$. Observe that the user defines the region $\Lambda$ by the orientation of $L$ (Figure 2.5).

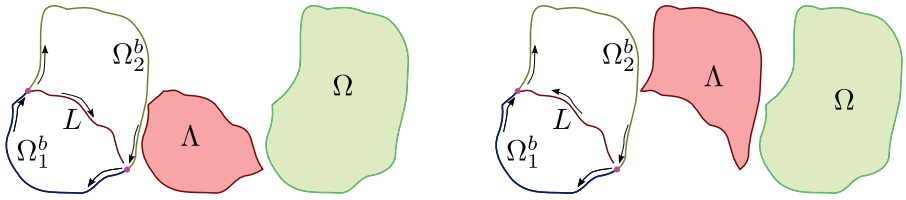



**Figure 2.5:** Defining two different $\Lambda$ regions using the same $L$ but with different orientations.

After obtaining the two regions we divide the boundaries curves into segments that are related to create the samples. If $L$ is closed we create four curves segments $\Omega_+^b$, $\Omega_-^b$, $\Lambda_+^b$, and $\Lambda_-^b$. If $L$ is open, one more curve segment is created, $F^b$. This fifth segment $F^b$ is the fixed boundary, as a result from now $\Omega^b = \Omega^b - F^b$ and $\Lambda^b = \Lambda^b - F^b$, if $F^b$ is empty these operations are still well defined. If no more information is given by the user we split $\Omega^b$ and $\Lambda^b$ in the half of the arc length creating $\Omega_+^b$, $\Omega_-^b$, $\Lambda_+^b$, and $\Lambda_-^b$ (Figure 2.6).

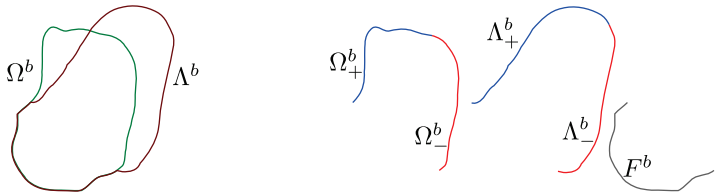



**Figure 2.6:** Dividing the boundaries curves in segments.

## 2.2.2 Increasing Control

To increase control on resulting field, the user can provide internal curves $\Omega^i$ and $\Lambda^i$ to constrain the deformation (Figure 2.7). If the internal curves touch the boundaries, they induce a partition of $B^b$ in two parts $B_+^b$ and $B_-^b$ where $B$ represents $\Omega$ or $\Lambda$. This partition allows the user to influence the sketch correspondence.

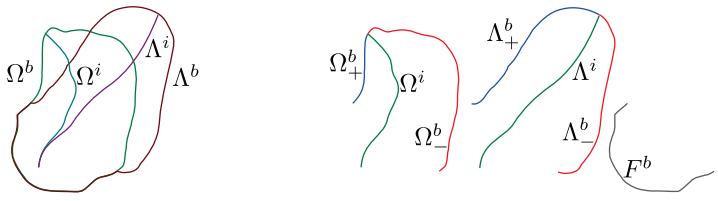
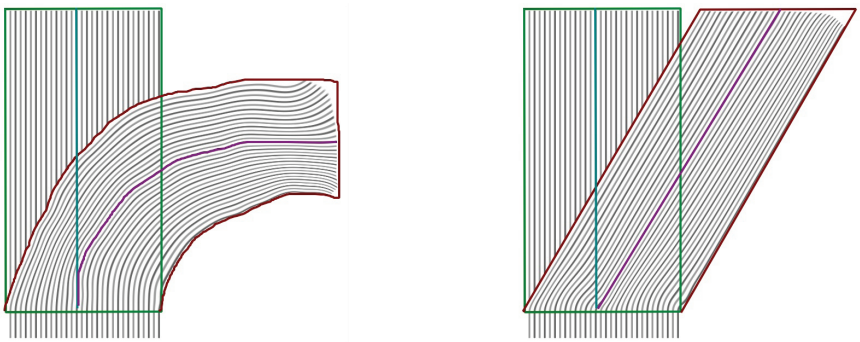



**Figure 2.7:** To increase control, the user can provide internal curves. If these curves touch the boundary they induce a partition of $\Omega^b$ and $\Lambda^b$ each one in two parts $\Omega_+^b$ and $\Omega_-^b$ and $\Lambda_+^b$, and $\Lambda_-^b$ respectively.

The internal curves allow the user to give additional information about the geometry of the warping. In Figure 2.8, we show the effect of using these curves and

partitioning the original sketches. When these curves are used, the internal samples guide the warping towards local rotations. Moreover, the induced partition improves sketch correspondence. Boundary and internal sketches provide good control. For example, the user can select between rotation-like and shear-like fields (Figure 2.9).

**Figure 2.8:** Comparison of two deformations: with and without internal curves. On the left, arc-length correspondence failed, resulting in extreme compression in the head; on the right, the partition induced by the internal curves led to a proper correspondence and warping.



(a) Rotation (b) Shear

**Figure 2.9:** We can sketch rotation-like or shear-like fields.

In Figure 2.10, we can observe the use of closed skecth $L$ and internal curves on a reservoir engineering illustration. In this case, content within a ROI is exaggerated by distortion as guided by the skeleton curves. This distortion creates emphasis for improving the visualization and it is inspired by traditional illustration techniques [39].

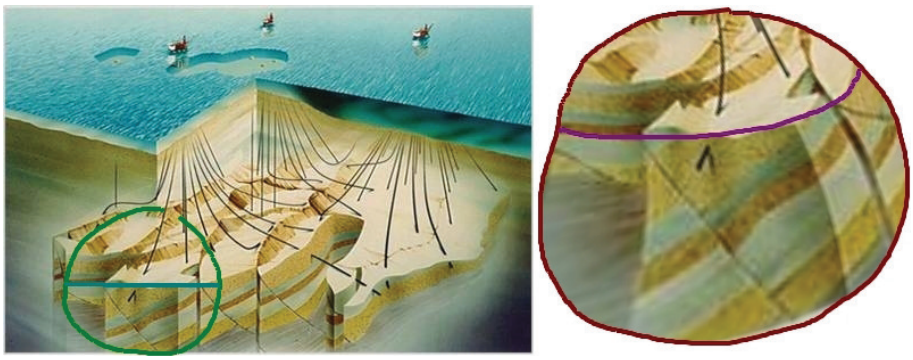


**Figure 2.10:** Curvature of reservoir layers is greatly exaggerated to bring emphasis in the visualization (Photo: Statoil).

### 2.2.3 Sampling

To create warping samples, the matching proceeds by pairing uniform samplings of corresponding curve segments, e.g., $\Omega^b_+$ with $\Lambda^b_+$. We found that this simple scheme to create samples by pairing uniform samplings provides a good compromise between the number of curves and field control; however, more automatic sketch-correspondence methods could be used such as those presented by Eitz et al. [21] or Zimmermann et al. [76].

The samples and the reconstruction methods define the field. In fact, when the final field does not have all properties desired, there are two possible approaches: improve the samples or substitute the reconstruction method. It is important to remark that we want fields that are $C^1$ and be valued 0 on the fixed boundary.

Like we said before, our first strategy to create the field was a simple method of MLS (Levin [41]) to approximate the samples. Our tests using MLS and samples over boundary curves and sketched internal curves, created fields that were almost constant or null. These results led us to develop a method to create more samples inside the regions. The method assumes the internal curves are defined; then these curves are evolved in "time" $t$ until they reach the boundary curve. Thus for each point in the internal curves there is a curve in $t$ which is it evolved until touching the boundary. We used these new curves to create the samples. If the internal curves are not sketched we create these curves connecting the projections points of $L$ on $\Omega^b$ (magenta points Figure 2.4(b)), and if $L$ is closed we connect the start and the middle points of each boundary curve. With this method of evolving the internal curves we create a large numbers of samples and improve the quality of the reconstructed field but the desired properties of continuity was not achieved (Figure 2.11).

Our second approach was to change the method to reconstruct the field; we chose the Hermit–Birkhoff Radial Basis Functions (HBRBF) [44] method which interpolates the samples (Figure 2.12). This method allows different restrictions over the samples. For our problem formulation the HBRBF method takes as input values or values and normals. Since HBRBF works well with sparse data we only use the sketched curves to create the samples. The points in $F^b$ take 0 as displacement vector; moreover we restrict their gradients to be 0 since we desire a $C^1$ field at these

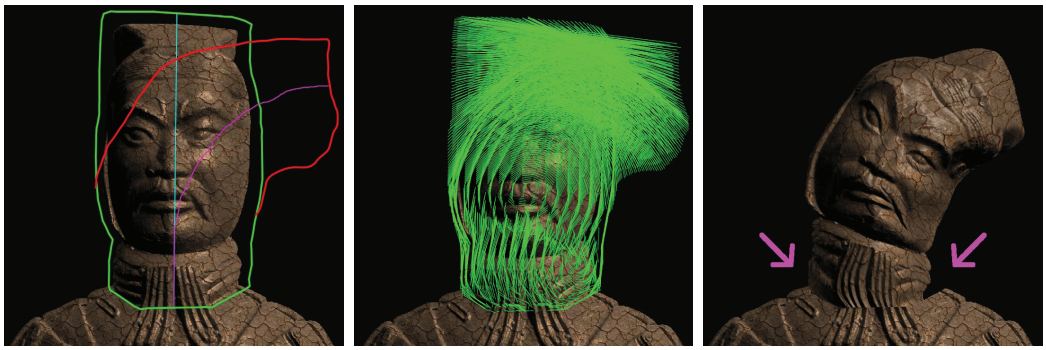


**Figure 2.11:** The MLS field applied on a RGBN image. Left to right: input curves, samples created, and the final image. Note (magenta arrows) the problem of discontinuity on the fixed boundary (Photo:Princeton Graphics Group RGBN dataset).

points. The other points of $\Omega$ take displacement vector $\Lambda^*(u) - \Omega^*(u)$, $u \in [0,1]$, e.g., $\Omega^b_+(1) - \Lambda^b_+(1)$ relates the point on the end of $\Lambda^b_+$ to the point on the end of $\Omega^b_+$. In the next section we discuss the HBRBF reconstruction method.

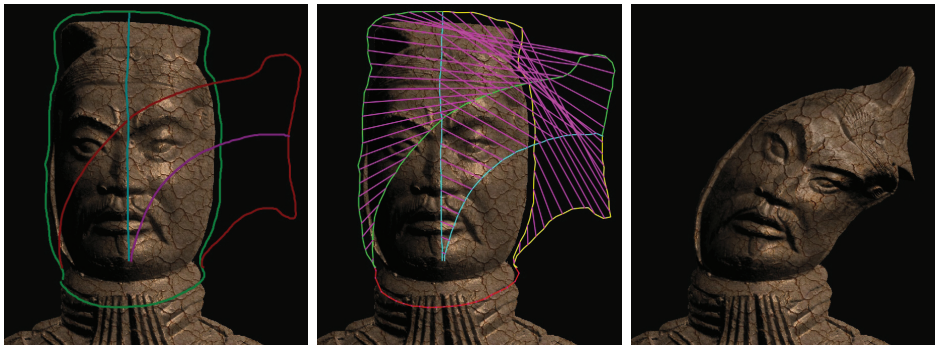


**Figure 2.12:** The HRBF field applied on a RGBN image. Left to right: input curves, samples created, and the final image (Photo:Princeton Graphics Group RGBN dataset).

## 2.3 Creating Fields

We build a displacement field by sparse interpolation of sketch samples. Our samples are vectors starting in $\Lambda$ and ending in $\Omega$ (Figure 2.4(c)). In particular, we start with samples that map boundary to boundary and internal curves to internal curves. Conceptually, we define a displacement field in the entire image, by restricting the interpolated field to $\Lambda$ and setting zero displacements outside of $\Lambda$. Any $C^1$ warping field in $\Lambda$ that interpolates these constraints will result in a continuous warp in the entire image, except on the new silhouette $L$. Notice that continuity on $F^b$ is a consequence of the constraints that fix it in place. However, a continuous field is not enough for applications that warp normals, because a non-smooth warp will result in discontinuous normals (Figure 2.13).

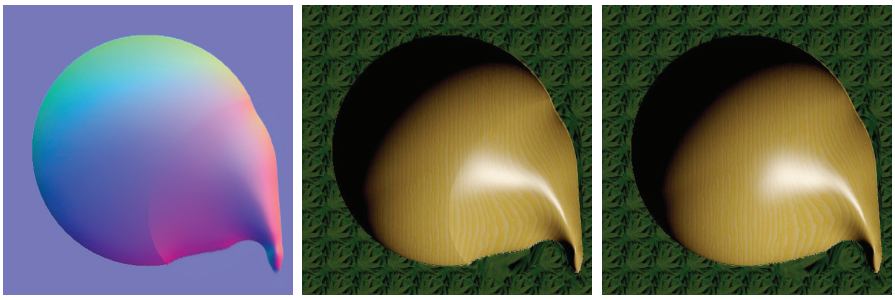


**Figure 2.13:** If the Jacobian is not restricted on the border, the field is not smooth, resulting in discontinuous normals (left) and shading (middle). On the right, a continuous derivative was used.

While many interpolation methods can be used to define $C^1$ fields inside $\Lambda$, we impose derivative constraints on the samples of $F^b$ that result in continuity of the derivative there. These more advanced constraints are satisfied using Hermite–Birkhoff RBF interpolation [70]. In theory, since derivatives are imposed only on the samples, our field may not be smooth between the samples of $\Omega_1^b$. Nevertheless, in practice, we have observed that the resulting field is sufficiently smooth for warping normals. The smoothness in the ROI is essential for color warping (Figure 2.14). The zebra's stripes break abruptly if derivative constraints are not used.

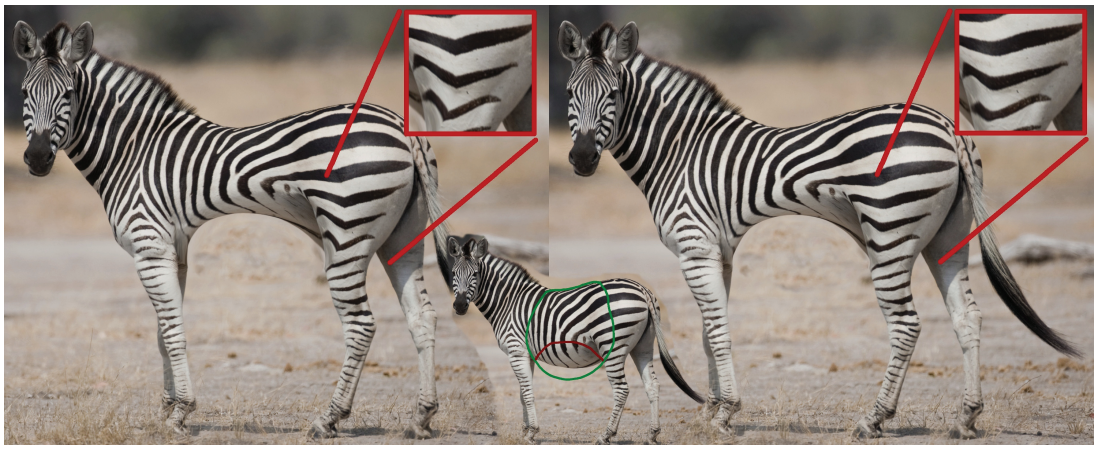


**Figure 2.14:** Smoothness in the border of the ROI is not useful only for normal warping. Without derivative constraints the stripes bend abruptly (left). We generate smooth color transitions (right).

### 2.3.1 Interpolant Field – HBRBF

In our system, we used a warping field computed by Hermite–Birkhoff interpolation based on radial basis functions [70]. Our constraints prescribe displacements at given points and, at some of them, they also enforce $C^1$-continuity of a restricted warping field.

Formally, we are looking for a displacement field $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\mathbf{F}(\boldsymbol{\lambda}^j) = \mathbf{c}^j$, $\mathbf{F}(\mathbf{x}^k) = \mathbf{0}$, and $\mathrm{D}\mathbf{F}(\mathbf{x}^k) = \mathbf{0}$, where $\boldsymbol{\lambda}^j \in \Lambda^b \cup \Lambda^k, \mathbf{x}^k \in F^b, \mathbf{c}^j = \boldsymbol{\omega}^j - \boldsymbol{\lambda}^j$ and $\boldsymbol{\omega}^j \in \Omega^b \cup \Omega^k$. The restrictions on the Jacobian $\mathrm{D}\mathbf{F}$ at the boundary points $\mathbf{x}^k$ enforce $C^1$-continuity of the interpolated field at the boundary of the region of interest, outside of which the field is supposed to be zero. Notice that the sets $\{\mathbf{x}^k\}$ and $\{\boldsymbol{\lambda}^j\}$ are disjoint. These constraints arise naturally in our context and lead to an instance of (multivariate and unstructured) Hermite–Birkhoff interpolation, which, unlike Hermite interpolation, does not require all derivative information at every sample.

Generalized interpolation problem can be computationally solved by means of radial basis functions techniques. First, notice that these constraints do not require coupling between the component fields $F_1, F_2 : \mathbb{R}^2 \to \mathbb{R}^2$ which define $\mathbf{F} = (F_1, F_2)$. This reduces the vector problem to two scalar problems defined by $F_l(\boldsymbol{\lambda}^j) = c_{j,l}$, $F_l(\mathbf{x}^k) = d_{k,l}$ and $\nabla F_l(\mathbf{x}^k) = \mathbf{0}$, $l = 1, 2$. Employing the generalized interpolation framework [23, 70, 43], we deduce appropriate forms of linearly-augmented RBF-based interpolants for both $F_1$ and $F_2$:

$$F_l(\mathbf{x}) = \sum_k \Big\{ \alpha_k \psi(\mathbf{x} - \mathbf{x}^k) - \langle \boldsymbol{\beta}^k, \nabla\psi(\mathbf{x} - \mathbf{x}^k) \rangle \Big\} + \sum_j \gamma_j \psi(\mathbf{x} - \boldsymbol{\lambda}^j) + \langle \mathbf{a}, \mathbf{x} \rangle + b$$

where $\psi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$ for a suitable radial basis function $\phi : \mathbb{R}_+ \to \mathbb{R}$ and $\alpha_k, \gamma_j, b \in \mathbb{R}$ and $\boldsymbol{\beta}^k, \mathbf{a} \in \mathbb{R}^2$ are the fitting coefficients uniquely determined by the aforementioned interpolation constraints along with the additional side-conditions:

$$\sum_k \Big\{ \alpha_k \mathbf{x}^k + \boldsymbol{\beta}^k \Big\} + \sum_j \gamma_j \boldsymbol{\lambda}^j = \mathbf{0}$$
$$\sum_k \alpha_k + \sum_j \gamma_j = 0.$$

Thus, after choosing a suitable $\phi$, the fitting coefficients can be computed by solving two symmetric indefinite linear systems which only differ on their right-hand sides. Our implementation employs an $LDL^T$ factorization of the system matrix which is used to fit both $F_l$. We found that `LAPACK`'s `xSYSV` $LDL^T$ routines provide a better balance in performance/memory/stability requirements than the alternative `xGESV` and `xSPSV` (also from `LAPACK`), which implement a general $LU$ decomposition and a packed $LDL^T$-factorization respectively.

There are several alternatives for choosing a suitable basis function $\phi$. As we are interested in warping fields that are at least $C^1$ to correctly propagate normals and to approximate the smoothness condition on the boundary of the region of interest, we seek a $\phi$ inducing a $\psi$ at least $C^2$, since the interpolant contains first-order differentials of $\psi$. We employed the globally-supported basis function $\phi(r) = r^3$, which has interesting variational properties, as studied by Duchon in his seminal paper [20]. Consequently, our method has no additional interpolation parameters.

It is noteworthy that, due to the use of Duchon's basis function, we need the polynomial part in the interpolant to ensure solvability of the resulting linear system. Although we could drop this affine term if we used Wendland's functions [70], we found that Duchon's provides better warping fields without noticeable degradation in performance due to the additional polynomial. In addition, although Wendland's functions are compactly-supported, the resulting system would still be dense because the associated radius parameter would need to be large due to the sparsity of our samples.

Our vector field interpolation reconstructed a displacement field $\mathbf{F}$. From it, we obtain a warping $\mathbf{W} : \Lambda \to \Omega$, $\mathbf{W}(\mathbf{x}) = \mathbf{x} + \mathbf{F}(\mathbf{x})$, which is used for deformation in the next section.

## 2.4 One Application: Warping RGBN Images

In Pereira et al. [52] we propose a sketch-based pipeline to deform images having more than pixel color information. We apply the HRBF displacement map to this problem because, as discussed in previous sections, it reconstructs a smooth field that interpolates sparse samples. To test the sketches with ROI control, smoothness of field, the concept of local free-form in the context of warping RGBN images, we developed a system with all these pieces together. In the next paragraph we give an overview of system (Figure 2.15).

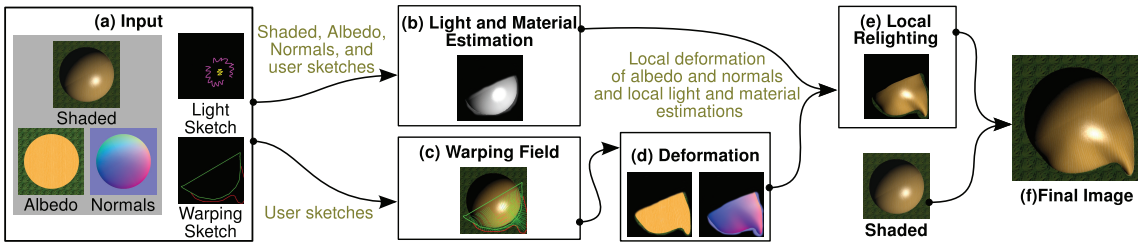
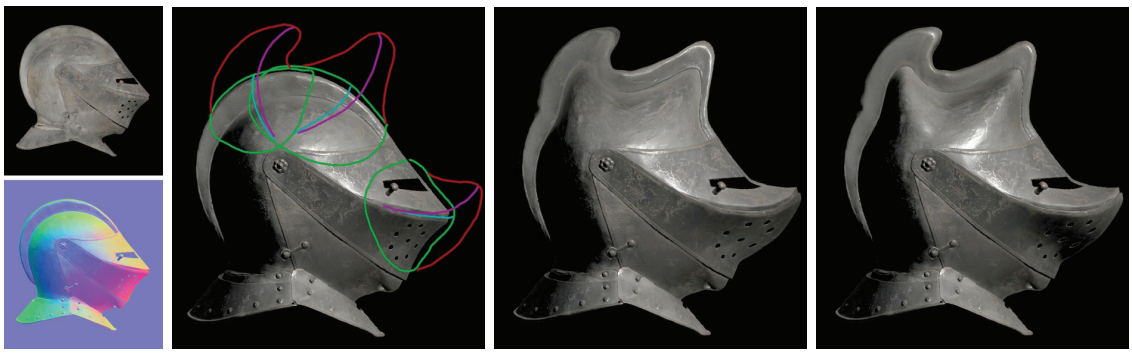



**Figure 2.15:** RGBN - Sketching Warping Fields: Overview.

In our system, we take as input three different buffers (Figure 2.15(a)): the shaded buffer, which is an input image containing colors that are influenced by scene lighting conditions; and diffuse albedo and normal channels, which specify the object's texture and geometry. Afterwards, assuming a constant material, we use light sketches and all three buffers to obtain a local estimate of specular properties, a single directional light and an ambient illumination term (Figure 2.15(b)), which are used to relight the final result. Additionally, warping sketches specify a region of interest (ROI) and a deformation. These are used to build the warping field (Figure 2.15(c)), which is used to locally deform the albedo and normal buffers (Figure 2.15(d)). To obtain the final image we relight the warped albedo and normals in the ROI using the estimated light (Figure 2.15(e)) and compose it with the rest of the original shaded image (Figure 2.15(f)). The warping field is built using Hermite–Birkhoff radial basis functions (HBRBF), resulting in a smooth field necessary for

normal manipulation.

Our contribution in that warping application is a sketch-based method to deform images that does more than simply transfer colors. Our method generates results that appear to be photometrically consistent (Figure 2.16). By relighting the warped image, we can generate new highlights, shading and shadows where necessary. To support relighting, we have developed a method to seamlessly deform RGBN images. We have shown examples of our method working both locally in single photographs and globally with photometric-stereo RGBN images. The sketch language implemented despite being simple is very powerful and the user achieve complex deformations with good control using few curves.



(a) RGBN image (b) Shaded input and sketches (c) Traditional image warping (d) RGBN image warping

**Figure 2.16:** Our system takes as input albedo and normals (a), and a shaded image (b). In addition, the user makes two kinds of sketches (b): warping sketches specify a region of interest and its deformed shape; lighting sketches help the system estimate lighting and material. We relight the warped result (c), obtaining objects illuminated coherently (d). Simple color warping stretches the original highlight and misses some shadows. With relighting, we recreate fine highlights and shadows in the right places. Three sequential deformations were used (Photo: ICT's Graphics Lab).

## 2.5 Conclusion and Future Works

In this chapter we developed sketch tools to create samples that define a warping field. These tools allows control the field created restricting it to be valued 0 outside the ROI and be at least $C^1$ where the user defines. We successfully apply the obtained field to deform simple images and RGBN images. One problem is that the field interpolation result is very dependent on the field samples. If the arc-length correspondence fails to capture user intent, it will result in a distorted warping field. This problem could be avoided by the user if he creates the final deformation using more than one step, as we show in Figure 2.16.

In future work, we intend to use our method in other applications, including warping of vector fields other than gradients. Moreover, by further exploiting the Hermite-Birkhoff interpolation theory with RBFs, more elaborate derivative restrictions can be used for much more than boundary restrictions. For instance, they could be used to specify local rotations and stretches, thus coupling the scalar fields which define the warping, especially if integrated in a friendly user interface. Finally, we would like to extend our method to image-based animation systems having as input either photographs [28] or drawings from cel animation pipelines as in Lumo [32] which is a good challenge to the sketch language as the samples in this problem has one more component: time.

# Chapter 3

# Sketching Implicit Surfaces

In this chapter, we introduce a new representation for implicit surfaces and show how it can be used to support a collection of free-form modeling operations (Figure 3.1). Our main contributions here is: (1) an energy-minimizing Hermite interpolation scheme for sketch-based modeling (SBM) of free-form surfaces, and (2) a collection of SBM operators inspired by traditional illustration techniques to demonstrate the usefulness of the proposed representation.

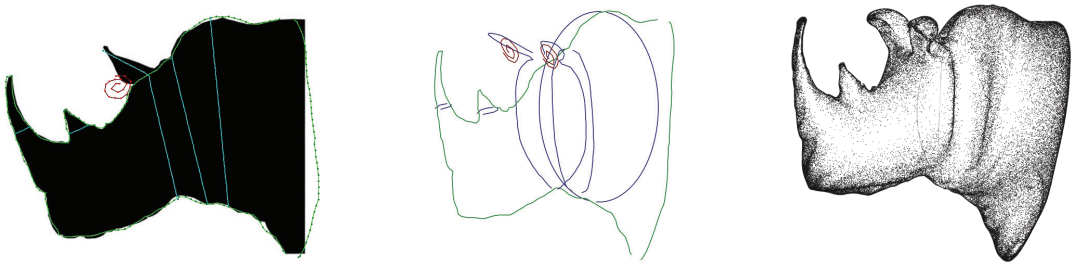


**Figure 3.1:** Modeling a rhinoceros head with our system: the user interactively sketches construction curves over a silhouette image (left), these curves are transported to 3D space (center), resulting in implicit model, shown in a stippled rendering (left).

This chapter is a version of the paper Vital Brazil et al. [68]. We extend the description of the Hermite interpolation scheme for sketch-based modeling (Section 3.2) as well as the rendering approach (Section 3.5).

## 3.1 Related Work

Previous and related works concerning the SBM method proposed in this chapter encompass the following areas: interactive modeling and visualization of implicit surfaces; sketch based interfaces; and surface reconstruction techniques from points and normals.

Early approaches for providing interactive implicit modeling and visualization involved creating a separate polygonization for each model primitive (Desbrun et al. [18]). Further work explored level-of-detail (LOD) implicit surface representations

(Bloomenthal and Wyvill [11], Barbier et al. [5]). As well as the use of particle systems for both surface editing and rendering, some of the main references along this line of research include: Witkin and Heckbert [72], and Hart et al. [27]. Other modeling techniques resort to a volumetric representation of implicit surfaces, such as the ones based on discrete volume datasets: Owada et al. [50]; adaptive distance fields: Ferley et al. [24], Perry and Frisken [53]; and more recently using level set methods: Museth et al. [46]. A different approach is based on interactively functional composed models: Barthe et al. [6]. Finally, we should mention the pioneering work in variational implicit surface modeling proposed by Turk and O'Brien [62].

Regarding sketch-based interfaces, implicit-based SBIM systems have contributed lately to provide effective and efficient modeling and visualization for implicits. Major efforts in this direction target adapting variational implicit surface models such as the works of Karpenko et al. [37], Araújo and Jorge [4], and Alexe et al. [2]. Sketch-based interfaces have also been applied to distance fields combined with mesh subdivision techniques (Markosian et al. [45], Igarashi and Hughes [29]); as well as to hierarchical implicit models (Schmidt et al. [57]); and to convolution implicit surfaces (Tai et al. [61], Bernhardt et al. [7]).

In terms of surface reconstruction methods, most related to our work are techniques presented in [37, 4], derived from classical interpolation theory by Duchon [20] and subsequently introduced in graphics by Turk and O'Brien [62]. In these works, the creation of artificial offset points is required to properly fit an implicit surface from a given set of points and normals. A recent approach proposed by Macêdo et al. [43, 44] reconstructs implicits from points and normals based on a generalized interpolation framework, and exhibits increased robustness for coarse and non-uniform samplings, as well as to close-sheets without the need of artificial offsets. The key ingredient to this method is the treatment of normals as hard constraints on the gradient field of the fitted implicit function. Macêdo et al. [43, 44] introduce a representation suitable for SBM of implicit surfaces built upon the work of Duchon [20]. This representation combines the advantages of Hermite reconstruction of surfaces with the variational characteristic of those representations based on Duchon's work, most notably, the ability to extrapolate the surfaces well on empty regions, thus filling unsampled areas.

## 3.2 Variational HRBF Implicits

Recently introduced by Macêdo et al. [43, 44], Hermite Radial Base Function (HRBF) Implicits provide a powerful tool to reconstruct implicitly-defined surfaces from points and normals. These papers present a theoretical framework for generalized interpolation with radial basis functions. They specialize their results to first-order Hermite interpolation that are employed to recover implicit surfaces.

We decided to investigate the use of their technique in SBM systems for implicits surfaces. We were motivated by the quality reconstructions they obtain from coarse and nonuniform samplings. This is especially important when compared to the offset-based methods often employed in sketch-based implicit modeling systems

[12, 36]. Although we obtained good results in our experiments, the sparsity and coarseness of the stroke curves rendered pointless the use of compactly-supported functions and their associated radius parameter. The main advantage of using functions with compact support lies in exploiting sparsity of the corresponding interpolation linear system when dealing with large datasets. This structure could not be exploited since the radius had to be taken large enough to compensate for the large areas not covered by the sparse set of strokes.

These shortcomings motivated us to look for alternatives that better exploited the structure of our application: sparsely-drawn strokes leading to small-to medium-sized datasets with large void areas. We found an answer to these issues in Duchon's seminal paper [20], which contains derivations analogous to those in [44] for a particular class of function spaces. It is noteworthy that, even though the classical-interpolation part of Duchon's work has influenced the offset-based methods used for variational implicit modeling today, we focus on his Hermite interpolation result and introduce Variational HRBF Implicits.

In the following, we provide a brief description of the Variational HRBF Implicits interpolant and its key properties which motivated its use for SBM of implicit surfaces.

### 3.2.1 Description

Macêdo et al. [43, 44] posed the problem of reconstructing an implicit surface from points $\{\mathbf{x}^j\}_{j=1}^N \subset \mathbb{R}^3$ and normals $\{\mathbf{n}^j\}_{j=1}^N \subset \mathbb{S}^2$ as a Hermite interpolation problem, in which a function $f : \mathbb{R}^3 \to \mathbb{R}$ was sought such that $f(\mathbf{x}^j) = 0$ and $\nabla f(\mathbf{x}^j) = \mathbf{n}^j$, for each $j = 1, \ldots, N$. The authors presented a framework that allowed them to derive a concrete form for such an interpolant, depending on the specification of a radial basis function $\psi : \mathbb{R}^3 \to \mathbb{R}$, which they named *HRBF Implicit*.

The variational Hermite interpolant deduced by Duchon in [20] has essentially the same form of a HRBF Implicit with $\psi(\mathbf{x}) = \|\mathbf{x}\|^3$, differing only by an additional low-degree polynomial term which comes accompanied by additional conditions to ensure well-posedness of the interpolation problem. It should be noted that this RBF does not satisfy the sufficient conditions required by the framework presented in [44]. However, Duchon's results ascertain the solvability of the interpolation system. For this reason, we named this representation a *Variational HRBF Implicit*.

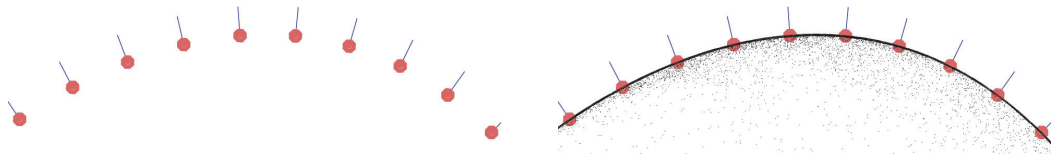
The concrete form of the Variational HRBF Implicit that we shall use is

$$f(\mathbf{x}) = \sum_{j=1}^{N} \left\{ \alpha_j \psi(\mathbf{x} - \mathbf{x}^j) - \langle \boldsymbol{\beta}^j, \nabla\psi(\mathbf{x} - \mathbf{x}^j) \rangle \right\} + \langle \mathbf{a}, \mathbf{x} \rangle + b. \tag{3.1}$$

with coefficients $\alpha_1, \ldots, \alpha_N, b \in \mathbb{R}$ and $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^N, \mathbf{a} \in \mathbb{R}^3$. These coefficients can be uniquely determined by enforcing the Hermite interpolation conditions above ($f(\mathbf{x}^j) = 0$ and $\nabla f(\mathbf{x}^j) = \mathbf{n}^j$) along with

$$\sum_{j=1}^{N} \alpha_j = 0, \qquad \sum_{j=1}^{N} \left\{ \alpha_j \mathbf{x}^j + \boldsymbol{\beta}^j \right\} = 0$$

as long as the sample points $\mathbf{x}^j$ are pairwise-distinct (a mild and natural assumption). This implies that we can find an implicitly-defined surface passing through given points with prescribed normals at all points (Figure 3.2) just by solving a symmetric indefinite system of linear equations for the coefficients of Variational HRBF Implicit's. We provide below all the formulas needed to assemble the interpolation system and to evaluate the implicit function as well as its gradient.



(a) Given data: points and normals. (b) Implicit surface reconstructed.

**Figure 3.2:** Interpolating points and Normals.

In order to assemble the interpolation system, we employ a direct implementation of the block matrix defined by a samplewise grouping of the conditions $f(\mathbf{x}^i) = 0$ and $\nabla f(\mathbf{x}^i) = \mathbf{n}^i$, where $f$ is given in (3.1), and by the side-conditions for degree-one polynomials. This results in the following set of equations

$$\begin{bmatrix} 0 \\ \mathbf{n}^i \end{bmatrix} = \sum_{j=1}^{N} \begin{bmatrix} \psi(\mathbf{x}^i - \mathbf{x}^j) & -\nabla\psi(\mathbf{x}^i - \mathbf{x}^j)^T \\ \nabla\psi(\mathbf{x}^i - \mathbf{x}^j) & -\mathrm{H}\psi(\mathbf{x}^i - \mathbf{x}^j) \end{bmatrix} \begin{bmatrix} \alpha_j \\ \boldsymbol{\beta}^j \end{bmatrix} + \begin{bmatrix} 1 & (\mathbf{x}^i)^T \\ \mathbf{0} & \mathrm{I}_{3\times 3} \end{bmatrix} \begin{bmatrix} b \\ \mathbf{a} \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix} = \sum_{j=1}^{N} \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{x}^j & \mathrm{I}_{3\times 3} \end{bmatrix} \begin{bmatrix} \alpha_j \\ \boldsymbol{\beta}^j \end{bmatrix}$$

where the unknowns $\{\alpha_j, \boldsymbol{\beta}^j\}_{j=1}^N$, $\mathbf{a}$, and $b$ are computed after an $LDL^T$-factorization of the resulting symmetric (indefinite) matrix and subsequent forward- and backward-substitutions, as implemented in the `DSYSV` routine from the `LAPACK` library [1]. The concrete formulas for the functions used in the subblocks above are

$$\psi(\mathbf{x}) = \|\mathbf{x}\|^3, \quad \nabla\psi(\mathbf{x}) = 3\mathbf{x}\|\mathbf{x}\|, \quad \mathrm{H}\psi(\mathbf{x}) = \frac{3}{\|\mathbf{x}\|}\left(\|\mathbf{x}\|^2\,\mathrm{I}_{3\times 3} + \mathbf{x}\mathbf{x}^T\right)$$

Next, we discuss some computational aspects of the numerical solution of our interpolation systems. In the following, we discuss the salient properties of this representation which are relevant for sketch-based implicit modeling.

### 3.2.2 Variational HRBF for Sketch-Based Modeling

The Variational HRBF Implicits representation has a number of interesting properties for the application in sketch-based modeling, either verifiable through theoretical arguments or indicated by experimentation, most notably:

- **Energy-minimizing Hermite interpolation.** Duchon's deductions, which resulted in the variational Hermite interpolant, sought a function minimizing

an energy measure on its derivatives under the pointwise Hermite interpolation constraints. This energy-minimizing property on the function's derivatives induces an interpolant that penalizes spurious oscillations, thus producing well-behaved surfaces obeying both point and normal constraints from the strokes.

- **Invariance under geometric similarities.** The implicit surface recovered as a Variational HRBF Implicit is invariant to geometric similarity transformations under the input samples. Intuitively this means that the shape of the reconstructed surface is the same, no matter how we consistently move, rotate, reflect, or uniformly scale the input samples.

- **Parameter-free.** The Variational HRBF Implicit interpolant does not depend on any kind of support-radius or normal-offset parameters usually required by current RBF-based representations (Carr et al. [12], Karpenko et al. [37]).

- **Smoothness.** It is guaranteed that the implicit function computed is globally $C^1$, is $C^\infty$ in the whole domain but the sample positions, and has bounded Hessian near the interpolated points. This property allows suitable estimation of differential quantities such as normals, curvatures, and principal directions at regular points.

- **Robustness under sparse samplings.** Since the recovered implicit function's gradient interpolates the unit normals provided along with the input points, Variational HRBF Implicit behaves robustly in their vicinity even under nonuniform and coarse samplings, while extrapolating the reconstructed surface well over large unsampled regions. This property is one of the main reasons Variational HRBF Implicits provide a powerful representation for SBM of implicit surfaces.

- **Well-behavior near close-sheets.** By directly interpolating normals with the implicit function's gradient, the HRBF surface reconstruction can deal with close-sheets, which usually pose a difficulty for offset-based schemes [12, 37].

- **Linear fitting.** To fit a Variational HRBF Implicit from given points and normals, all that is required is to solve a symmetric indefinite linear system on $4N + 4$ unknowns. As we discussed before, this can be accomplished by employing off-the-shelf linear algebra packages.

- **General data.** To ensure well-posedness of the Hermite interpolation process and the invertibility of the associated linear system, the only and mild requirement on the dataset is that the input points need to be pairwise distinct.

- **Linear reproduction.** If the data is sampled from an affine function (defining a hyperplane), the recovery is exact. Intuitively, the interpolant approximates the data as well as it can by an affine function, and then corrects the errors with the HRBF-expansion.

After describing our basic surface representation, we now proceed with its actual use in our sketch-based implicit surface modeling system.

## 3.3 Modeling Pipeline

Our pipeline is divided in two main steps. First, as preprocessing step, we handle the user input strokes and process them to create samples (points and normals) in $\mathbb{R}^3$. Then, we use a Hermite RBF fitter to obtain our final implicit surface (Figure 3.3). In the following subsections, we discuss these issues, as well as the sketch preprocessing and the fitter used in our system.

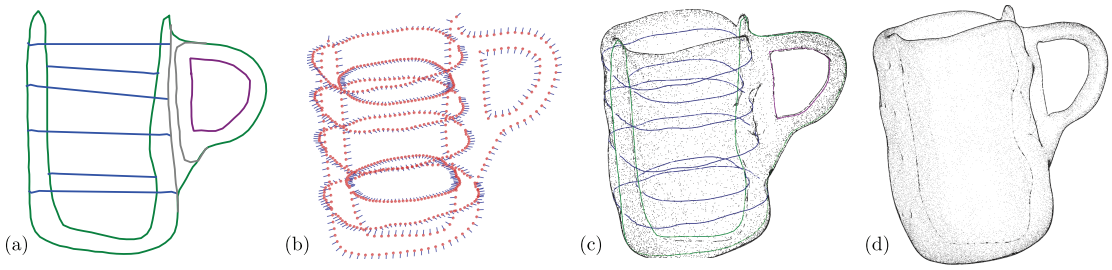



**Figure 3.3:** Pipeline: (a) user input sketches, (b) samples (points and normals), (c) preview rendering during a prototype modeling session, computed after preprocessing 4K primitives in about 5 seconds, (d) final rendering with 100K primitives processed in about 5 minutes. Green curve is the external contour, magenta is the internal contour, the blue are the cross editing curves and the grey are curves not used after the oversketching operations.

### 3.3.1 Sketch Preprocessing

The process starts from a blank canvas, where the user sketches the curves for defining the final model. These curves are labeled as one of our sketch-based operators (Section 3.4). Next, all curves are processed on the plane (to filter noise, Section 2.1), and transported to model space. Normals are then created for each curve point while observing the specific rules of the associated operator. We assume all curves are parametrized in the interval $[0, 1]$.

The next step is to create the 3D samples (points and normals) which will be interpolated by our implicit surface. A set of SBM operators was created for testing our implicit representation. These operators are grouped into two classes for creating *contour* and *inflation* curves. Contour curves have the semantics of an object silhouette; inflation curves control how the surface is outside the drawing plane. In contrast to inflation curves, which are immersed in $\mathbb{R}^3$ but not necessarily lie in a plane, we are assuming that contours are planar curves. All strokes are sketched on one same plane; in particular we use the plane $\Pi_d : (z = 0)$.

Since the contour curves are directly transported by an affine transformation to the final position in space, we can consider the contour points as in $\Pi_d$. We also use this plane as a reference frame for the inflation curves. We treat two different

types of contour curves: *incomplete* and *complete* contours (Section 3.4.1). To assign normals to contour samples, we compute the cross-product between the normal of the drawing plane $\mathbf{N}_d$ and the unit tangent vector $\mathbf{t}$ at the given sample point on the curve, and assign to that point the normal $\mathbf{n} = \mathbf{N}_d \times \mathbf{t}$ (Figure 3.4).

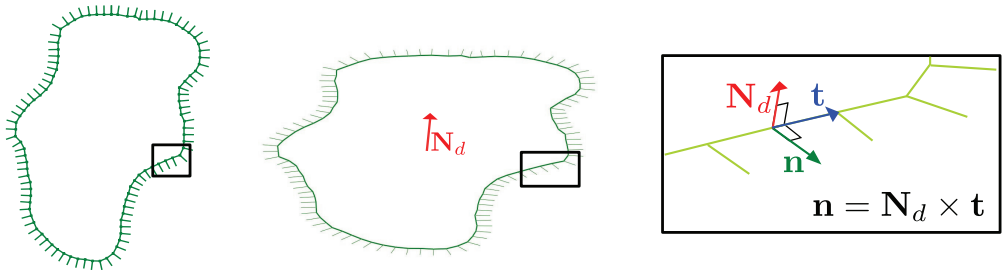



**Figure 3.4:** Assigning normals to contour samples. Left to right the input curve and its normals, the curve and normals transported to space, depiction of the process of assigning normals.

Inflation curves are used to create samples outside the drawing plane. We have two operators to create them: *kneading* and *cross-editing* (Section 3.4.2). A kneading curve $K$ is transported to a user-defined plane, whose normal is assigned to each sample (Figure 3.5). We define this plane using a point $p_0$ and a normalized vector $\mathbf{n}$ defined by the user. To determine $p_0$ we use the barycentric $p_b$ of the input curve $K$ plus a height $h$ in the direction of $\mathbf{N}_d$, i.e., $p_0 = p_b + h\mathbf{N}_d$.

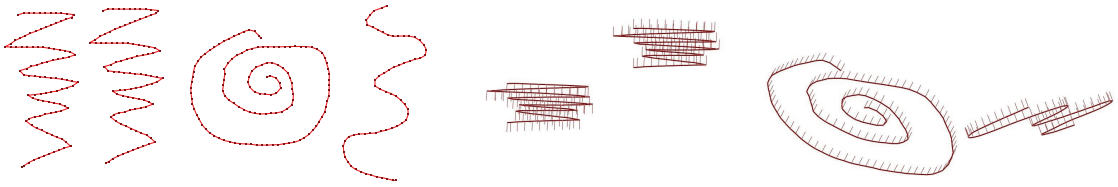


**Figure 3.5:** Kneading curves: Left curves in 2D, right kneading curves and its normals in model space using different user-defined planes.

For each cross-editing curve $C(t)$ drawn, we associate two plane curves $\gamma_1$, $\gamma_2 : [0,1] \to [0,1] \times [0,1]$, i.e., for each cross-path $C(t)$ drawn in $\Pi_d$ we create two profiles $\gamma_1(t)$ and $\gamma_2(t)$ defined in another plane. With $x_t^i$ and $y_t^i$ being the coordinates of $\gamma_i$ at $t$, we define two 3D curves $I_i(t) = C(x_t^i) + y_t^i \cdot \mathbf{N}_d$ for each $i = 1, 2$. The normal assigned to $I_i(t)$ is the normal of $\gamma_i(t)$ transported to the plane $\Pi_t$ defined by the tangent of $C(x_t)$ and $\mathbf{N}_d$. The curves $\gamma_i$ can be defined either automatically, as fixed functions of the arc-length of the curve $C$, as example a semi-circle, or by input sketches. In Figure 3.6 we show the process to create samples using the cross-editing curve $C(t)$ and sketched $\gamma_1$, $\gamma_2$.

## 3.4 Sketch-Based Modeling Operators

In this section, we present a set of SBM operators to create and test a variety of implicit objects modeled using our Variational HRBF scheme. The design of these

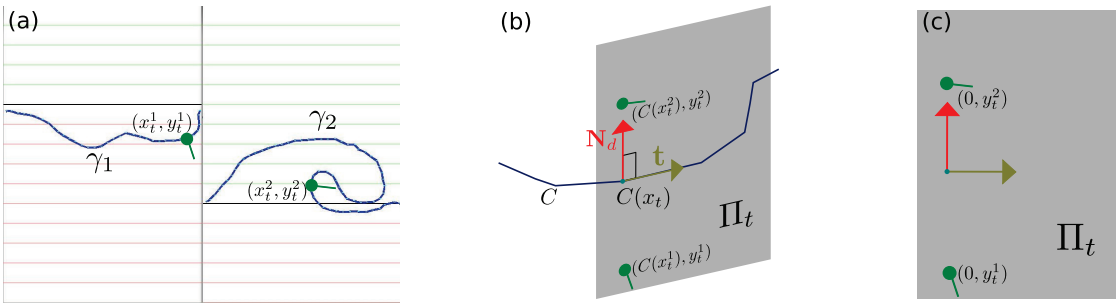



**Figure 3.6:** Creating samples using the cross-editing curve $C$. (a) The associate plane curves $\gamma_1$ and $\gamma_2$ drawn and points $(x_t^1, y_t^1)$ and $(x_t^2, y_t^2)$ and its normals. (b) The plane $\Pi_t$ created using $\mathbf{N}_d$ and $\mathbf{t}$; and points and normals in $I_1(t)$ and $I_2(t)$. (c) The local coordinates in plane $\Pi_t$ of points and normals in $I_1(t)$ and $I_2(t)$.

SBM operators were inspired in traditional techniques and tools used by artists and illustrators (Andrews [3]).

### 3.4.1 Contouring

To outline the external and internal contours of a model, the user has two options: *incomplete* and *complete* contours (Figure 3.7). Incomplete contours are used to suggest the contour of the model using a few number of open curves [3]. Complete contours are used to define the entire contour of the model using closed curves. In both operators, the drawing direction in the drawing plane defines the sample normals. This allows the user to define either exterior or interior contours (i.e., holes) by drawing clockwise or counterclockwise, respectively.

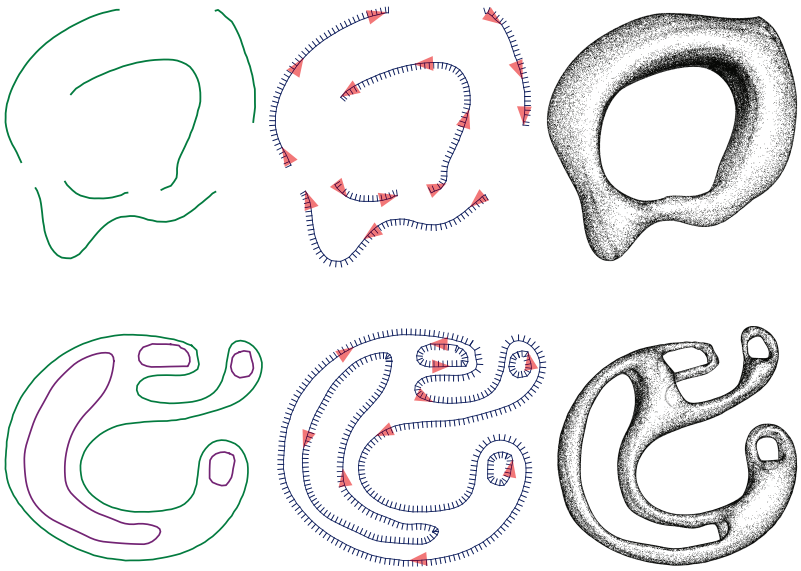


**Figure 3.7:** Contouring operator: open and closed curves defined by *incomplete* (top row) and *complete* (bottom row) contours. Left to right: input strokes, normals and final stippled model. Red triangles depict curve orientation.

### 3.4.2 Inflation

There are many ambiguities in how to inflate a model with only coplanar samples [36]. After the user delimits the overall shape contour, its inflation can be controlled by using two classes of curves: *kneading* and *cross-editing*.

Kneading curves allows to freely mold regions across the model. This operator is inspired by traditional kneaded erasers made of pliable material that can be shaped by hand. By using this operator the user has additional control on the inflation of the model (Figure 3.8).

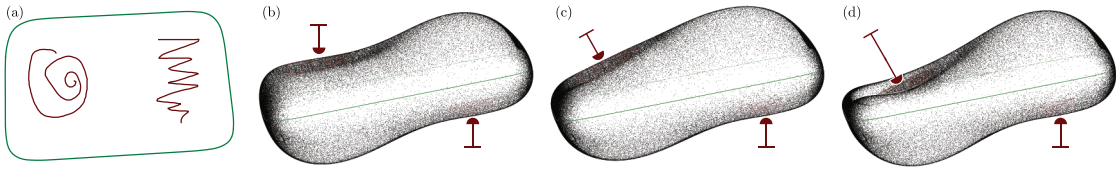



**Figure 3.8:** Using kneading to control the inflation: (a) Input strokes of the external contour of the model (green) and the free-form curves (red) defining the regions to be kneaded; (a), (b), and (c) the resulting 3D model with different user-defined planes, red arrows indicating the kneading orientation.

Cross-editing curves define the profiles of the final model. These curves are built in two steps: first the user draws a path over $\Pi_d$ and then defines the profiles of the curves. Each path is associated with two curves defining the top and bottom regions in relation to the drawing plane. These paths represent the projection of the final 3D curve on the drawing plane. Our system provides the users two options to define these profiles. The first option automatically creates two semicircles; the second one allows the user to define a profile by sketching it on an auxiliary canvas. Cross-editing curves can be used to define either linear (Figure 3.9(a)) or free-form cross-sections (Figure 3.9(b)), most notably when combined with the kneading operator (Figure 3.10).

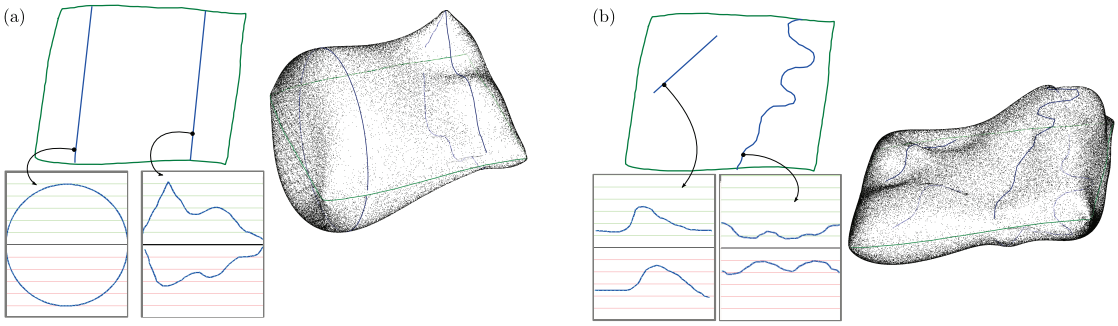



**Figure 3.9:** Using the cross-editing curves to define the inflation: (a) cross-section paths, with left and right curves for the default and free-form profiles, respectively; (b) free-form paths, both curves using free-form sketched profiles.

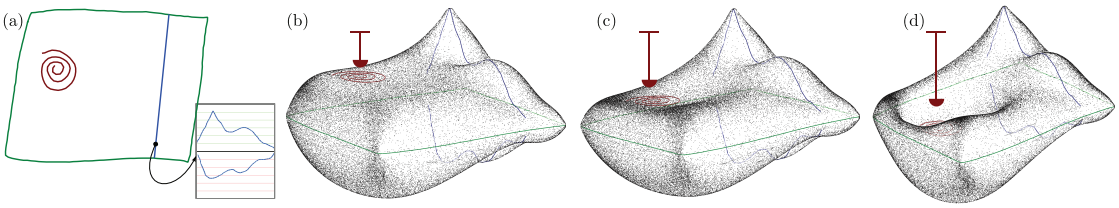



**Figure 3.10:** Combining different kinds of inflation: (a) sketched curves defining the external contour (green), cross-editing (blue) and kneading (red); (b), (c) and (d) final models with different levels of kneading.

### 3.4.3 Oversketching

The oversketching operator is applied over contour curves and is organized in two classes. The first class corresponds to a correction or augmentation to a single curve (Figure 3.11(a)). The user draws a new curve $S$ near an existing curve $C$ to be modified; $S(0)$ and $S(1)$ are then projected onto the line $C$ to define two pieces of $C$ between these two projected points, one of which will be replaced by $S$. The ambiguity on which piece will be replaced is solved by simply enforcing coherence on the orientation of the final curve. The second class of oversketching operator allows to merge two existing curves (Figure 3.11(b)). The user first selects these two curves, $C_1$ and $C_2$, to be merged and then draw two paths, $P_1$ and $P_2$. Next, $P_1(0)$ and $P_2(0)$ are both projected onto $C_1$, while $P_1(1)$ and $P_2(1)$ onto $C_2$. The curve segments between those points are eliminated and the new curve is built by 'gluing' $C_1$, $P_1$, $C_2$ and, $P_2$. For this class of operator, the ambiguities are solved by both the orientation criterion used for the first class of oversketching operator and the order in which the paths are drawn.

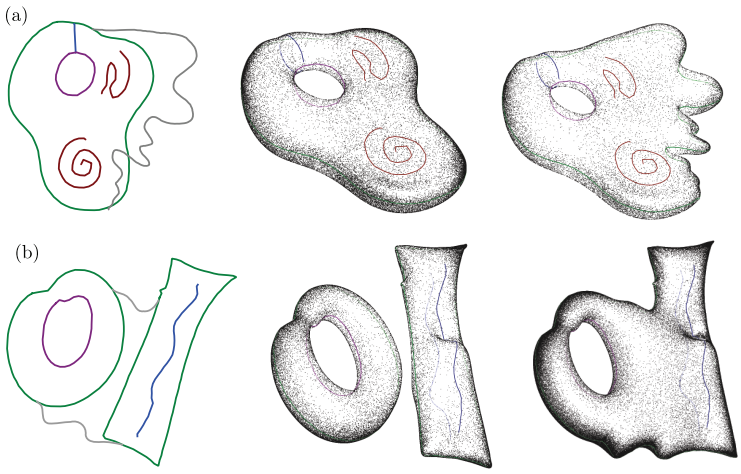



**Figure 3.11:** Oversketching (grey curves): (a) editing a single contour, (b) blending two contours.

## 3.5 Rendering Approach

How to visualize the model is an important issue in modeling applications that use implicit surfaces. Graphical hardware has polygon rendering engines, therefore to visualize the model usually the surface is triangulated using a method for polygonization of implicit surfaces. Additionally the problem of approximating implicit surfaces by polygons is well studied and different methods have been developed, e.g., Bloomental [9] and Velho [63]. However these methods depend on parameters to capture the correct topology of the model. This strategy of triangulating implicit surfaces can be observed in Schmidt et al. [57], where they provide control over the polygonizer resolution, allowing the user to determine the trade-off between accuracy and interactivity (Figure 3.12 left). Bernhardt et al. [7] also use this strategy but their polygonizer resolution is defined automatically by a intrinsic value of their system (Figure 3.12 right).

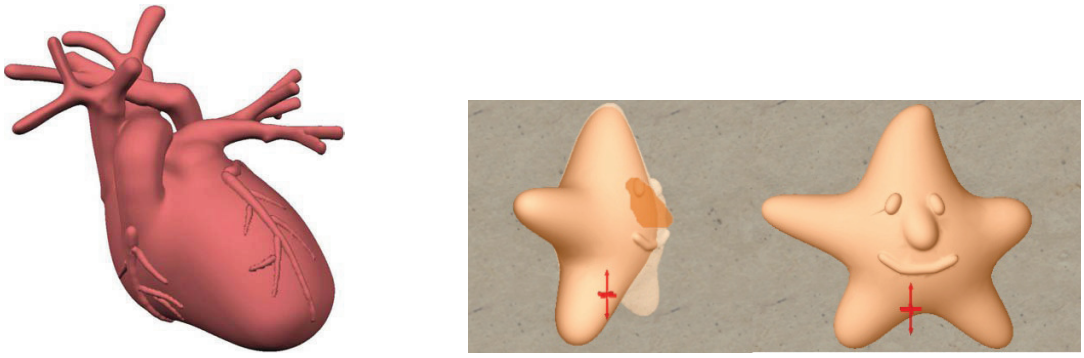


**Figure 3.12:** Typical render style of systems that use implicit surface polygonizer. Left, a heart model from Schmidt et al. [57] and right, a star from Bernhardt et al. [7].

In contrast, our application uses a simplified version of the point-based technique presented by Vital Brazil et al. [67, 69] to render the resulting implicit models. This method was designed to depict shape and tone for implicit surfaces by satisfying on a few basic geometric operations to place and modulate the tone of stippling primitives. Next we present the basic ideas of this technique, which we use to visualize the results of this chapter.

### 3.5.1 Overview of the Rendering Pipeline for Implicits

In the first step of the rendering algorithm we place points onto the surface which will later be used as seeds to create more points over the surface. In [69, Sec. 4] Vital Brazil et al. developed a variety of tools to place seeds; in fact the seed placement is a very important step to define the final rendering quality. Since our main purpose is modeling we use the simplest technique to place seeds: we fill with points the model's bounding box and project these points on the surface. Vital Brazil et al. [69] discuss the issue of bad placement of seeds using this simple approach; however for our application the bounding box strategy is satisfactory.

After the seeds have been placed on the surface, their positions are ready to be used to generate render points. Vital Brazil et al. [69] divide the points in three groups: stippling, principal directions of curvature, and combing directions. Stippling points are placed in a scattered fashion, focusing on covering the surface uniformly, while the two other groups provide linear mark depictions by clustering points along a directional field. We use only stippling points because they provide a good shape depiction with few points. All three groups share the same recursion idea. We use the actual seed position to place a new point near the surface, located in space at a distance $\rho$ from its seed. After that, we project the new points onto the surface. In the next step, all the recently generated points become seeds, and divide $\rho$ by 3 (Figure 3.13). The process goes on until the desired visual effect is achieved. It is important to notice that, by using this 1/3 rule, the distance between a seed and all its descendants is limited; in fact, after $k$ steps, the distance between the original seed and any descendant will be less than $1.5\rho_0$ and two points with the same original seed will be at most $1/3^k\rho_0$ apart.

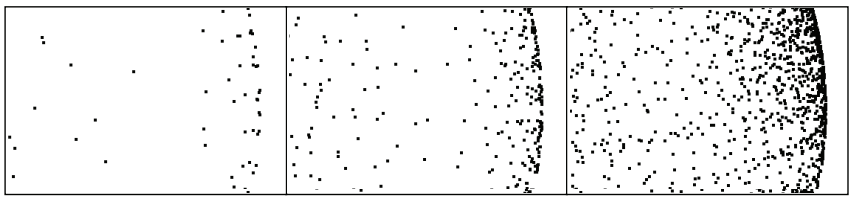


**Figure 3.13:** Multi-Level Sample Refinement. left to right, levels of refinement: one, two, and three.

At this stage, we are ready to use the render points already placed over the surface to visualize the implicit model in different styles. The render points are classified in three sets: front, back, and silhouette. After that, they are assigned a point size and an alpha value and are subsequently sent to the standard graphics pipeline. We calculate $\nu = \mathbf{n} \cdot \mathbf{v}$, where $\mathbf{n}$ is the normal at the point and $\mathbf{v}$ is the viewing vector. Using a small threshold $\delta > 0$, we identify front points when $\nu < -\delta$, back points when $\nu > \delta$, and silhouette points otherwise. After classifying all points, different rendering effects can be created, as described in [69, Sec. 6]. We use in this modeling pipeline two render effects: lighting and silhouette enhancement. The silhouette points are always displayed, back points are plotted using the background color (white) and front points are drawn in black.

In order to enhance the silhouettes the user has the option to draw a thicker line in the tangent direction of the silhouette, i.e., a line with direction $\mathbf{n} \times \mathbf{v}$. Figure 3.14 illustrates this effect.

In our approach, lighting effect is depicted by removing front-points from the surface. The front points are removed randomly, using different probability density functions. Lighting effects are achieved by calculating the tone $\tau \in [0, 1]$ at the point using $\tau = (\mathbf{n} \cdot \mathbf{l})^2$, where $\mathbf{l}$ is the unit light vector. Taking a random variable $u \sim \mathcal{U}[0, 1]$. A front point is displayed only if $u \geq \tau$. Figure 3.15 illustrates this effect.

To keep the modeling sessions interactive, we use coarse samplings to feed our

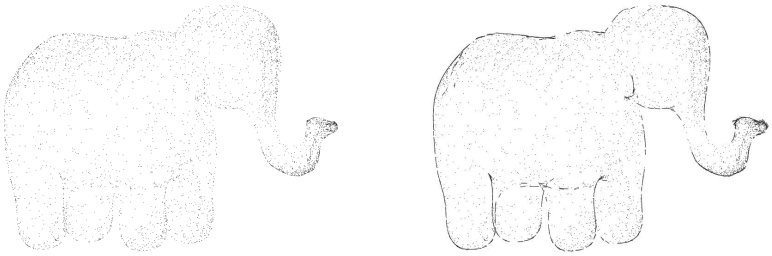


**Figure 3.14:** Elephant model. Left without and right with silhouette enhancement.

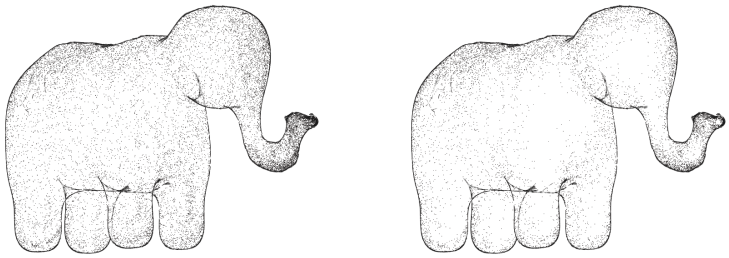


**Figure 3.15:** Elephant model. Left without lighting effect and right with lighting effect.

stippling renderer to achieve fast visualization previews of the implicit surfaces. Figure 3.16 (left) shows a typical model preview computed after a 2s-long preprocessing to place 4K points. If higher quality depiction of the modeled surface is desired, the user can incrementally refine the stippling. A very useful feature of this technique is its ability to depict hidden structures in the surface, providing the user a good perception of occluded regions of the model as well as the spatial relations between the final shape of the model and the strokes being sketched. Figure 3.16 (right) shows a frame from a simple final rendering computed after preprocessing 100K primitives for about 5min.

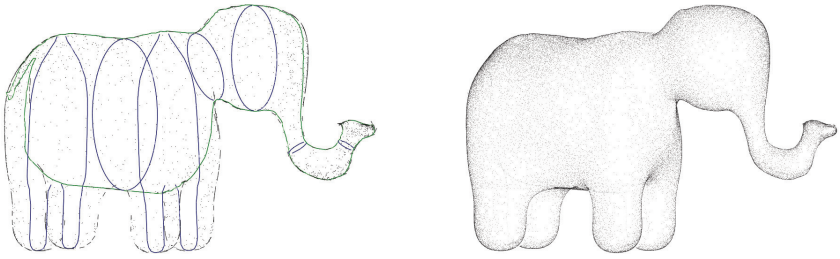


**Figure 3.16:** Elephant model. Left typical model preview and right the final rendering quality.

## 3.6 Results and Discussion

Our SBM techniques successfully model Variational HRBF Implicit surfaces by interpolating the sketched curves, thus preserving the intended shape to be modeled.

All the results were generated on a 2.66 GHz Intel Xeon W3520, with 4 gigabytes of RAM and an OpenGL/nVIDIA Quadro FX 3800 graphics card. The user sketches directly on a Wacom Cintiq interactive pen display. All input strokes preprocessing, Variational HRBF surface fitting, and run-time rendering were computed on the CPU only.

Results were generated for a variety of shapes and objects, sketched by users with different drawing abilities. We observed that the overall modeling time (from the initial input strokes to the finished model) took around 15 minutes. As expected more samples result in more complex Variational HRBF computations. However, we observed that these computations took less than 5 seconds. Our approach produces promising results requiring few strokes from the user to construct a model while preserving the intended shape. We evaluated our results by observing the timings for the overall modeling stages and the quality of the final model.

Figures 3.7 to 3.11 show examples of each of our sketch-based modeling operators. Figure 3.3 shows the result of modeling a simple mug. The user sketched 17 strokes (including cross-editing and oversketching curves), resulting in 720 samples, fitted in less than 2 seconds. The overall modeling (from initial strokes to finished model) took 12 minutes. Figures 3.1 and 3.17 show the result of sketching directly over a reference image to reconstruct its 3D shape. For the rhinocerus head (Figure 3.1), 11 strokes were sketched, resulting in 490 samples, fitted in less than 1 second. The overall modeling took 9 minutes. For the rubber duck model (Figure 3.17), the user sketched 19 strokes, resulting in 989 samples, fitted in less than 3 seconds. The overall modeling took around 14 minutes. Figure 3.18 shows the result of modeling a terrain in 22 strokes, 838 samples, fitted in less than 3 seconds. The overall modeling took about 16 minutes.

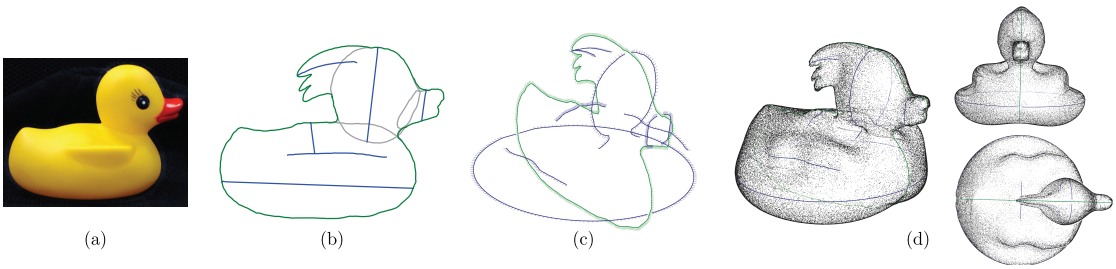



**Figure 3.17:** Modeling a rubber duck: (a) reference photograph, (b) input strokes sketched directly over the photograph, (c) 3D construction curves with normals and (d) stipple rendering of the final model.

## 3.7 Conclusion and Future Work

In this chapter, we introduced a representation for variational implicit surfaces suitable for sketch-based modeling. This representation retains the good properties of the currently used techniques for variational implicits while additionally being more robust to the sparse and coarse samplings resulting from sketch modeling sessions

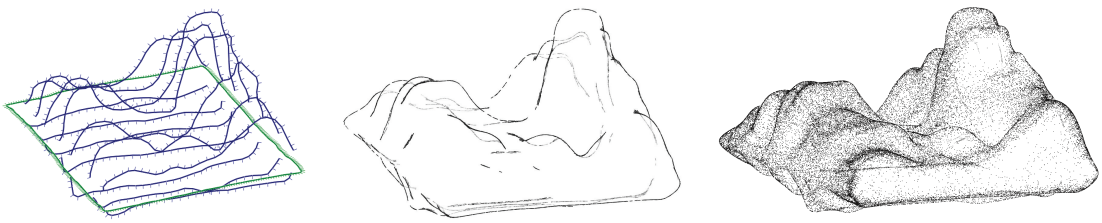


**Figure 3.18:** Modeling a terrain: construction curves, silhouettes and stippling.

even in the presence of close-sheets. Moreover, the representation does not depend on artificial parameters commonly found in other RBF-based methods, e.g. offsets and support radii.

We employ this representation as the basis for sketch-based modeling system for which we introduce a small, albeit powerful, set of simple modeling operators for contouring, cross-editing, kneading, oversketching and also merging different parts of a model.

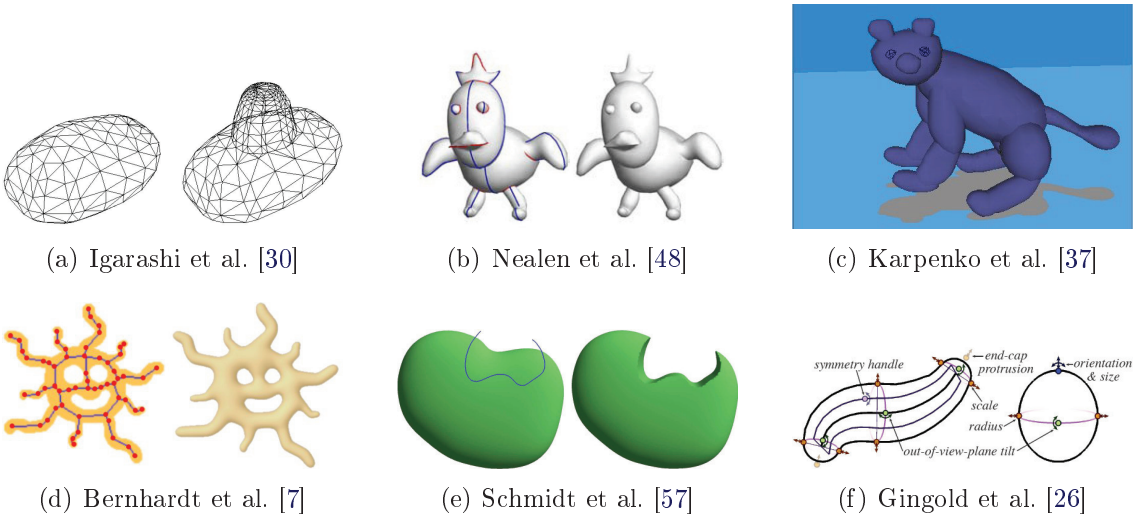
There are still many avenues for improvement and further investigation. On the modeling side, we plan to extend the language and operators to support more complex edition tasks as well as specific application-domains. Also, there is the need for a systematic usability evaluation in order to assess the language and operators and point directions to enhance them as well as our system's user interface.

# Chapter 4

# A Surface Representation for Sketch-Based Surface Modeling

In this chapter we present a novel way to think the sketch-based surface modeling framework. Until now we have a specific mathematical representation for our final object and we suit the modeling process to this representation. Specifically, in Chapter 2 the representation consisted of samples interpolated with an Hermite-Birkhoff RBF, while the object being modeled was warping fields; in Chapter 3 we use the Hermite RBF interpolant to model implicit surfaces. Now we propose a more general mathematical representation for surfaces, conceived for specific sketch-based surface modeling processes. We conceived this representation with two main goals in mind: good control for global and local transformations and flexibility to be applied in different kinds of sketch-based modeling systems. We also desire two secondary qualities: the ability to save and edit modeling stages, and allowing templates.

There are many ways to represent surfaces in $\mathbb{R}^3$, the most common and general are implicit and parametric representations. However, to be used in computer graphics applications, these representations should be more specific and have practical qualities. As examples we can cite the: BlobTree (Wyvill et al. [74]), piecewise algebraic surface patches (Sederberg [58]), convolution surfaces (Bloomenthal and Shoemake [10]), generalized cylinders, polygonal meshes, subdivision surfaces, among others. Different representations for sketch-based modeling have been explored since the seminal work of Igarashi et al. [30]. Notably polygonal mesh representation was used by Igarashi et al. [30] and Nealen et al. [48], implicit RBF used by Karpenko et al. [37] and Vital Brazil et al. [68], Bernhardt et al. [7] use convolution surface, Schmidt et al. [57] create a system to model BlobTrees, Gingold et al. [26] use generalized cylinders, among others (Figure 4.1). One natural question is "Which is the best surface representation for the problem of SBM?". This is certainly an open question, and probably does not have only one answer. We believe that a good surface representation for sketch-based modeling should be tailored to each specific application. Take the case of modeling terrains where we want organic forms, and symmetry and perfect sharp edges probably are not desired. On the other hand, when modeling a chair we desire the opposite. It is a hard problem to find a surface representation that has all these different properties. However, there are




(a) Igarashi et al. [30] (b) Nealen et al. [48] (c) Karpenko et al. [37]

(d) Bernhardt et al. [7] (e) Schmidt et al. [57] (f) Gingold et al. [26]

**Figure 4.1:** Different sketch-based modeling systems uses different representations. Polygonal mesh representation (a) and (b), implicit RBF (c), convolution surface (d), BlobTrees (e), and generalized cylinders (f).

general aspects that are desired in all sketch-based surface modeling applications. Therefore we will discuss these properties and present an abstract representation that is suitable for different applications.

In the next section we present our abstract representation and discuss its properties. In Section 4.2 we show a practical example of application of this representation with a simple implementation that exploits its main characteristics.

## 4.1 A Composite Surface Representation

The main idea that we are proposing is to separate the surface representation into levels of detail and properties. This is not a new notion in computer graphics; many works have developed this concept for specific domains and applications. Blinn [8] introduces the idea of bump-mapping that stores geometric information at two levels, the base geometry and a displacement map that is used to create rendering effects. The same concept is found in Krishnamurthy and Levoy [38] and Lee et al. [40]; however, in these two works the authors suggest other applications besides rendering, such as animation and compression. Inspired in these works, Pereira [51, Chapter 4] presents a representation that uses a smooth base geometry and a map of normals to handle details over the surface. All these works have two different types of data, the first one defines the smooth geometry and the second which maps the first one in a parametric space that stores details, similar to a texture mapping.

The main goal of our study is to be able to control local editing without changing parts of the model out of the region of interest, and when big deformations are introduced, details are kept coherently. Hence, we believe that decomposing the model representation into a base surface that supports different types of properties is a powerful tool for sketch-based surface modeling. Now we will describe an abstract

representation and introduce some notation.

Let the model $\mathcal{S}$ be a manifold $S \subset \mathbb{R}^d$ and a set of properties $\mathcal{P} \subset \mathbb{R}^k$, i.e., $\mathcal{S} \subset \mathbb{R}^d \times \mathcal{P}$, and let $D : \mathcal{S} \to \mathbb{R}^d \times \mathcal{P}$ be a transformation map of $\mathcal{S}$. We define our new model as:

$$\widetilde{\mathcal{S}} = \mathcal{S} \oplus D(\mathcal{S}). \tag{4.1}$$

Typically $d = 3$, $\mathcal{S}$ is the low frequency information of surface $S$, while, $D(\mathcal{S})$ performs local changes. The properties $\mathcal{P}$ can be geometric, such as the normal at a point, or general, such as color, or many attributes together. The binary operator $\oplus : \left(\mathbb{R}^d \times \mathcal{P}\right)^2 \to \mathbb{R}^d \times \mathcal{P}$ defines how to compose $\mathcal{S}$ and $D(\mathcal{S})$.

A simple example of how to translate this representation for a concrete use is bump-mapping [8]. In this case, $S$ is a smooth surface with $\mathcal{P} \subset S^2 \times \mathbb{R}$, for instance $\mathcal{P}_p = (N_p, h_p)$, thus $\mathcal{S} \subset \mathbb{R}^3 \times S^2 \times \mathbb{R}$ , $D(p) = (h_p \cdot N_p, 0, 0)$ where $N_p$ is the normal at point $p$ and, $h_p$ is the hight of its displacement in the normal direction. Finally $\oplus$ is the usual vector sum in $\mathbb{R}^{3+3+1}$, then $\widetilde{\mathcal{S}} = S + D(S)$. Another example is the RGBN image used in the Chapter 2, where $S$ is the image support, $\mathcal{P}$ is color, normals, and warping field, and finally $D$ applies the warping field. If we have an alpha-channel then $\oplus$ is the composite operator of images using alpha-channel.

For many cases we will need a more structured function $D$. The natural way to construct $D$ is by using an atlas $\mathcal{A}$ of $S$ and an auxiliary parametric space $\mathcal{P}'$ for properties, and two auxiliary functions to define $D$:

$$\begin{aligned} \Psi &\colon \mathcal{S} \to \mathcal{A} \times \mathcal{P}' \\ \Phi &\colon \mathcal{A} \times \mathcal{P}' \to \mathbb{R}^d \times \mathcal{P} \end{aligned} \tag{4.2}$$

so that

$$D = \Phi \circ \Psi. \tag{4.3}$$

We will call $\Psi$ the *parametrizer* of $\mathcal{S}$ and $\Phi$ the *deformation* function.

It is important to note that we do not restrict $S$ to any particular kind of surface. Generally the application only requires some properties of $S$, e.g., smoothness, normals, curvatures, or how to project a point onto $S$. These attributes should be defined for each specific use.

This formulation allows us to define a composition process that we can define using different notations. The first one is a simple idea of function composition; we replace equation (4.1) by:

$$\widetilde{\mathcal{S}} = \mathcal{S} \oplus_1 D_1 \oplus_2 D_2 \cdots \oplus_n D_n,$$

where $D_j = D_j\left(\widetilde{\mathcal{S}}^{j-1}\right)$, $\widetilde{\mathcal{S}}^{j-1} = \mathcal{S} \oplus_1 D_1 \cdots \oplus_{j-1} D_{j-1}$ and $\mathcal{S}^0 = \mathcal{S}$. In particular all $D_j$ and $\oplus_j$ have the same behavior of $D$ and $\oplus$ of equation (4.1). If we suppose that $\widetilde{\mathcal{S}}$ has the same properties of $\mathcal{S}$, i.e., $\widetilde{S}$ is a manifold, we can consider $\widetilde{\mathcal{S}}$ as $\mathcal{S}$ and have the sequence: $\mathcal{S}^{m+1} = \mathcal{S}^m \oplus_m D_m(\mathcal{S}^m)$. This sequence has the semantics of modeling stages: after $n$ modeling stages our final surface is $\mathcal{S}^n$. The concept of layers can be applied together with the equation (4.3) and parameter space of

(4.2), creating a local composition, i.e., the deformation function $\Phi_j$ of $D_j$ with a composition operator $\otimes_j$ that affects each chart of $\mathcal{A}$ separately. Writing

$$\ell = \ell_1 \otimes_2 \ell_2 \otimes_3 \cdots \otimes_n \ell_n, \tag{4.4}$$

where $\ell_j : \mathcal{A} \times \mathcal{P}' \to \mathcal{A} \times \mathcal{P}'$. We see $\ell_j$ as layers over the atlas $\mathcal{A}$, for each chart in $\mathcal{A}$ they can have the same concepts of image layer composition. Introducing the notation $D^*(\mathcal{S}) = \mathcal{S} \odot \ell(\Psi(\mathcal{S}))$, where $\odot : \mathcal{S} \times \{\mathcal{A} \times \mathcal{P}'\} \to \mathbb{R}^d \times \mathcal{P}$ we have:

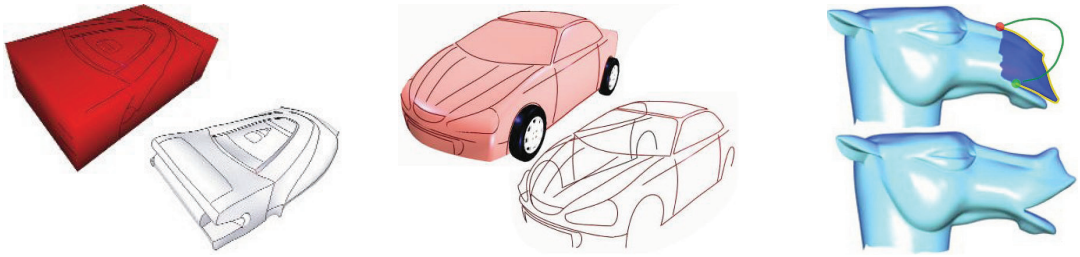
$$\widetilde{\mathcal{S}} = \mathcal{S} \oplus D^*(\mathcal{S}). \tag{4.5}$$

These definitions have many advantages which we will discuss in the next paragraphs, but an issue should be observed. It is very hard to constrain $\mathcal{S}$ and $D$ to $\widetilde{\mathcal{S}}$ be a manifold without restrict too much them, indeed, if you try to force $\widetilde{\mathcal{S}}$ to be a manifold; probably the system will be very limited. But it is not a real problem because this representation will be used in sketch-based modeling, and the user can decide if the result is desired or not. One simple example of restriction is inspired in the example of bump-mapping [8]; if we in fact displaced the points using $h$, it is easy to find a height that results in a non-manifold object, however if we constrain $h$ to be less than the radius of the tubular neighborhood we do have a manifold (do Carmo [19]).

**Control: Local $\times$ Global.** In many modeling pipelines the difference between global and local modeling stages appears naturally. Take the case of clay potter as described by Woody [73]: the artist starts by creating an overall pottery shape using a wheel; that is an example of *global* modeling. Later the artist refines and clarifies a form, rather than totally change it, paddling the clay with a slightly absorbent tool, such as a wood spoon, that means the potter desires *local* editing tools. Keeping this example in mind, from now on, the base surface $\mathcal{S}^0$ is interpreted as the first approximation of our final surface, to put it in another way, the coarse shape with the basic attributes. Equally important, $D$ will handle the detail information. Hence we can group the tools that manipulate our model in two sets, one that work globally which creates and edits $S^0$ and another that works locally which creates and edits $D$. In short, this representation will be natural for control global and local deformations.

**Modeling Stages and Templates.** When artists draw complex models they always start with simple lines and then pass to the next level of detail, they add more and more detail and often save old lines to use as reference or to come back to a specific step (Ryder [54]). Hence, it is natural that some applications should save the steps of modeling, i.e., the system allows the user to recover an early modeling stage and edit it. We can interpret template-based modeling as "recover early stage", where the user chooses a predefined object and edits its properties or creates new features based on the existing attributes. To illustrate, in sketch-based modeling the works of Kara and Shimada [34, 35] and Nealen et al. [47] apply templates in different ways (Figure 4.2). The first one [34] uses the template shapes to help the user to sketch the first lines of the model. With a more specific application

(car industry) in [35] a basic mesh is adapted to a car sketch and then the user can edit the model using the adapted mesh as template. On the other hand, the work [47] starts its modeling pipeline with a triangle mesh and regions of interest over this mesh, and by using sketches the user reshapes the model. To Translate these template concepts to our representation is simple; the template is $\mathcal{S}_0$ and the subsequent modeling stages define $D$'s. We can cite two examples of sketch-based modeling system that approximate the semantic of modeling stages, Nealen et al. [48] and Vital Brazil et al. [68]. Both keep the constructions curves and allow the user to edit them, which has a similar semantics of drawing stages however there is not a hierarchic sequence of stages. This hierarchy can be created saving the stages in groups of separate layers $D_j$.



(a) Kara and Shimada [34] (b) Kara and Shimada [35] (c) Nealen et al. [47]

**Figure 4.2:** Examples of sketch-based modeling systems that use templates.

## 4.2 A Sketch-Based Modeling Application

We implemented a modeler that uses the concepts discussed in the previous section in a pipeline tailored to use this representation. Although we advocate that SBSM should be created for specific domains, to test how far a simple sketch-based system goes using our pipeline we implemented a system that does not have a definite domain. The main focus of our system is to guarantee the control over global and local changes in the model, i.e., when the user adds detail over the model, the base surface should not change and when the base model is changed all details should change consistently with the global transformation done. In the next sections we present the pipeline and its parts.

### 4.2.1 Pipeline

We propose a pipeline divided into four parts; each part plays a fundamental role in our system and is strongly linked with the representation proposed in Section 4.1. We start with the coarse form defined by an implicit surface (Chapter 3); after that we build a base mesh (Section 4.2.2) that has the same topology and approximately the same geometry of the implicit surface. The base mesh induces an atlas (Section 4.2.3) and provides a 4-8 base mesh (Section 4.2.4). The atlas is built using a partition of the set of faces of the mesh, and we use it to edit the model locally. The 4-8

mesh has two roles in the pipeline: to build a map between surface and atlas and to visualize the final surface. After we have all parts the 4-8 mesh is used to edit details that are saved in the atlas, and the atlas maps details onto the 4-8 mesh. In Figure 4.3 we depict our pipeline.

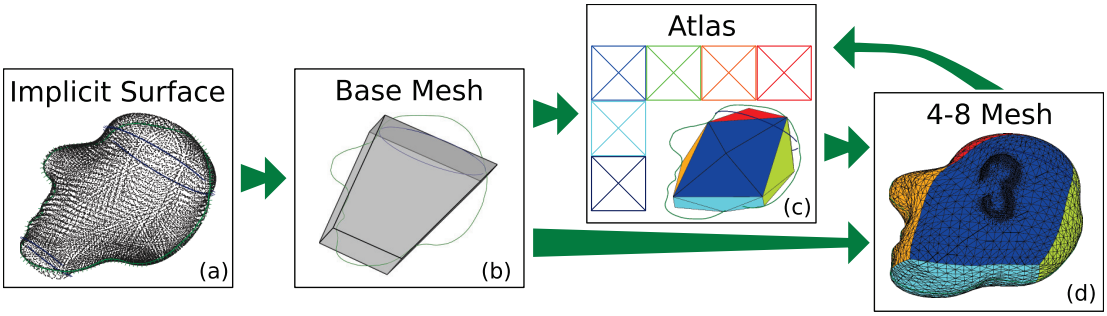



**Figure 4.3:** The pipeline of our Sketch-based surface system. The arrows depict the information flow.

The first pipeline step is to obtain a coarse shape of the final model (Figure 4.3(a)). As discussed in Chapter 3, implicit models provide a compact, flexible, mathematically precise representation. We use the same implementation described in Chapter 3, the implicit representation (HRBF) used fits well with our pipeline because it has good projection properties as well as being simple and compact. In the terminology of Section 4.1 this implicit model and its normals are the $\mathcal{S}_0$ of our representation.

After we obtained $\mathcal{S}_0$ we need to create the $D^*$ functions. We adopt equation (4.5) for our system, i.e., we use a parametric space to handle details. However, we have an implicit surface without information about atlas. There are many approaches to parametrize implicit surfaces, e.g., Bloomenthal [9], Velho [63], Stander and Hart [60], but to find the correct topology of the model, these approaches depend on parameters [9, 63] or require differential properties of the surface [60]. Besides the topology problem such methods neither guarantee the mesh quality nor have a direct way to build an atlas structure. As a result, we opted to develop a method that is based on our problem and desired surface characteristics. First of all, observe that we have two different scales of detail to be represented: $\mathcal{S}_0$, which is coarse, and $D^*$, which is finer. The naive approach is to use the finest scale of detail to define the mesh resolution. However, there are two problems: we do not know this finest scale a priori and if the details appear in a small area of the model we will waste memory and time with a heavily refined mesh.

To avoid these problems, we adopted a dynamic adaptive mesh, the semi-regular 4-8 mesh (Velho [65]): using it we can control where the mesh is fine and coarse using a simple error function. Going back to the problem of parametrization of our implicit surface, now we want more than just a mesh: we need an adaptive mesh. The framework presented by de Goes et al. [16] starts with a semi-regular 4-8 mesh and refines it to approximate surfaces using simple projection and error functions. From now on we say 4-8 mesh in place of semi-regular 4-8 mesh. To obtain a good approximation of the final surface, the 4-8-base-mesh should have the same topology

and should approximate the geometry of the final surface. Thereupon we exchange the problem of parametrization for how to find a good 4-8 base mesh (Section 4.2.2), and how to construct a good error function (Section 4.2.4).

The parametrization of the implicit surface is built in three parts: base mesh (Figure 4.3(b)), atlas (Figure 4.3(c)), and semi-regular 4-8 mesh (Figure 4.3(d)). In Section 4.2.2 we present a base mesh which has two roles in our system, to induce an atlas for the surface and to create a 4-8 mesh. We develop a method in Section 4.2.3 to create an atlas for adaptive meshes based on stellar operators. In Section 4.2.4 we discuss how build an error function for the 4-8 mesh that is sensitive to levels of detail.

### 4.2.2 Base Mesh

The base mesh is the first step to parametrize our surface. This is a very important piece of our pipeline because three important aspects of the final model depend on the base mesh: the topology of the final model, the atlas, and the 4-8 mesh quality. Finding good base mesh only using information from the implicit model is a very hard problem in geometric modeling, but since we are proposing sketch-based modeling, it is natural that we use the user input to get more information about the model and to create the base-mesh.

The user works with a simple unit of tessellation element (tesel) which can have the topology of a cube or torus. This tesel is projected onto the drawing plane (Chapter 3) and the user edits it to get a better approximation of the model (geometric and topological). The user can move tesel vertices in the plane, divide a tesel or change its topology. Afterwards the system creates a tessellation in the space by moving each tesel vertex along the draw plane's normal direction. In Figure 4.4 we show the typical steps to create the base mesh. The user starts with a bounding box of the sketched lines, then divides tesels, moves vertices and changes tesel topology to build a better approximation of the intended shape. Our system defines the vertices heights seeking along the draw plane's normal direction for a root of the implicit surface. Each face defines a chart and after that, it is triangulated to be the 4-8 base mesh.

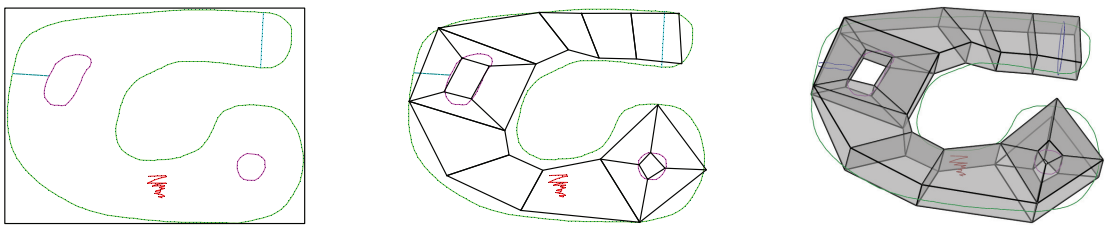


**Figure 4.4:** Creating a base-mesh for an implicit surface created using the construct lines described in Chapter 3. Left to right, the first approximate, after the user correct the topology and better approximate the geometry, and the final result en $\mathbb{R}^3$.

### 4.2.3 Atlas

The second step to obtain a manifold structure for our model is to construct an atlas, i.e., a collection of charts $c_i$ that are open sets $\Omega_i \subset \mathbb{R}^2$, and functions $\phi_i : \Omega_i \to S$ that are homeomorphisms (do Carmo [19]). Besides that, in our system the atlas $\mathcal{A}$ is associated with a parametric space that helps to define $D$, the $\mathcal{P}'$ in (4.2). Specifically for this application, each chart of $\mathcal{A}$ is associated with a height-map. This height-map is used to define a displacement along the normal direction, and in Section 4.2.4 we use that height-map to define an error function that locates where the 4-8 mesh need be more refined. We create two types of height-map layers: pre-loaded gray images and height-maps directly sketched on the surface.

In Figure 4.5 we depict the steps to create an atlas for a 4-8 mesh $M$. After the base mesh is obtained and each of its faces is triangulated, one refinement step is done and then each base mesh face is associated with a chart (Figure 4.5(a)). When the mesh is refined to better approximate the geometry, the atlas is updated and the user can draw a curve over the $M$ which is transported to the charts and then this curve creates/edits the height-maps (Figure 4.5(b)). If the mesh resolution is not enough to represent the details, $M$ need be refined; usually that happens when the user creates/edits the height-maps (Figure 4.5(c)). Next we describe our method to build this atlas and the sketched height-maps.

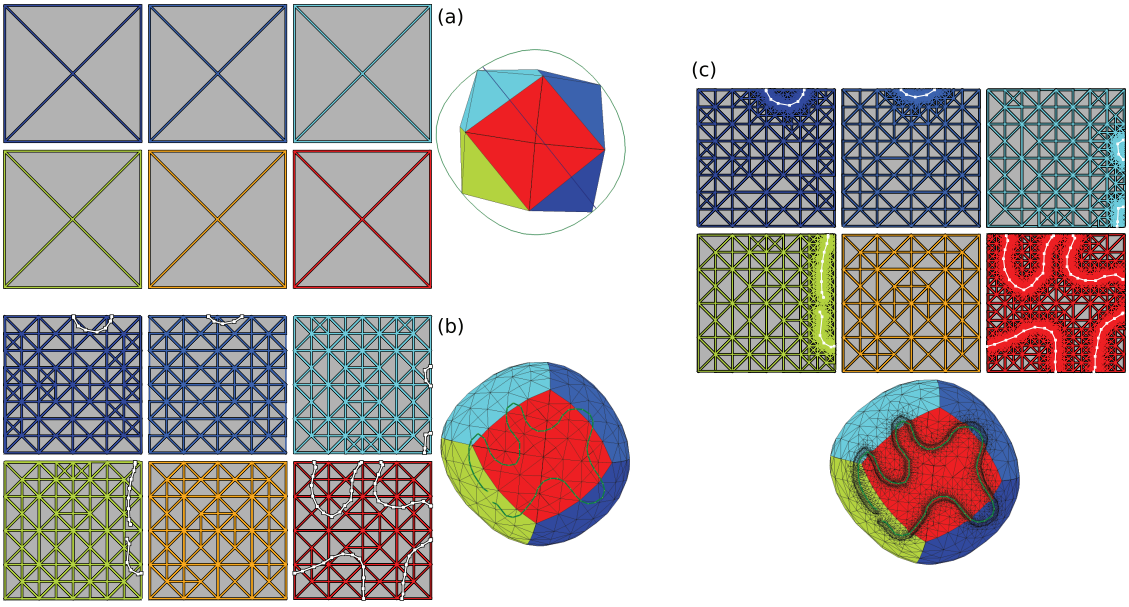



**Figure 4.5:** Atlas steps: (a) The atlas is defined after one refinement step of $M$. (b) $M$ is refined and the user defines an augmentation sketching over the surface, and the sketches are transported to $\mathcal{A}$ to built a height-map. (c) $M$ is refined to represent details of the final surface with height-map.

### Vertex-Map

In this section and in the next we construct the theoretical framework to build an atlas using a label function over the vertices of a mesh. We work with a general description of adaptive surfaces, based on *stellar subdivision grammars* (Velho [64]). Our choice for parametric representation, the 4-8 mesh developed by Velho [65], is an example of application of this grammar. The atlas defined using vertices of the mesh has the following advantages: it is compact and simple; it naturally classifies edges as inner and boundary; and it is suitable for work with dynamic adaptive meshes.

As discussed before we need an adaptive mesh to represent the high-frequency details. However, when we do one step of refinement in a mesh, new elements (vertices, edges, faces) are created, hence we need to update the atlas. The problem that we propose here is how to construct and update an atlas using the natural structure of adaptive surfaces.

First of all we formalize the concept of the *regular labeled mesh*. After that we use these definitions to build an atlas with guarantees for adaptive surfaces that uses Stellar subdivision operators.

**Definition 4.1.** A mesh $M = (V, E, F)$ is *$k$-labeled* if each vertex $v \in V$ has a label $L(v) \in \{0, 1, 2 \ldots, k\}$, i.e., if there is $L : V \to \{0, 1, 2 \ldots, k\}$. $L$ is called *$k$-label function*. If $L(v) = i \neq 0$, then $v$ is an *inner-vertex* of the chart $c_i$; if $i = 0$, $v$ is a *boundary-vertex*.

**Definition 4.2.** A face $f \in M$, is *regular $k$-labeled* or *$rk$-face* if there is $v \in f$ with $L(v) \neq 0$ and $\forall\ v_1, v_2 \in f$ such that $L(v_1) \neq 0 \neq L(v_2) \Rightarrow L(v_1) = L(v_2)$. A mesh is *regular $k$-labeled* (or *$rk$-mesh*) when all their faces are $rk$-faces. The function $L : V \to \{0, 1, 2 \ldots, k\}$ that produces a $rk$-mesh is called a *regular $k$-label* or *$rk$-label*.

Observe that an edge in a regular $k$-labeled mesh has vertices with the same label or one of them has label 0. If the edge has at least one vertex $v$ such that $L(v) = i \neq 0$; we call it an *inner-edge* of the chart $c_i$ or $L(e) = i$; if it has the two vertices labeled as zero it is a *boundary-edge* or $L(e) = 0$.

**Proposition 4.3.** *A regular $k$-label function induces a partition on the set of faces.*

*Proof.* Let $M = (V, E, F)$ be a $rk$-mesh. Define the set $F_i = \{f \in F |\ \exists\ v \in f$ such that $L(v) = i\}$, $i \in \{1, 2, \ldots, k\}$. By definition 4.2 every $f \in F$ has at least one $v$ with $L(v) \neq 0$ then:

$$\bigcup_{i=1}^{k} F_i = F,$$

and if there is more than one $v \in f$ such that $L(v) \neq 0$ then all such vertices will have the same value of $L$, i.e., the face belongs to only one $F_i$, so we conclude:

$$F_i \cap F_j = \varnothing \quad \text{if} \quad i \neq j.$$

□

This proposition allows us to define a collection of charts over a $rk$-meshes. We say that a face $f$ is in the chart $c_i$ ($L(f) = i$) if there is at least one $v \in f$ such that $L(v) = i$. However for our application it is not enough to have a static map because our mesh is adaptive. Hence we need rules to assign a $L$ value to the new vertices created by the refinement step of the mesh.

We develop our techniques based on the two works by Velho [64, 65] on stellar operators and dynamic adaptive meshes. We are working with 4-8 mesh which is a multi-resolution triangle-mesh for manifold surfaces. The 4-8 mesh uses to refine and simplify the mesh the stellar operators that come from the theory of the stellar subdivision [42]. Hence we study how to update the atlas after we apply one of these operators: *edge split*, *face split*, and their inverse *edge weld* and *face weld* (Fig. 4.6). We use the concepts of sequence of meshes $(M_0, M_1, \ldots, M_k)$ and *level* of a mesh element exactly as presented by Velho [64].

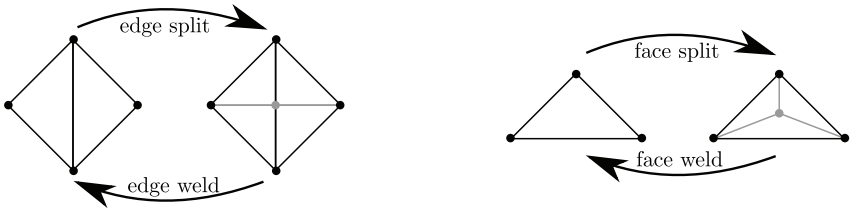



**Figure 4.6:** Stellar subdivision operators and inverses. (Illustration inspired in Velho [64])

When we apply one of these two subdivision operators (split), it adds only one vertex. As a result, to update the atlas we only need rules to label the new vertex $v_n$ created for these two operators over a $rk$-mesh.

- **Face Split** – when the face $f$ is split we define:

$$L(v_n) = L(f) \tag{4.6}$$

- **Edge Split** – when the edge $e$ is split we define:

$$L(v_n) = L(e) \tag{4.7}$$

**Proposition 4.4.** *A stellar subdivision step using the previous rules on a rk-mesh $M$ produces $M'$ that is a rk-mesh too.*

*Proof.* Case we split a face $f$ we create 3 news faces $(f_1, f_2, f_3)$, since $M$ is a $rk$-mesh the equation 4.6 is well defined. Moreover $v_n \in f_1 \cap f_2 \cap f_3$ then they have at least $v_n$ with $L(v_n) = i \neq 0$. Since $f$ is a $rk$-face all $v \in f$, $L(v)$ is 0 or $i$, and for $j = \{1, 2, 3\}$, $v \in f_j \Leftrightarrow v = v_n$ or $v \in f$, we conclude if $v \in f_j \Rightarrow L(v) = 0$ or $L(v) = i$, i.e, $f_j$ is a $rk$-face.

The edge split create four new faces $f_j, j = 1, 2, 3, 4$. Note that the operator edge split subdivides two faces; lets name these faces *west-face* ($f^w$) and *east-face* ($f^e$); and their opposite vertex as $v_e$ and $v_w$ respectively. i.e., $v_* \in f^*$ and $v_* \notin e$.

If $e$ is an inner-edge then for at least one of its vertices $L(v) = i \neq 0$. Since $e$ is in $f^w$ and $f^e$ we have $L(f^w) = L(f^e) = i$ it implies that if $v \in f^w \cup f^e$ then $L(v) = i$ or $L(v) = 0$. As a result when we split a inner-edge we have $L(v_n) = i$ and $v_n \in \bigcap_j f_j$ and $v \in f_j \Rightarrow v \in f^w \cup f^e$ or $v = v_n$, then $f_j$ is a $rk$-face.

The fact of $e$ being a boundary-edge and $f^w$ and $f^e$ be $rk$-faces imply $L(v_e) \neq 0$ and $L(v_w) \neq 0$. Since $v_w \in f_j$ or $v_e \in f_j$ we have at most one $v \in f_j$ such that $L(v) \neq 0$ and $L(v_n) = 0$, then we conclude that $f_j$ is $rk$-face. □

The simplification step of an adaptive mesh is very important for our application, because when the user changes the sketches the mesh is dynamically updated that implies that the two steps (refinement and simplification) are done. If we start with a $rk$-mesh (level 0) and perform $n$ refinement steps for any $m \leq n$ steps of simplification we yet have a $rk$-mesh. This fact is easy to see because when we do a refinement step we do not change the value of the vertices of the current level $j$, thus when we do the inverse operator to simplify only vertices of level $j + 1$ are deleted so then the $L$ function over faces is well defined in level $j$.

To create a $rk$-mesh using our base-mesh, i.e., to create the $M_0$, we label all vertices of the base-mesh as boundary ($L(v) = 0$) and split each face, the new vertex added is labeled with a new value not 0. After that each face of the base-mesh generates a new chart into the atlas, i.e., if the base mesh had $k$ faces the atlas has $k$ charts. In Figure 4.7 we illustrate the process of creating a mesh $M_0$ that is a $r2$-mesh and three refinement steps.

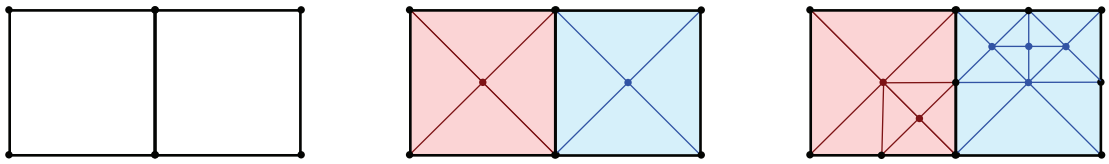


**Figure 4.7:** Creating a $r2$-mesh and refinements. Left to right: the base-mesh, $M_0$ which is $r2$-mesh, and after 3 refinement steps: $M_3$. Black elements are boundary ($L(\cdot) = 0$), blue elements are into chart $c_1$ ($L(\cdot) = 1$), and red elements are into chart $c_2$ ($L(\cdot) = 2$).

### Creating a Manifold Structure

Now we have a partition over the surface and we know how to refine and simplify the mesh respecting this partition. However we do not have yet all elements of an atlas for the surface $S$, is missing to define open set $\Omega_i \supsetneq c_i$ and homeomorphisms $\phi_i$.

First of all let us built $c_i$. We define the functions $\phi_i : \Omega_i\big|_{c_i} \rightarrow S$ based on the structure of adaptive levels of the 4-8 mesh $M$. We define $c_i = [0, 1] \times [0, 1]$, then we set the four vertices of the base-mesh face $f_i = \{v_1, v_2, v_3, v_4\}$ to be the boundary of $c_i$, i.e., the local coordinates in $\Omega_i$ of these vertices are: $v_1^i = (0, 0)$, $v_2^i = (1, 0)$, $v_3^i = (1, 1)$, $v_4^i = (0, 1)$. We overload the notation for chart, when we refer a chart of our atlas, $c_i \in \mathcal{A}$, that has two meanings, the first is a set of faces, edges and vertices, used in previous section. The second is the parametric space $[0, 1]^2 \subset \Omega_i$,

more precisely when we say a point of $M$ belongs to a chart $c_i$ it means if we can write this points in $\Omega_i$ coordinates then its coordinates is in $[0,1]^2$. At this point all vertices $v$ of $M$ have at least two geometrical information its coordinates in $\mathbb{R}^3$ and its coordinates in at least one $\Omega_i$. We will use the notation $v^i$ to be clear when we are using $v$ in coordinates of $\Omega_i$, how to recover this information we will discuss later.

Since $M$ is an adaptive mesh and now it has two geometrical aspects, its coordinates in $\mathbb{R}^3$ and in $\mathcal{A}$, we need rules to update this information. When we split an edge $e = \{v_1, v_2\}$ we get its middle point $v_m$ and project it on $S$ and if $e \in c_i$ then $v_m^i = (v_1^i + v_2^i)/2$. Despite the rules to be simple they achieve goods results, in Section 4.2.4 we discuss more about that rules. Now we suppose that the approximation of the adaptive mesh is less than $\varepsilon > 0$, i.e., for all points $p$ on $M$ imply $|p - \Pi_S(p)| \leq \varepsilon$ where $\Pi_S(p)$ is the projection of $p$ on the surface $S$. We are assuming that $\Pi_S$ is well defined for $\mathcal{V}_\varepsilon = \bigcup_{p \in S} \mathbf{B}(p, \varepsilon)$, where $\mathbf{B}$ is the open ball with center $p$ and radius $\varepsilon$. That is true when $\mathcal{V}_\varepsilon \subseteq \mathcal{V}$ the tubular neighborhood of $S$ (Velho and Gomes [66]). Particularly, the vertices of the base mesh start close to $S$ then their projection is well defined, therefore we replace their start position $v$ by $\Pi_S(v)$. We will also use the $\Pi_M(p)$, the projection of $p \in S$ on $M$, and again we are supposing that the mesh approximates well the surface. We say the chart $c_i$ is well defined after one refinement step (Figure 4.7), thereupon if a point $p^i \in c_i$ then there is a face $f^i = \{v_1^i, v_2^i, v_3^i\}$ such that $p^i$ is a convex combination of its vertices. More precisely $p^i = \sum_{k=1}^3 \alpha_k v_k^i$ with $\alpha_k > 0$, $\sum_{k=1}^3 \alpha_k = 1$. So then we define:

$$\phi_i(p^i) = \Pi_S \left( \sum_{k=1}^{3} \alpha_k \phi_i(v_k^i) \right).$$

Specifically when we split an edge $e$, which belongs to $c_i$, $e^i = \{v_1^i, v_2^i\}$ we have:

$$\phi_i(v_n^i) = \Pi_S \left( \frac{\phi_i(v_1^i) + \phi_i(v_2^i)}{2} \right). \tag{4.8}$$

**Proposition 4.5.** *For all $i, j$ and $v \in V$ such that $v \in c_i$ and $v \in c_j$ holds $\phi_i(v^i) = \phi_j(v^j)$.*

*Proof.* We proof that proposition by induction for all levels of refinement of $M$. When we start the charts $c_i$ and $c_j$ all edges that are in their boundary come directly of the base mesh, if $v \in c_i$ and $v \in c_j$ then $\phi_i(v^i) = \Pi_S(v) = \phi_j(v^j)$, by construction. Now suppose the Proposition 4.5 is true for $v$ with level less or equal the current. Observe that by (4.6) and (4.7) a boundary-vertex $v$ is created only when a boundary-edge is split, consequently by (4.8) and induction hypothesis holds:

$$\phi_i(v^i) = \Pi_S \left( \frac{\phi_i(v_1^i) + \phi_i(v_2^i)}{2} \right) = \Pi_S \left( \frac{\phi_j(v_1^j) + \phi_j(v_2^j)}{2} \right) = \phi_j(v^j).$$

□

To define the inverse of $\phi_i$ we use the projection $\Pi_M$, the idea is to project the point on the mesh, identify which face it laid and use the barycentric coordinates to define it coordinates in $\Omega_i$. More precisely, let $\Pi_M(p) = \sum_{k=1}^{3} \alpha_k v_k$, with $\alpha_k > 0$, $\sum_{k=1}^{3} \alpha_k = 1$ and $f = \{v_1, v_2, v_3\}$ where $L(f) = i$, then we have:

$$\phi_i^{-1}(p) = \sum_{k=1}^{3} \alpha_k v_j^i. \tag{4.9}$$

Since we are supposing that $M$ is close to $S$ we have $\phi$ and $\phi^{-1}$ well defined, i.e., $\phi_i \circ \phi_i^{-1}(p) = p$ and $\phi_i^{-1} \circ \phi_i(p^i) = p^i$ for all $p \in S \cap \phi_i(c_i)$ and $p^i \in c_i$.

To build and to glue the height-maps consistently we need to know how to write a inner-points of $c_i$ in $\Omega_j$ coordinates when $c_i$ and $c_j$ are neighbors, i.e., we need be able to write a point $p^i \in c_i$ in $\Omega_j$ coordinates when $c_i$ and $c_j$ have common vertices. Since we started our chart with quadrangle domains we use the approach develop by Stam [59] to convert $p^i$ to $p^j$. Stam [59] recover the relative affine coordinates of $\Omega_i$ to $\Omega_j$, he achieve that by matching commons edges of $c_i$ and $c_j$. Case $c_i$ and $c_j$ do not share edges we recursively propagate the coordinate information in the connected components by the neighbors.

### Sketching over the Surface

To allow the user add an augmentation we freeze the camera and she or he draws polygonal curves over the surface. These strokes are transported to atlas $\mathcal{A}$ where they are used to define height-maps, we name these projected curves as *height-curves*. To transport the curves to $\mathcal{A}$ we use the Equation (4.9) in their points, i.e. we project the curve points directly on $M$, identifying which face they was project, and use their barycentric coordinates to transport them to the correspondent $c_i$. If the line segment $pq$ starts in the chart $c_i$ and ends in the chart $c_j$ then to guarantee continuity we write $p^i q^i$ and find the point of this segment that is in the boundary of $c^i$ and add this point to the height-curve. We do the same thing to the segment $p^j q^j$. In Figure 4.8 we show the result of this process in two charts.

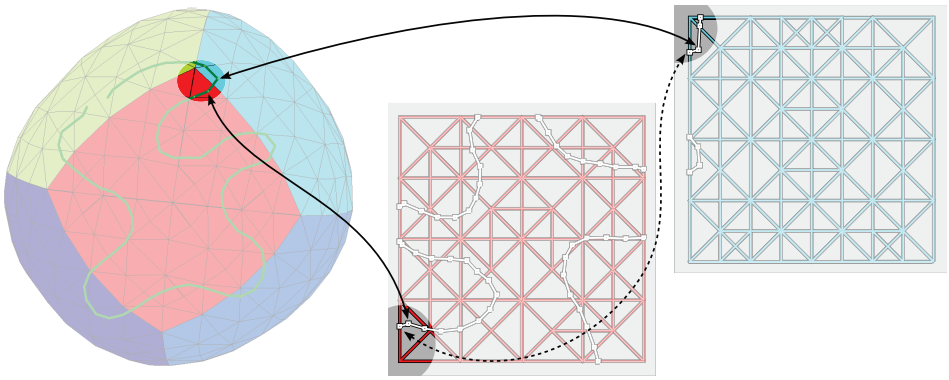


**Figure 4.8:** Sketch over surface and transported curve to $\mathcal{A}$. The two solid arrows show points on $M$ that are transported to $\mathcal{A}$, the dashed arrow shows points that are created in the chart boundaries to guarantee hight-curve continuity.

To define the height map we use the distance of a point in $c_i$ to the height-curve. We create this field using the approach presented by Frisken [25]. In this work she uses a vector distance field which represent the distance at any point as a vector value. That field is used for fitting on-the-fly an analytic curve to a sequence of digitized points. For simplicity we define a height $h$ for all points of one same height-curve $\lambda$, and we tested as height function $f_\lambda(p^i) = h\exp(-25d^4/4r^4)$ where $d$ is the distance of $p^i$ to $\lambda$ and $r$ is the radius of influence of $\lambda$.

After all, we have a height-map $h^i_u$ for each chart $c_i$ that was sketched by the user. Using the Equation (4.4) we can compose this height-map with another, such as a gray depth image $h^i_d$, for example to obtain a final height at $p \in M$ adding the heights, $h_p = h^i_d(p^i) + h^i_u(p^i)$. Then we have $D^*(p) = h_p N_p$ where $N_p$ is it normal at $p$. Thus we complete the formulation of the final surface: $\widetilde{\mathcal{S}} = \mathcal{S} + D^*(\mathcal{S})$ or specifically for all $p \in M$ we have $\tilde{p} = p + h_p N_p$.

### 4.2.4 4-8 Mesh

The 4-8 mesh $M$ has two main roles in our system, the first one was described in the last section, we use the mesh to transport points to the atlas. Besides that we use $M$ to visualize the approximate final surface. We use the library developed by Velho [65]. This library implements the 4-8 mesh with a good level of abstraction that means with little implementation effort we use this library as *black box* that provide a 4-8 mesh refined. We start the mesh with the base mesh triangulated, i.e., a face $f = \{v_1, v_2, v_3, v_4\}$ is split in two faces $f_1 = \{v_1, v_2, v_3\}$ and $f_2 = \{v_1, v_3, v_4\}$, and then we apply one refinement step to define the atlas this is the mesh level 0 (Figure 4.5(a)). In addition we need provide a function that samples a edge returning a new vertex, and two error functions. One to classify the edges for the refinement step and one to classify the vertices for simplification step.

To define a new vertex we adopt the naive approach that takes the middle point of an edge and project it on surface, i.e., to split a edge $e = \{v_1, v_2\}$, we create a new vertex $v_n = \Pi_S\left((v_1 + v_2)/2\right)$ and as described in the last section if $v_n \in c_i$ it saves its local coordinates too. In spite of this approach being simple it achieves good results for our application.

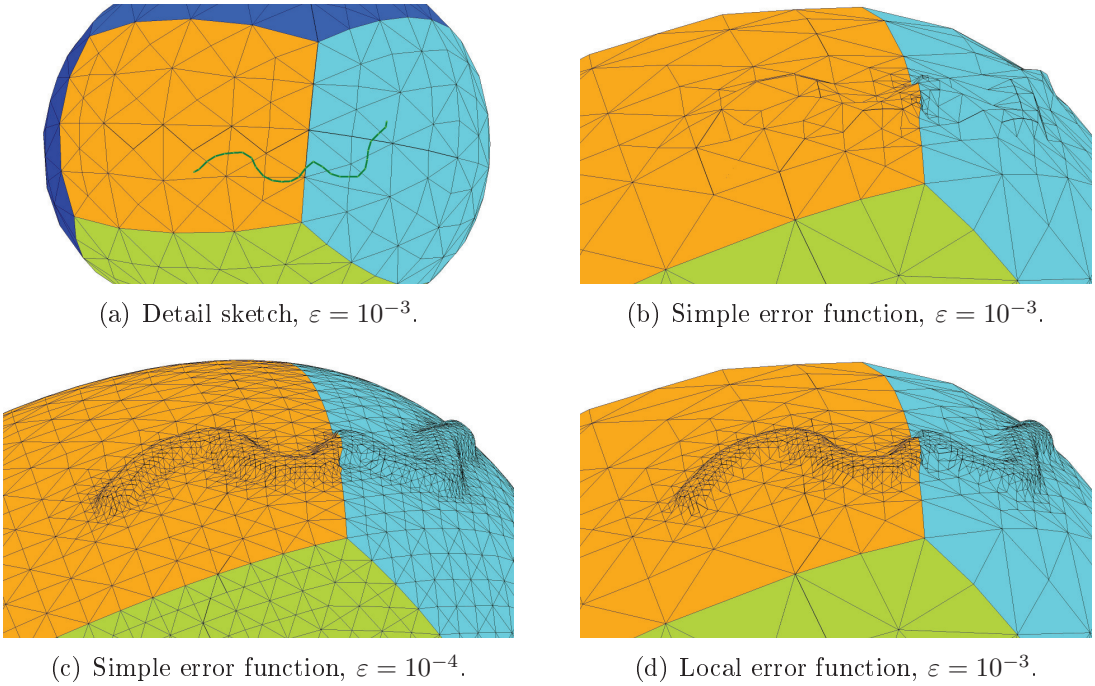
To complete the adaptive process of 4-8 mesh we need to choose which edges will be split, in order to refine the mesh and which vertices will be removed to simplify the mesh. In our implementation, this classification is done providing two error functions and one parameter. To define our error function we need to describe how we measure the distance between a point and the surface. First, observe that $\Pi_S$ is the projection on $S \neq \widetilde{S}$, thereupon the $\Pi_S$ is not enough to define the distance. To project a point $p$ on $\widetilde{S}$, first we project $p$ on $S$ then we apply $D$, more precisely,

$$\Pi_{\widetilde{S}}(p) = \Pi_S(p) \oplus D(\Pi_S(p)), \tag{4.10}$$

thus the distance between $p$ to $\widetilde{S}$ is the usual

$$d_{\widetilde{S}}(p) = |p - \Pi_{\widetilde{S}}(p)|. \tag{4.11}$$

Now we can determine the error functions using the stochastic approach presented by de Goes et al. [16]. Let us define the error on faces, we randomly take $n$ points on the face and calculate the distance of the point to the surface then we sum all distance and divide by $n$. Therefore the error function for edge is the error average of its faces, and the vertex error function is the error average of its star neighborhood. To control the mesh adaptation we define an error threshold $\varepsilon > 0$, if the edge error is above that threshold the edge is refined. Observe that, the $\varepsilon$ controls the size of our final mesh. If the $\varepsilon$ is small we have a good approximation of the surface though the mesh will have too many vertices which will be computationally expensive to execute simple operations such as project a line (Figure 4.9(c)). On the other hand, if the $\varepsilon$ is big the mesh will be computationally cheap however the mesh will not represent the final surface details (Figure 4.9(b)).



(a) Detail sketch, $\varepsilon = 10^{-3}$.

(b) Simple error function, $\varepsilon = 10^{-3}$.

(c) Simple error function, $\varepsilon = 10^{-4}$.

(d) Local error function, $\varepsilon = 10^{-3}$.

**Figure 4.9:** Local error control.

It is natural to have an approximation for $S$ geometry coarser than for $\widetilde{S}$ geometry because we are assuming that $S$ is only the coarse information in contrast to $\widetilde{S}$ that has also details (Figure 4.9(a) and (c)). However, since generally details are restricted to small surface areas if we use $\widetilde{S}$ to choose $\varepsilon$ we will have a expensive mesh that do not bring real benefits. Since our application works with two different levels of details so then is natural use that structure to define the error functions that depend on the detail level of a surface point. In fact, we are adding one more parameter in $\mathcal{P}'$ (4.2): the detail level $E(p)$. In our representation the details are encoded in $D$ however not all parameters of $\mathcal{S}$ will influence the final mesh, thus we introduce the notation $D_g$ for these parameters that affect the refinement. As a

result we define our level of detail at a point $p$ as

$$E(p) = \eta(D_g(p)), \tag{4.12}$$

where $\eta : \mathbb{R}^d \to \mathbb{R}_+$. We implement that using the height maps since they are our details over the surface, specifically the Equation (4.12) is rewritten as $E(p) = \max\{2|\nabla h_p|, 1\}$, where $\nabla$ is the gradient.

Now we have all elements to define an error function that is not blind to level of detail at a point over the surface. We define the local error function using Equation (4.11) and (4.12) so then we have $\Delta(p) = d_{\tilde{S}}(p)E(p)$. Now we apply this new definition in the face error calculation and as result we reformulate the edge error and the vertex error functions. In Figure 4.9 we can observe the difference between to use the simple error function and to use the local error function. The mesh in Figure 4.9(b) has 460 vertices however we lost the details of the final surface, if we decrease the $\varepsilon$ (Figure 4.9(c)) we reveal the details though the mesh grows ten times with 4.8k vertices, when we use the local error function (Figure 4.9(d)) we reveal the detail and the mesh size does not grow too much, 1.3k vertices.

### 4.2.5 Work-flow and Results

In this section we present all peaces of our pipeline working together. Our work-flows are based on the framework presented by de Goes et al. [16] to adapt dynamic meshes. There are three different work-flows in the pipeline: (1) the user starts the modeling system with a blank page or changes the actual model topology, (2) the geometry of the implicit surface changes, and (3) we need change the mesh resolution. The (3) usually happens when the height-maps are changed. The overview of the work-flow is depicted in figure 4.10.

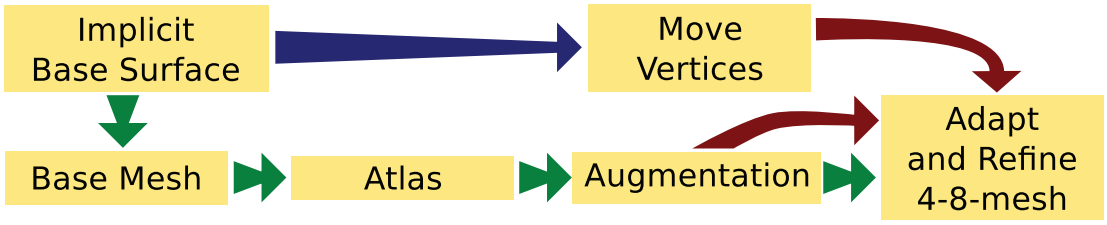



**Figure 4.10:** Overview of system work-flows: green arrows are the startup and topological change step sequence, blue arrow are stepped when the implicit surface is edited, and the red arrow is done when the mesh resolution changes.

The user starts the model with construct lines, using the system described in Chapter 3, these lines create samples that define an implicit surface (Figure 4.11(a)). After that the user creates a planar version of the base mesh that approximates the geometry and has the same topology of the final model (Figure 4.11(b)). Thus, the base mesh is transported to space (Figure 4.11(c)). Now the base mesh is used to creates an atlas structure (Figure 4.11(d)) for a 4-8 mesh. This mesh is adapted and refined creating the first approximation of the final model (Figure 4.11(e)). These steps described up to now are the common steps for all modeling sessions. They

are represented by the green arrows in Figure 4.10. In addition, these steps are illustrated in Figures 4.12(a) and (b), 4.13(a), and 4.14(a). Note that when we change the topology we need change the base mesh, then the process starts again, in particular in Figure 4.12(a) and (b). If there is a predefined height-map, the model ends this stage with one or more layer of details. Specifically in Figure 4.14(a) we start the model with a height-map encoded as a gray image.

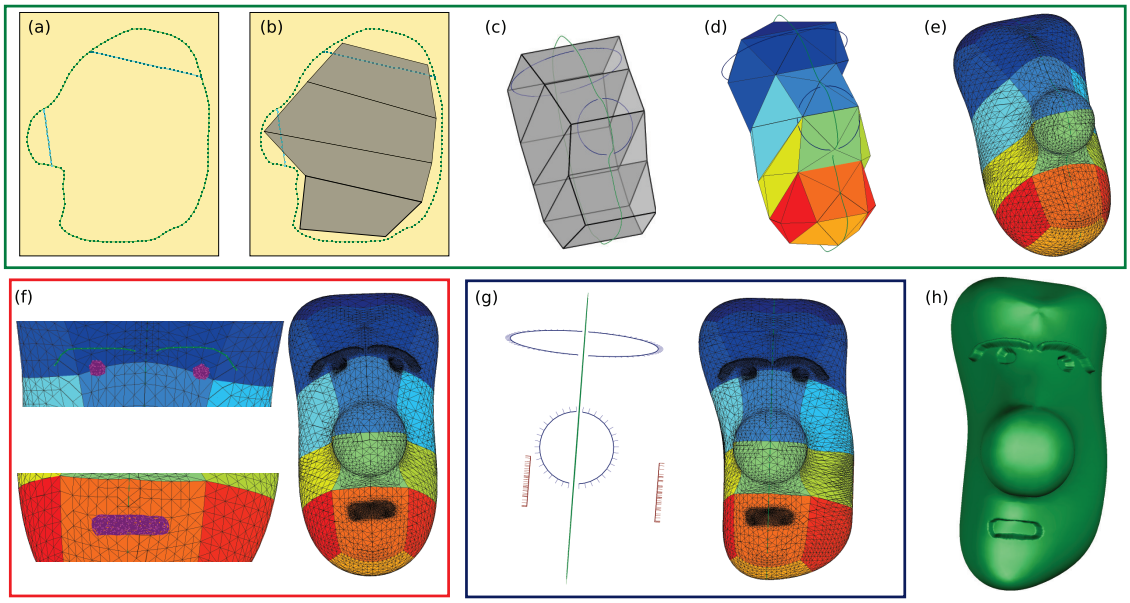



**Figure 4.11:** Steps to model a head.

After the first approximation for the final surface the user can edit the implicit surface and create/edit a height-map. When details are added on the surface imply in almost all cases that the resolution of the mesh is not fine enough to represent the new augmentation, then we ought adapt and refine the mesh. In Figures 4.11(f), 4.12(c), 4.13(b), and 4.14(b): the user sketches a height map over the surface and the mesh is refined to represent the geometry of the augmentation correctly. At any stage the user can change the implicit surface and if the topology continues the same to obtain a fast approximation the system allows to move the vertices without adapt and refine. Since detail are codified separately when we move the vertices the details are moved consistently. We illustrate that in Figures 4.11(g), 4.14(c), and 4.13(c), (e) and (f). Specifically in Figure 4.13(e) and (f) we can compare the good final result preserving the details despite the big change of the implicit surface. Some times, when only the implicit surface is changed, to move the vertices is not enough to reach the quality desired then the user can adapt and refine the mesh decreasing the error threshold, as shown in Figure 4.13(d), where the user starts the $\varepsilon = 10^{-3}$ and after some modeling steps the user chooses a new threshold $10^{-4}$.

The four models presented in this section took around 20 minutes each to be modeled, we are considering the time that starting from the blank page up to the final mesh generation. All the results were generated on an 2.66 GHz Intel Xeon W3520, 12 gigabyte of RAM and OpenGL/nVIDIA GForce GTX 470 graphics. The

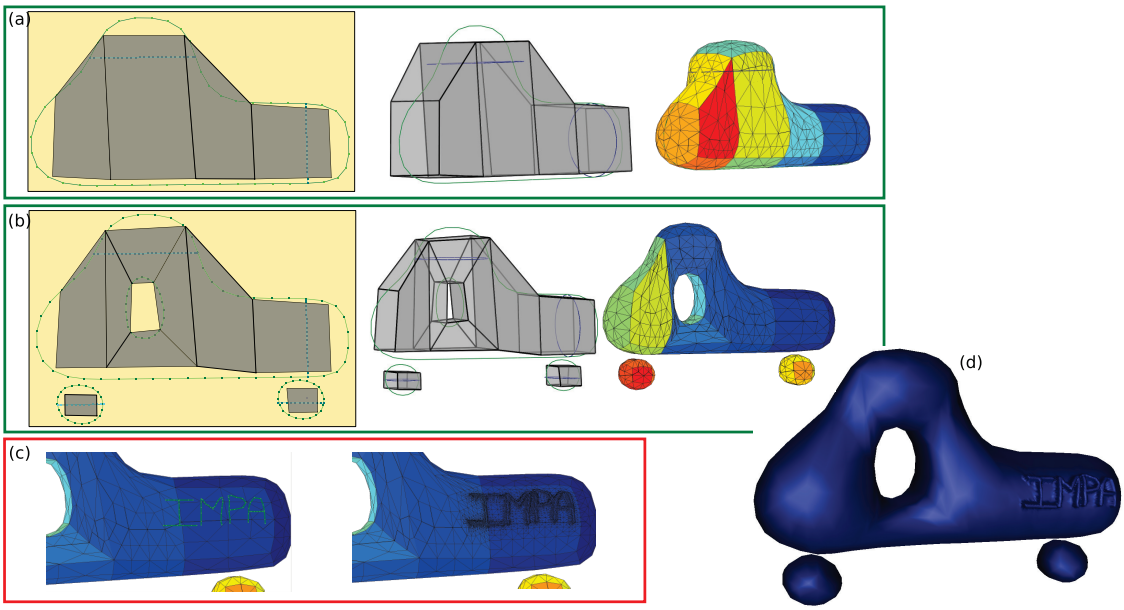



**Figure 4.12:** Steps to model a space car.

most expensive step was to create the implicit surface, after that to create the base mesh, to augment, and small adjusts in the implicit surface do not take too much time. The system bottle neck is the mesh update if the mesh has too many vertices (about 10k) a step of refinement after an augmentation takes about 10 seconds, for example the Head models has 11k vertices and to adapt the mesh after an augmentation takes about 10 seconds. The final models of space car, terrain, and party balloon have 14k, 11k, and 13k vertices respectively.

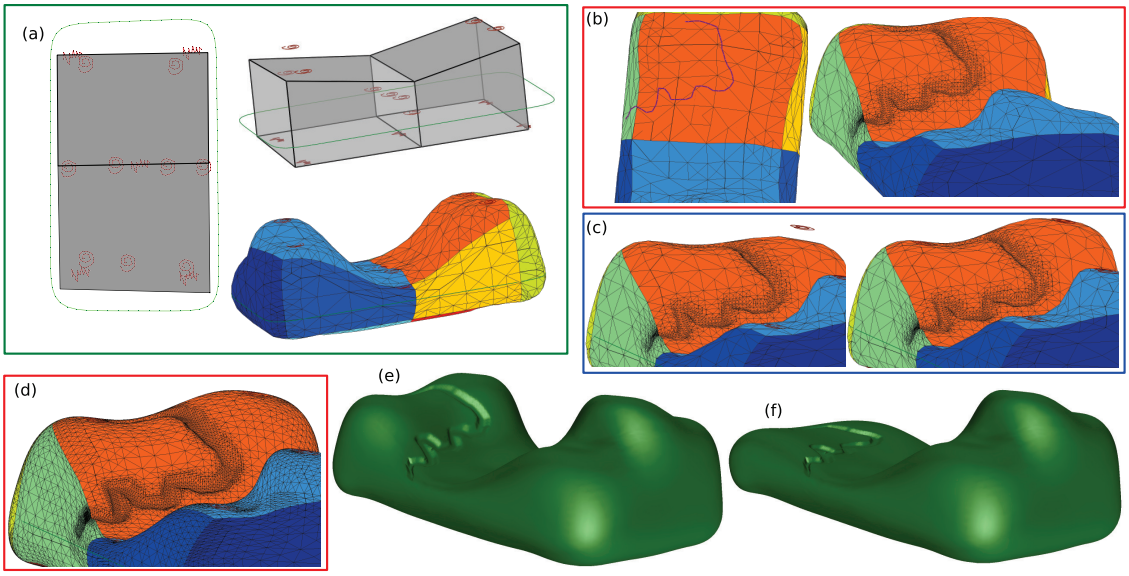



**Figure 4.13:** Steps to model a terrain.

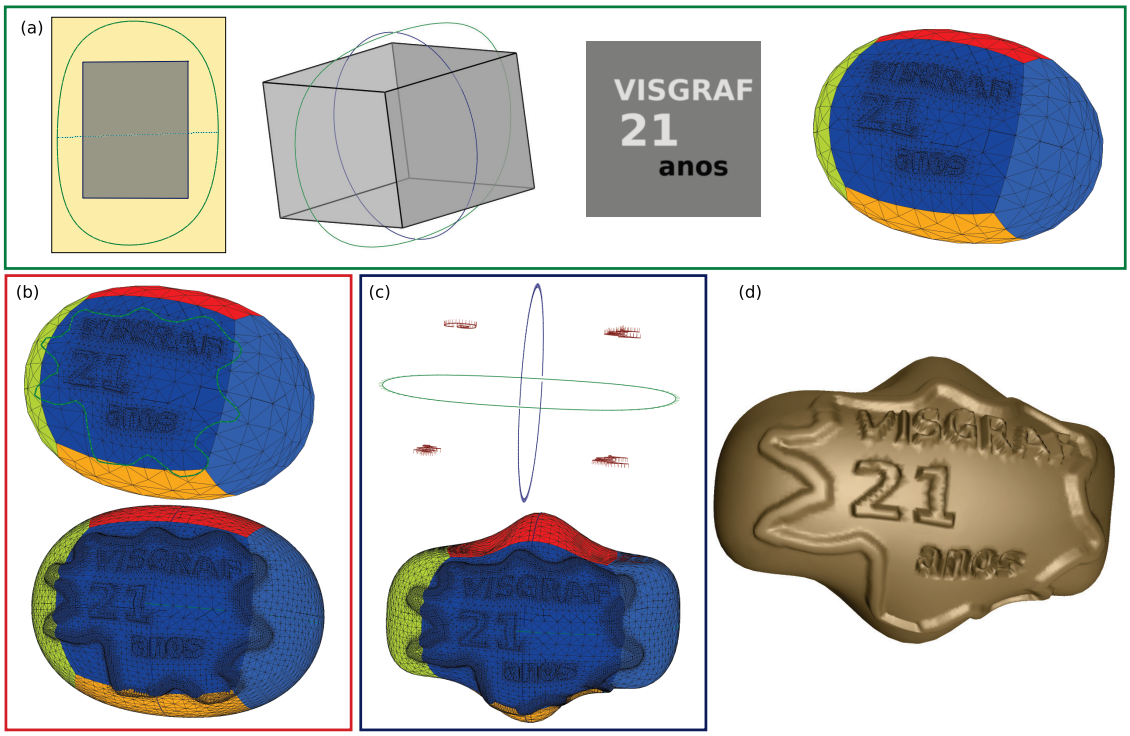



**Figure 4.14:** Steps to model a party balloon.

## 4.3 Conclusion and Future Work

In this chapter we discussed the problem of what is a good surface representation for sketch-based modeling. We focused in developing a representation that allows us to control the model edition in two scales: global and local. Therefore we proposed a general mathematical representation for surfaces tailored to SBM problem. The main idea of this abstract representation is to split the object between base and detail using the semantics of function composition and binary operations. In addition we developed a Sketch-based Surface Modeling system using a pipeline based on the theory developed for the mathematical representation. The pipeline proposed has four main elements: implicit surface, base mesh, atlas and 4-8 mesh. For each element there are several issues and possible approaches. Since we desired to investigate whether that representation is powerful enough to build the SBM system, for each pipeline element we give a off-the-shelf solution or we created simple, and some times naive, approach for each element problems. In spite of these simple approaches, our system allows us to model different shapes controlling the local and global changes. The advantages of these simple approaches are: we are sure that representation is doable and powerful for SBIM problem and we have found avenues for further research.

Our chose for base surface was to use ours previous work about implicit although this modeling system be very powerful we would like to test our abstract formulation in others system with others smooth surfaces. As example the *Matisse* system (Bernhardt et al. [7]) that uses a convolution surface or *Fibermesh* (Nealen et al.

[48]) that creates a mesh.

One important example of element that demand more research is the base mesh. We implement a semi-automatic approach that the user place the vertices to approximate the geometry and topology, after that the base mesh is created in the space. this approach achieves good results however we only can work in one plain. Since the base mesh is the responsible for the topology of the final model, we are restricted in models that topology can be handle in one plain. Thereupon we plan research two approaches for the base mesh problem. The first one is to transport the actual semi-automatic solution to 3D, in this case the user handles box directly in the space, the main challenge is to develop an interface that the user do not waste too much time and effort creating the base mesh. The other approach is use a mesh simplification as example the method presented by Daniels et al. [15], although this approach be automatic it starts with a dense mesh, then we are changing the problem of how to find a base mesh for the problem of create a mesh with the same topology.

We developed a theory to construct atlas which is responsible to control the local edition of the model. The label theory developed gives a constructive algorithm with guarantees to create a partition over the set of faces of a stellar adaptive mesh. We use this partition to build the atlas structure. However there are much more to be done in this problem, we aim to develop: tools (mathematical and computational) to handle with the scale of the atlas, an interface to control predefined height-maps, and algorithms that split the atlas if it has a high level of deformation in relation of the surface.

Although we use the 4-8 mesh library as a black box, there is some work to be done about the error function. In our current implementation the final mesh quality depends on the base meshes. If we have a base mesh in which each face looks like a square and approximates well the local geometry, we will have good triangles. However, if some initial base-mesh faces are not squares we may have very skinny triangles. We plan to research methods to improve the mesh quality independently of the base mesh. To achieve that we have two approaches: the first one is using filters for mesh smoothing, the other one is to improve our error function.

Finally we want to apply this pipeline for SBM system in specific domains. We believe that the potential of our representation and pipeline will be better exploited if we apply those in a specific domain, e.g., figure modeling or geological modeling.

# Chapter 5

# Final Remarks

We presented a study on sketch as input for modeling systems (i.e. sketch-based modeling, or SBM), focusing on mathematical representations for sketch-based modeling system. We advocated that the representation of the model plays a fundamental role in this problem, requiring the underlying representations to be specially tailored for use in SBM application and commons requirements can be abstracted, to guide the definition of a specific representation for a specific domain. In Chapters 2 and 3 the sketch tools were developed to satisfy the restriction of specific representations: Hermit–Birkhoff Radial Basis Functions (HBRBF) and Hermit Radial Basis Functions (HRBF), respectively. As it was observed in those chapters these representations are powerful and well suitable for the problem of sketch-based modeling. However, they do not provide any good answer to one of the basic questions in SBM: "Which is the best representation for modeling using sketches for a specific domain?". We are looking for a more general answer, a more abstract mathematical structure that gives us a starting point whenever we are seeking for a concrete representation for a sketch-based modeling pipeline to be applied for a specific domain. In Chapter 4 we started to answer this question. We chose some common desired properties for sketch-based modeling and based on that we created an abstract mathematical framework to define representations to SBM systems. However, there is still much work to be done.

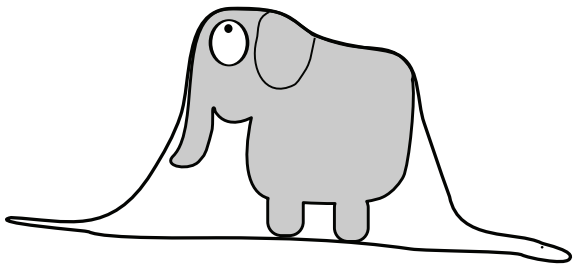
In Chapter 2 we developed sketch tools for modeling warping fields. These fields were successfully applied to deform RGBN images. This problem demands specific properties on the sketch tools and representation. We had to develop tools to create $C^1$ fields with boundary restrictions. Now we would like to study the same problem using the pipeline proposed in Chapter 4 in addition, we plan to apply this techniques to create animations.

In Chapter 3 we approached the problem of modeling implicit surfaces. The modeler operators created in this chapter reach good results despite using only one drawing plane to be defined. One directly extension is use more drawing planes to allow more complex shapes, by the same token, we would like to create tools to design and edit curves directly in 3D. Regarding the representation we plan to implement the HBRBF tools in our system. One drawback in our system was the restriction that all samples have to Hermite samples. Thus, the HBRBF will yield

a more flexible and intuitive way to define the samples.

In Chapter 4 we proposed a novel approach to the problem of SBM, we developed an abstract mathematical representation to be suitable for SBM in specific domains. We focused on this abstract representation to give a concrete pipeline that allows to control levels of detail. We implemented this pipeline using the implicit modeler developed in Chapter 3 as base surface and we created tools to parametrize the surface and edit details. This pipeline was split in four parts: implicit surface, base mesh, atlas and 4-8 mesh. In all these elements there are still many avenues to be exploit. Finally, the next main step is to apply that abstract representation to a specific domain.

In conclusion, the mathematical representation of the model plays a central role in the problem of using sketches to create and edit the model itself. We use the experience acquired in the earlier developed SBM applications to abstract and propose a novel representation applying that to create a SBM system with more control.



# Bibliography


[1] LAPACK – Linear Algebra PACKage, Last visit in August 2010. http://netlib.org/lapack/.

[2] A. Alexe, V. Gaildrat, and L. Barthe. Interactive modelling from sketches using spherical implicit functions. In *Proc. of AFRIGRAPH '04*, pages 25–34, 2004.

[3] W. Andrews. Introduction to perceptual principles in medical illustration: Lines & illusions. In *Tutorial "Illustrative Visualization for Medicine and Science", Eurographics '06*, 2006.

[4] B. Araújo and J. A. Jorge. Blobmaker: Freeform modelling with variational implicit surfaces. In *Proc. of 12th Enc. Português de Computação Gráfica*, pages 17–26, 2003.

[5] A. Barbier, E. Galin, and S. Akkouche. Complex skeletal implicit surfaces with levels of detail. In *Proc. of 12th Intl. Conf. in Central Europe on Computer Graphics, Vis. and Computer Vision (WSCG '04)*, pages 35–42, 2004.

[6] L. Barthe, B. Mora, N. Dodgson, and M. Sabin. Interactive implicit modelling based on $c1$ reconstruction of regular grids. *Intl. Journal of Shape Modeling*, 8 (2):99–117, 2002.

[7] A. Bernhardt, A. Pihuit, M.-P. Cani, and L. Barthe. Matisse: Painting 2D regions for modeling free-form shapes. In *SBIM '08: 5th Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 57–64, 2008.

[8] J. F. Blinn. Simulation of wrinkled surfaces. pages 286–292, 1978.

[9] J. Bloomenthal. An implicit surface polygonizer. In *Graphics gems IV*, pages 324–349, San Diego, CA, USA, 1994. Academic Press Professional, Inc.

[10] J. Bloomenthal and K. Shoemake. Convolution surfaces. pages 251–256, 1991.

[11] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. In *Proc. of 1990 Symposium on Interactive 3D Graphics (I3D '90)*, pages 109–116, 1990.

[12] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects

with radial basis functions. In *Proc. of SIGGRAPH '01*, pages 67–76. ACM, 2001.

[13] J. J. Cherlin, F. Samavati, M. Costa Sousa, and J. A. Jorge. Sketch-based modeling with few strokes. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, pages 137–145, New York, NY, USA, 2005. ACM.

[14] F. Cordier and H. Seo. Free-form sketching of self-occluding objects. *IEEE Comput. Graph. Appl.*, 27(1):50–59, 2007.

[15] J. Daniels, C. T. Silva, J. Shepherd, and E. Cohen. Quadrilateral mesh simplification. *ACM Trans. Graph.*, 27:148:1–148:9, December 2008.

[16] F. de Goes, S. Goldenstein, and L. Velho. A simple and flexible framework to adapt dynamic meshes. *Computers & Graphics*, 32(2):141–148, 2008.

[17] A. de Saint-Exupéry. *The little prince.* Richard Howard, 2000.

[18] M. Desbrun, N. Tsingos, and M.-P. Gascuel. Adaptive sampling of implicit surfaces for interactive modelling and animation. *Computer Graphics Forum*, 15(5):319–325, 1996.

[19] M. P. do Carmo. *Differential geometry of curves and surfaces.* Prentice-Hall Inc., Englewood Cliffs, N. J., 1976.

[20] J. Duchon. *Splines minimizing rotation-invariant semi-norms in Sobolev spaces*, volume 571 of *Lecture Notes in Mathematics*, pages 85–100. Springer Berlin Heidelberg, 1977.

[21] M. Eitz, O. Sorkine, and M. Alexa. Sketch based image deformation. In *Proceedings of Vision, Modeling and Visualization (VMV)*, pages 135–142, 2007.

[22] H. Fang and J. C. Hart. Detail preserving shape deformation in image editing. *ACM Trans. Graph.*, 26, July 2007.

[23] G. Fasshauer. Solving partial differential equations by collocation with radial basis functions. In *Surface Fitting and Multiresolution Methods*, pages 131–138. Vanderbilt University Press, 1997.

[24] E. Ferley, M.-P. Cani, and J.-D. Gascuel. Practical volumetric sculpting. *Visual Computer*, 16(8):469–480, 2000.

[25] S. F. Frisken. Efficient curve fitting. *Journal of graphics, gpu, and game tools*, 13(2):37–54, 2008.

[26] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2D-to-3D modeling. *ACM Trans. Graph.*, 28(5):148, 2009.

[27] J. C. Hart, E. Bachta, W. Jarosz, and T. Fleury. Using particles to sample and control more complex implicit surfaces. In *Proc. of Shape Modeling Intl. (SMI'02)*, pages 129–136, 2002.

[28] A. Hornung, E. Dekkers, and L. Kobbelt. Character animation from 2D pictures and 3D motion data. *ACM Trans. Graph.*, 26(1):1, 2007.

[29] T. Igarashi and J. F. Hughes. Smooth meshes for sketch-based freeform modeling. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 139–142, New York, NY, USA, 2003. ACM.

[30] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *Proc. of SIGGRAPH '99*, pages 409–416. ACM, 1999.

[31] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[32] S. F. Johnston. Lumo: illumination for cel animation. In *NPAR '02: 2nd Intl. symposium on Non-photorealistic animation and rendering*, pages 45–52, 2002.

[33] J. A. Jorge, N. F. Silva, and T. D. Cardoso. Gides++. In *Proc. of 12th Encontro Português de Computação Gráfica*, 2003. http://vimmi.inesc-id.pt/~tfdc/gides++/index.html.

[34] L. B. Kara and K. Shimada. Sketch-based 3D-shape creation for industrial styling design. *IEEE Comput. Graph. Appl.*, 27(1):60–71, 2007.

[35] L. B. Kara and K. Shimada. Supporting early styling design of automobiles using sketch-based 3D shape construction. *Computer-Aided Design and Applications*, 5(6):867–876, 2008.

[36] O. Karpenko and J. F. Hughes. Smoothsketch: 3D free-form shapes from complex sketches. *ACM Trans. Graph.*, 25(3):589–598, 2006.

[37] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21:585–594, 2002.

[38] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. of SIGGRAPH '96*, pages 313–324. ACM, 1996.

[39] E. Lane. *Karst in Florida. Florida Geological Survey*. Florida Bureau of Geology, Tallahassee, 1986. Special Report no.29. http://www.dep.state.fl.us/geology/.

[40] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proc. of SIGGRAPH '00*, pages 85–94. ACM, 2000.

[41] D. Levin. The approximation power of moving least-squares. *Mathematics of computation*, 67(224):1517–1531, 1998.

[42] W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Proceedings of the Kirbyfest (Berkeley, CA, 1998)*, volume 2 of *Geom. Topol. Monogr.*, pages 299–320 (electronic). Geom. Topol. Publ., Coventry, 1999.

[43] I. Macêdo, J. P. Gois, and L. Velho. Hermite interpolation of implicit surfaces with radial basis functions. In *Sibgrapi 2009 (XXII Brazilian Symposium on Computer Graphics and Image Processing)*, Rio de Janeiro, RJ, october 2009. IEEE.

[44] I. Macêdo, J. P. Gois, and L. Velho. Hermite radial basis functions implicits. *Computer Graphics Forum*, 30(1):27–42, 2011.

[45] L. Markosian, J. M. Cohen, T. Crulli, and J. F. Hughes. Skin: A constructive approach to modeling free-form shapes. In *Proc. of SIGGRAPH '99*, pages 393–400, 1999.

[46] K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr. Level set surface editing operators. *ACM Trans. Graph.*, 21(3):330–338, 2002.

[47] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147, 2005.

[48] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph.*, 26(3):41–50, 2007.

[49] L. Olsen, F. F. Samavati, M. Costa Sousa, and J. Jorge. Sketch-based modeling: a survey. *Computer & Graphics*, 33(1):85–103, 2009.

[50] S. Owada, F. Nielsen, K. Nakazawa, and T. Igarashi. A sketching interface for modeling the internal structures of 3D shapes. In *Proc. of 4th International Symposium on Smart Graphics*, pages 49–57, 2003.

[51] T. Pereira. Normalshop: Modeling surface mesostructure. Master's thesis, IMPA – Instituto Nacional de Matematemática Pura e Aplicada, 2010. http://www.visgraf.impa.br/Data/RefBib/PS_PDF/student-msc-2010-03-thiago-pereira/tspereira-ms-thesis.pdf.

[52] T. Pereira, E. Vital Brazil, I. Macêdo, M. Costa Sousa, L. H. de Figueiredo, and L. Velho. Sketch-based warping of RGBN images. *Graph. Models*, 73(4): 97–110, 2011.

[53] R. N. Perry and S. F. Frisken. Kizamu: A system for sculpting digital characters. In *Proc. of SIGGRAPH '01*, pages 47–56, 2001.

[54] A. Ryder. *The Artist's Complete Guide to Figure Drawing: A Contemporary Perspective on the Classical Tradition*. Watson-Guptill, New York, 1999.

[55] F. Samavati and R. Bartels. Local filters of b-spline wavelets. In *Proceedings of International Workshop on Biometric Technologies 2004*, pages 105–110, 2004.

[56] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25(3):533–540, 2006.

[57] R. Schmidt, B. Wyvill, M. Costa Sousa, and J. A. Jorge. ShapeShop: Sketch-based solid modeling with blobtrees. In *SBIM '05: 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 53–62, 2005.

[58] T. W. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2(1–3):53–59, 1985.

[59] J. Stam. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.*, 22: 724–731, July 2003.

[60] B. T. Stander and J. C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *SIGGRAPH 2005 Courses*. ACM, 2005.

[61] C.-L. Tai, H. Zhang, and J. C.-K. Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23(1): 71–83, 2004.

[62] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *Proc. of SIGGRAPH '99*, pages 335–342, 1999.

[63] L. Velho. Simple and efficient polygonization of implicit surfaces. *Journal of graphics, gpu, and game tools*, 1(2):5–24, February 1996.

[64] L. Velho. Stellar subdivision grammars. In *Proc. of Eurographics/ACM SIGGRAPH symp. on Geom. proc.*, SGP'03, pages 188–199. Eurographics Association, 2003.

[65] L. Velho. A dynamic adaptive mesh library based on stellar operators. *Journal of graphics, gpu, and game tools*, 9(2):21–47, 2004.

[66] L. Velho and J. Gomes. Approximate conversion of parametric to implicit surfaces. In *Computer Graphics Forum*, volume 15, pages 327–337, 1996.

[67] E. Vital Brazil, I. Macêdo, M. Costa Sousa, L. H. de Figueiredo, and L. Velho. A few good samples: Shape & tone depiction for Hermite RBF implicits. In *NPAR '10: 8th Intl. Symposium on Non-Photorealistic Animation and Rendering*, pages 1–8, 2010.

[68] E. Vital Brazil, I. Macêdo, M. Costa Sousa, L. H. de Figueiredo, and L. Velho. Sketching variational Hermite-RBF implicits. In *SBIM '10: 7th Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 1–8, 2010.

[69] E. Vital Brazil, I. Macêdo, M. Costa Sousa, L. Velho, and L. H. de Figueiredo. Shape and tone depiction for implicit surfaces. *Computers & Graphics*, 35(1): 43–53, 2011.

[70] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2005.

[71] Y. Weng, X. Shi, H. Bao, and J. Zhang. Sketching MLS image deformations on the GPU. *Computer Graphics Forum*, 27(7):1789–1796, 2008.

[72] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. of SIGGRAPH '94*, pages 269–278, 1994.

[73] E. S. Woody. *Pottery on the Wheel*. Allworth Press, 2008. http://books.google.com/books?id=F2XhzdK3ZEsC.

[74] B. Wyvill, A. Guy, and E. Galin. Extending the CSG tree-warping, blending, and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, 1999.

[75] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: an interface for sketching 3D scenes. In *Proc. of SIGGRAPH '96*, pages 163–170. ACM, 1996.

[76] J. Zimmermann, A. Nealen, and M. Alexa. Silsketch: automated sketch-based editing of surface meshes. In *SBIM '07: 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 23–30. ACM, 2007.