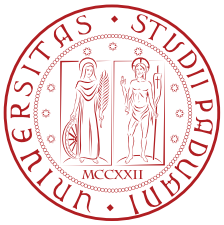


Universitá degli studi di Padova
Dipartimento di Matematica

SCUOLA DI DOTTORATO DI RICERCA IN SCIENZE MATEMATICHE
Indirizzo Matematica Computazionale - XXV Ciclo

Geometric Surface Processing and Virtual Modeling

Dottorando:
Marco Rucci

Supervisore:
Prof.ssa Serena Morigi

Direttore della Scuola:
Prof. Paolo Dai Pra

Table of Contents

Introduction - English

In this work we focus on two main topics "Geometric Surface Processing" and "Virtual Modeling". The inspiration and coordination for most of the research work contained in the thesis has been driven by the project New Interactive and Innovative Technologies for CAD (NIIT4CAD), funded by the European Eurostars Programme. NIIT4CAD has the ambitious aim of overcoming the limitations of the traditional approach to surface modeling of current 3D CAD systems by introducing new methodologies and technologies based on subdivision surfaces in a new virtual modeling framework. These innovations will allow designers and engineers to transform quickly and intuitively an idea of shape in a high-quality geometrical model suited for engineering and manufacturing purposes. One of the objective of the thesis is indeed the reconstruction and modeling of surfaces, representing arbitrary topology objects, starting from 3D irregular curve networks acquired through an ad-hoc smart-pen device.

The thesis is organized in two main parts: "Geometric Surface Processing" and "Virtual Modeling". During the development of the geometric pipeline in our Virtual Modeling system, we faced many challenges that captured our interest and opened new areas of research and experimentation. In the first part, we present these theories and some applications to Geometric Surface Processing. This allowed us to better formalize and give a broader understanding on some of the techniques used in our latest advancements on virtual modeling and surface reconstruction.

The research on both topics led to important results, briefly summarized in this thesis introduction, that have been published and presented in articles and conferences of international relevance.

Part 1: Geometric Surface Processing

The application of mathematical models based on Partial Differential Equations (PDE) to image processing and computer graphics problems has been extremely successful over the past 20 years. In particular, geometric surface flows have been extensively used in mesh processing. While a large part of the image processing community solve the PDE models using an Eulerian methodology (typically, with level sets), Lagrangian representations of surfaces based on triangle meshes are most common in graphics. In this Lagrangian setting, discretization of continuous flows is usually achieved through the use of discrete differential operators or using finite element techniques. In this thesis, we follow the former approach and we propose differential models on evolving manifold and numerical solutions to surface processing problems such as reconstruction, smoothing, remeshing, simplification and deformation.

Let $\mathcal{M}_0 = Image(X_0) := \{X_0(u), u \in [0,1] \times [0,1]\}$ be a compact, closed immersed orientable surface in $\mathbb{R}^3$ and X_0 be the corresponding parameter map. A geometric surface evolution consists of finding a family $\mathcal{M}(t) = Image(X(\cdot,t)), t \in [0,T), T > 0$ of smooth, closed, immersed orientable surfaces in $\mathbb{R}^3$ which evolve according to the flow equation (geometric flow)

$$\frac{\partial X}{\partial t} = -\beta \overrightarrow{N} + \alpha \overrightarrow{T},$$

where $\overrightarrow{N}$ is the unit normal vector to the surface, β is a velocity applied along the normal direction and α is the velocity in the tangent direction $\overrightarrow{T}$.

The family of manifolds $\mathcal{M}(t) \in \mathbb{R}^4$ moves along the normal direction driven by a normal velocity β which may be a function, for example, of the curvature and spatial position. The normal motion controls the geometry of the surface while the role of the tangential velocity is a sort of redistribution of the nodes which improves the accuracy of the surface representation.

In order to numerically approximate the PDEs on the evolving surface $\mathcal{M}(t)$, we define a discrete setting. The spatial approximation of $\mathcal{M}(t)$ is an evolving interpolated polyhedral mesh consisting of a union of faces whose vertices $X(t)$ lie on $\mathcal{M}(t)$, and $X(t)$ represents the parameteri-

zation of the surface itself. We can define, discretize and approximate differential evolutive PDE models based on local operators such as the Laplace-Beltrami, the intrinsic gradient and divergence.

We consider both discrete geometric flows, i.e., flows based on discrete analogous of continuous differential geometry quantities, see for example [69, 94], and variational methods. Different approaches are based on the classical discretization of continuous models by finite volume and finite elements schemes. For example, in [25] and [6] finite element approaches are considered.

In this thesis, numerical solutions to the **fairing** or smoothness problem are presented. Such problem is formulated in terms of variational or energy based models in order to derive a nonlocal approach that performs smoothing by evolving the surface according to a fourth order Non Local Surface Diffusion Flow (NL-SDF) on $\mathcal{M}$. Results are summarized in [78].

Remeshing refers to the redistribution of the sampling and connectivity of the geometry in order to satisfy mesh property requirements while maintaining surface features. In this thesis, we present an adaptive remeshing method that uses the mean curvature as an intrinsic measure of regularity. We propose an adaptive remeshing method which consists of a two steps PDE model where in the first step the vertex area function $A(X)$ defined on M is diffused over the mesh, influenced by the mean curvature map. In the second step the vertex position is tangentially relocated to obtain edges on element stars approximately of the same size. Results are reported in [75].

Mesh **simplification** is the process of reducing mesh complexity while preserving the topology and a good approximation to the original geometry. Mesh simplification is a fundamental step in common 3D acquisition system and in multiresolution deformation. In this thesis we focus on incremental algorithms and present a new approach to multilevel surface mesh simplification based on the evolution of surfaces under p-Laplacian flow. The p-Laplacian flow is used to clusterize vertices toward region of high curvature while an incremental decimation process orderly removes the shorter and less feature-representative edges from the polygonal mesh. Results have been presented in [77].

Interactive shape **deformation** allows intuitive manipulation of a 3D ob-

ject global and local aspect. The Reverse Engineering and Virtual Modeling system (described in Part 2 of the thesis), allows for interactive shape manipulation through energy-based deformation methods. The main requirement for physically-based surface deformation is an elastic energy that measures how much an object has been deformed from its initial configuration. We present a new proposal for global and local shape deformation methods based on the minimization of the Total Curvature energy of a surface.

Combining the developed approach with our multiresolution simplification method we can provide a complete multiresolution deformation framework for surface meshes.

Part 1 is organized as follows. After a brief introduction PDE models on surfaces in Chapter 1, we introduce the notation and the geometric setting used Section 1.1. In Section 1.2 and 1.3 we describe the intrinsic geometric operators. Then we discuss energy-based continuous geometric flows (Section 1.4) and discrete geometric flows (Section 1.5). In Chapter 2 we describe in details our advances and results in the solution of common problems in the field of Computer Graphics and Computer Vision through variational methods. In particular, smoothing, remeshing, deformation and simplification will be the subjects of Sections 2.1, 2.2, 2.3 and 2.4, respectively.

Part 2: Virtual Modeling

In the second part we introduce and discuss in details a new method of reverse engineering for acquisition and reconstruction of a virtual 3D model representing an existing physical object. The proposed Fast Interactive Reverse Engineering System (FIRES)[9] exploits a pen-based active stereo acquisition system supported by a reconstruction and visualization layer based on subdivision surfaces. By simply dragging a smart-pen device in space, the user draws and refines arbitrary 3D style-curves that define an outline of the desired shape. The 3D curve sketching process is achieved by an active stereo vision system made of two infrared cameras and a smart-pen device. The curve sketching process is integrated with

the surface reconstruction step into an iterative and incremental process that allows the user to have a real-time visual feedback on the ongoing work. The process of interactively and incrementally drawing the irregular curve network is called Interactive Surface Sketching (ISS).The raw data gathered both from the smart-pen and from the cameras is processed in an acquisition pipeline where the projected led image points of both cameras are first matched and then triangulated.Then, after identification, the 3D points are used to estimate the position of the pen-tip and the sequence of 3D pen-tip positions traces a curve that is shown to the user after a real-time smoothing and sampling step.

The ISS produces a curve network which is represented as a polyline mesh that is, a mesh with faces, vertices and edges augmented with polylines associated to each edge. Irregular curve networks can lead to associated polyline meshes with n-sided, non-planar, and non-convex faces.

During the development of FIRES we strived for the design of a flexible and robust tool for reconstructing surfaces from irregular and noisy curve networks. Our first approach consisted in a multi step process involving bilinearly blended Coons patches that interpolated points on given curves. More recently, further studies in geometric surface processing allowed us to introduce an alternative reconstruction approach based on a surface diffusion flow. This latter approach greatly improved the reliability of the reconstruction system and opened new interesting area of research and experimentation. Results have been presented in [76].

The reconstruction method is based on a preliminary step where we construct a sufficiently refined mesh X_0 by a flat tessellation (grid) of each polygon in the base mesh. Then we solve a global optimization problem by applying directly to the coordinate maps X a curvature based fourth order flow. The proposed surfaced diffusion flow is able to reconstruct a smooth surface, but can also reproduce the sharp feature of the object (creases, corners and edges).

Part 2 is organized as follows. Section 3.1 presents some related works and Section 3.2 is an overview on FIRES. The data structure layer of FIRES is presented in Section 3.3. Then, following the main stages of the reverse engineering pipeline, we describe in Section 4.1 the acquisition phase, in Section 5 the interactive surface sketching methodology, and in Sec-

tion 6 the surface reconstruction method adopted in FIRES. Examples in Section 7 illustrate the performance of the reconstruction method when applied to a few object reconstructions.

Summary of key contributions.

- A novel smoothing method based on a two-step algorithm that solves a nonlocal surface diffusion flow PDE. The method is able to remove spurious oscillations while preserving sharp features and is discussed in Section 2.1.5.
- A new adaptive remeshing scheme based on the idea of improving mesh quality by a series of local modifications of the mesh geometry and connectivity. We contribute to the family of parametrization-free remeshing techniques with a curvature-based mesh regularization. The new approach allows for the control of both triangle quality and vertex sampling over the mesh, as a function of the surface mesh curvature. The adaptive remeshing proposal is discussed in Section 2.2.1
- A new approach to simplification based on the evolution of surfaces under p-Laplacian motion that provides a natural geometric clustering process where the spatial effect of the p-Laplacian allows for identifying suitable regions that need to be simplified. Discussed in Section 2.4.
- A Fast Interactive Reverse Engineering System (FIRES) enabling real-time acquisition and manipulation of complex geometrical shapes through wireless and interactive input devices. The developed project represents a low-cost solution to the challenging Reverse Engineering problem. We demonstrated that, by means of FIRES and the proposed virtual modeling tools, we can achieve optimal reconstruction results in terms of balncing cost and accuracy, as discussed in Part 2.
- A novel differential model for reconstructing free-form surfaces from sketched irregular curve networks. The proposed surfaced diffusion flow is able to reconstruct a smooth surface, while reproducing the sharp feature of the object (creases, corners and edges) as discussed in 6.4.

Introduzione - Italiano

In questa tesi sono trattati due argomenti principali "Geometric Surface Processing" e "Virtual Modeling". L'ispirazione e la coordinazione di gran parte del lavoro di ricerca contenuto nella tesi e' dovuta al progetto New Interactive and Innovative Technologies for CAD (NIIT4CAD), finanziato dall'European Eurostars Programme. NIIT4CAD ha l'ambizioso obiettivo di superare le limitazioni degli approcci tradizionali alla modellazione di superfici dei moderni sistemi di progettazione assistita al calcolatore, introducendo nuove metodologie e tecnologie basate su superfici di suddivisione in un nuovo framework virtuale di modellazione. Tali innovazioni permetteranno progettisti ed ingegneri a trasformare velocemente ed intuitivamente l'idea di una forma in un modello geometrico ad alta qualita' adatto per scopi ingegneristici e di produzione. Uno degli obiettivi della tesi e' proprio la ricostruzione e modellazione di superfici, rappresentanti oggetti a topologia arbitraria, partendo da curve 3D irregolari acquisite tramite un dispositivo smart-pen sviluppato ad-hoc.

La tesi e' organizzata in due parti: "Geometric Surface Processing" e "Virtual Modeling". Durante lo sviluppo della pipeline geometrica del nostro sistema di modellazione virtuale, abbiamo affrontato diverse problematiche che hanno attratto il nostro interesse ed aperto nuove aree di ricerca e sperimentazione. Nella prima parte, presentiamo tali teorie ed alcune applicazioni nell'ambito di Geometric Surface Processing. Questo ci permette di formalizzare meglio e dare una visione piu' ampia ad alcune delle tecniche usate nelle ultime versioni del nostro sistema ricostruzione di superfici e modellazione virtuale.

Il lavoro di ricerca per entrambi gli argomenti ha portato al raggiungimento di importanti risultati, brevemente riassunti in questa introduzione, che sono stati pubblicati e presentati in articoli e conferenze di rilevanza

internazionale.

Parte 1: Geometric Surface Processing

L'applicazione di modelli matematici basati su equazioni alle differenze parziali (PDE) a problemi di image processing e computer graphics e' stata estremamente proficua negli scorsi 20 anni. In particolare, flussi geometrici di superfici (geometric surface flows) sono stati estensivamente usati per l'elaborazione di mesh. Mentre una larga parte della comunita' scientifica in ambito di visione risolve modelli PDE basandosi su una metodologia Euleriana (tipicamente con level sets), rappresentazioni Lagrangiane di superfici basate su mesh triangolari sono piu' comuni in ambito di computer graphics. Nel setting Lagrangiano, la discretizzazione di flussi continui e' normalmente realizzata attraverso l'uso di operatori differenziali discreti oppure utilizzanto tecniche agli elementi finiti. In questa tesi, seguiremo il primo approccio e proporremo modelli differenziali su manifold e soluzioni numeriche a problemi di elaborazione di superfici quali ricostruzione, smoothing, remeshing, semplificazione e deformazione.

Sia $\mathcal{M}_0 = Image(X_0) := \{X_0(u), u \in [0,1] \times [0,1]\}$ una superfice compatta, chiusa e orientabile immersa in $\mathbb{R}^3$ e sia X_0 la corrispondente mappa parametrica. Una evoluzione geometrica della superfice consiste nel trovare una famiglia $\mathcal{M}(t) = Image(X(\cdot,t)), t \in [0,T), T > 0$ di superfici continue, chiuse e orientabili in $\mathbb{R}^3$ che evolvono secondo l'equazione del flusso (geometric flow)

$$\frac{\partial X}{\partial t} = -\beta \overrightarrow{N} + \alpha \overrightarrow{T},$$

dove $\overrightarrow{N}$ e' il vettore unitario normale alla superfice, β e' la velocita' lungo la direzione normale ed α e' la velocita' lungo la direzione tangente $\overrightarrow{T}$.

La famiglia di manifolds $\mathcal{M}(t) \in \mathbb{R}^4$ si muove lungo la direzione normale guidata dalla velocita' normale β dipendente, per esempio, dalla curvatura e posizione spaziale. Il moto normale ha un effetto di controllo sulla geometria della supeficie, mentre il moto tangenziale ha un effetto di ridistribuzione dei nodi.

Per approssimare numericamente le PDEs sulla superfice $\mathcal{M}(t)$, definiamo un setting discreto. L'approssimazione spaziale di $\mathcal{M}(t)$ e' una mesh poliedrica che consiste nell'unione di facce i cui vertici $X(t)$ giacciono su $\mathcal{M}(t)$, e $X(t)$ rappresenta la parametrizzazione della superfice stessa. In questo setting, possiamo definire, discretizzare ed approssimare modelli differenziali basati su operatori locali come il Laplace-Beltrami, il gradiente e la divergenza intrinsica.

Noi consideriamo sia flussi geometrici discreti, cioe' flussi basati sul corrispondente discreto di quantita' differenziali geometriche continue, vedi per esempio [69, 94], che metodi variazionali. Altri approcci sono basati sulla classica discretizzazione di modelli continui tramite schemi a volumi finiti ed elementi finiti[25], [6].

In questa tesi, presentiamo soluzioni numeriche al problema del **fairing**. Tale problema e' formulato in termini variazionali per derivare un approccio nonlocal che esegue il processo di smoothing della superfice seguendo un flusso di diffusione nonlocal (NL-SDF) su $\mathcal{M}$. I risultati sono raccolti in [78].

Remeshing significa ridistribuire e manipolare la connettivita' in modo da soddisfare specifiche proprieta', mantenento inalterate le caratteristiche importanti della superfice. In questa tesi presentiamo un metodo di remeshing adattivo che fa uso della curvatura media come misura instrinseca di regolarita'. Il metodo consiste in un modello PDE a due passi dove, nel primo passo la funziona area associata ai vertici $A(X)$, definita su M, e' diffusa sulla mesh, influenzata dalla curvatura media. Nel secondo step rilochiamo tangenzialmente i vertici in modo da ottenere edge approssimativamente della stessa lunghezza. I risultati sono esposti in [75].

La **semplificazione** di mesh e' il processo di riduzione della complessita' della mesh, preservandone topologia ed una buona approssimazione della geometria originale. La semplificazione di mesh e' un passaggio fondamentale nei sistemi di acquisizione 3D e nella deformazione multiresolution. In questa tesi presentiamo un nuovo approcio alla semplificazione di mesh basato sull'evoluzione della superficie con un flusso p-Laplaciano. Il flusso p-Laplacian e' usato per raggruppare i vertici verso regioni ad alta curvatura, mentre un processo di decimazione incrementale rimuove in modo ordinato gli edge piu' corti e meno rappresentativi. I risultati

sono stati presentati in [77].

La **deformazione** interattiva permette di manipolare intuitivamente un oggetto 3D in modo locale e globale. Il sistema di Reverse Reverse Engineering e modellazione virtuale (descritto nella Parte 2 della tesi), permette la manipolazione interattiva di forme attraverso metodi di deformazione basata su minimizzazioni di energie. Il requisito principale per deformazioni physically-based di superfici e' un'energia elastica che misura di quanto un oggetto si e' deformato rispetto alla sua configurazione iniziale. Noi presentiamo una nuovo metodo di deformazione globale e locale basato sulla minimizzazione della Curvatura Totale di una superfice.

Part 2: Virtual Modeling

Nella seconda parte discutiamo in dettaglio un nuovo metodo di reverse engineering per l'acquisizione e ricostruzione di modelli virtuali 3D rappresentanti oggetti fisici esistenti. Il sistema proposto, chiamato Fast Interactive Reverse Engineering System (FIRES)[9], sfrutta un sistema di acquisizione stereo attivo basato su un dispositivo tipo penna e supportato da un sistema di ricostruzione e visualizzazione basato su superfici di suddivisione. Semplicemente trascinando il dispositivo smart-pen nello spazio, l'utente disegna e raffina curve di stile 3D che definiscono l'outline della forma desiderata. Il processo di sketching di curve 3D e' realizzato tramite un sistema di stereovisione attiva costituito da due telecamere ad infrarossi ed un dispositivo smart-pen. Lo sketching di curve e' integrato con la ricostruzione in un processo iterativo ed incrementale, chiamato Interactive Surface Sketching (ISS), che permette all'utente di avere un feedback in tempo reale sul risultato del lavoro in corso. I dati grezzi ricevuti sia dalle camere che dalla smart-pen sono interpretati in una pipeline di acquisizione dove le proiezioni delle immagini dei led sono prima associati e triangolati. Poi, dopo essere identificati, i punti 3D sono usati per stimare la posizione della punta della penna, e la sequenza di tali punti traccia una curva che e' mostrata all'utente dopo un passo di smoothing e campionamento in tempo reale.

L'ISS produce un network di curve rappresentato da una Polyline Mesh,

cioe', una mesh con facce, vertici ed edge e ad ogni edge e' associata una polyline. Network di curve irregolari possono generare una polyline mesh con faccie ad n lati, non planari e non convesse.

Durante lo sviluppo di FIRES, abbiamo dedicato molte attenzioni al design di uno metodo di ricostruzione di superfici flessibile e robusto, capace di lavorare su network di curve irregolari e rumorosi. Il nostro primo approccio usava un metodo a piu' passi basato su patch di Coons che interpolano dati punti sulle curve. Piu' recentemente, dopo i risultati ottenuti nella Parte 1 sul processing geometrico di superfici, abbiamo introdotto un metodo di ricostruzione alternativo basato su un flusso di diffusione. In particolare viene prima costruita un mesh sufficientemente raffinata X_0 tramite una tassellazione dei poligoni nella mesh iniziale e poi viene applicato un flusso di diffusione del quarto ordine basato sulla curvatura. Tramite tale flusso il sistema di ricostruzione riesce a riprodurre superfici smooth ma anche angoli e spigoli degli oggetti. I risultati sono stati presentati in [76].

Part I

Geometric Surface Processing

Chapter 1

Geometric Flow on Surfaces and Variational Approaches

In this chapter we introduce geometric flows on surfaces. In the context of this thesis, we are interested in the study of how differential continuous models defined on manifolds can be approximated to discrete geometric flows on triangular meshes and applied to computer graphics and geometric modeling problems of our interest.

Traditional methods for surface design and processing have been focused on achieving specific levels of inter-element continuity via a combination of heuristics and constructions to achieve an ultimate shape. The final shape lacks, in general, of fairness, defined as smooth and minimal variation of curvature[71]. Pioneer works in [71, 72, 73, 105] introduced new techniques for curve and surface design based on a variational approach, that is a constrained optimization of a fairness functional.

Since then, many problems in surface processing has been reformulated as variational problems, i.e. minimizing a certain kind of energy functional. This leads to the solution of PDEs deriving from the Euler-Lagrange equations associated to the variational problem. Sometime the behavior of a particular phenomenon is well known and this makes it possible to directly "design" PDE models that suits a particular phenomenon without passing through the minimization of an energy functional.

The solution of PDE models on surfaces is strictly tied with the representation used to define the surface. The two common surface represen-

Table 1.1: Solving variational problems with different methods on different surface representations.

Representation	Method	Principle	Comments
Implicit	Level set approach	The surface is represented by the zero set of a level set function and the PDE on the surface is extended to a PDE that is defined on a narrow band of the surface.	Easy to handle topological changes.
Parametric	Surface parameterization	The surface is parametrized to a simple domain such as the 2D rectangle. Differential operators on the surface are expressed within the coordinates system.	Not easy to obtain parametrization for an arbitrary topology surface.
Lagrangian	Discretization on surface triangulation	Discrete differential geometry	Hard to handle topological changes. Can deal with any surface without preprocessing.

tations and corresponding approaches to variational problem and PDE models are described below and summarized in Table 1.1.

From a high level point of view, there are two major classes of surface representations: parametric representations and implicit representations.

Parametric surfaces are defined by a vector-valued parametrization function $f : \Omega \to \mathcal{M}$, that maps a two-dimensional parameter domain $\Omega \in \mathbb{R}^2$ to the surface $\mathcal{M} = f(\Omega) \in \mathbb{R}^3$. In contrast, an implicit (or volumetric) surface is defined to be the zero-set of a scalar-valued function called signed distance function, $F : \mathbb{R}^3 \to \mathbb{R}$, i.e., $\mathcal{M} = \{x \in \mathbb{R}^3 | F(x) = 0\}$.

Parametric and implicit representations have their particular strengths and weaknesses, such that for each geometric problem the better suited one should be chosen. Geometric operations can be classified in the following three categories:

Evaluation:
sampling of the surface geometry or of other surface attributes.

Query:
spatial queries are used to determine whether or not a given point $p \in \mathbb{R}^3$ is inside or outside of the solid bounded by a surface $\mathcal{M}$.

Modification:

a surface can be modified in terms of geometry, or in terms of topology.

Variational problems or PDEs on surfaces can be solved by parameterizing the surface onto the 2D parameter domain. In this case, classical differential geometry provides a well-known support to define and compute differential operators on the surface with respect the parametric domain[33].However, the computation of a parametrization is a complicated pre-processing for arbitrary given surfaces that can require the use of atlas in case of arbitrary topology. Another common approach is to solve the PDE on the implicit manifold, which is based on the level set method.In this approach, the surface is the zero set of level set function defined in $\mathbb{R}^3$, in which the surface is embedded in. The PDE on the surface is extended to be defined on a narrow band of the surface. In the Lagrangian representation, the surface $\mathcal{M}$ is explicitly represented as a piecewise-linear mesh M.The differential operators are approximated on surfaces by combining the standard Euclidian differential operators with projection along the normal direction. The biggest advantage of implicit representation of surfaces is that one can easily handle topological change under surface evolution. However, it has its own limitations. For instance, for open surfaces or surfaces with complicated structures it is not easy to obtain their implicit representations. In addition, the cost of the implicit representations is the pre-step to extend all data on the definition domain of implicit function. These additional increasing data might decrease the computation speed.

The above methods mainly focus on converting problems on surfaces to problems in Euclidean space. They require pre-processing, either extending data to the narrow band of the given surface or finding a parametrization of the given surface. Different numerical approaches to the approximation of continuous PDE models on meshes are based on the classical discretization of continuous models by finite volume, finite differences and finite elements schemes. For example, the authors in [108] use finite difference schemes, while in [25] a finite element approach is considered.

Over the last decade, a popular method to approach geometry processing and evolving surfaces has consisted of a Lagrangian setup where the surface $\mathcal{M}$ is explicitly represented as a piecewise-linear mesh M, and

vertices are moved so as to achieve the desired deformation[14]. Great success with this approach has been reported for editing, smoothing, and parameterization, often using variational formulations[88, 96]. In this approach, the variational models are directly solved on the given surface mesh instead of converting the models to be problems in Euclidean spaces.

Lagrangian methods have their own drawbacks including mesh element degeneracies, self-intersections, and topology changes, all of which require delicate treatment. Some of the shortcomings of solving PDEs or variational problems in a Lagrangian settings are not as noticeable for "short living" evolutions, that will be the case for most of our applications and examples.

In conclusion, since our focus is going to be primarily on PDE models that describe geometric flows for Computer Graphics and Computer Vision applications, where the Lagrangian surface representation is prevalently used, in this chapter we will delve into the process of discretization of differential operators for the purpose of solving discrete geometric flows, i.e., flows based on discrete analogous of continuous differential geometry quantities[69, 94].

1.1 Notation and geometric setting

We assume that the surface $\mathcal{M}$ is a 2-dimensional manifold of arbitrary topology embedded in $\mathbb{R}^3$. For some index set $\mathcal{A}$, we denote by $(\Omega_\alpha, X)_{\alpha\in\mathcal{A}}$ a chart of $\mathcal{M}$, that can be viewed as a piecewise parametrization of $\mathcal{M}$, where $\Omega_\alpha \subset \mathbb{R}^2$ is an open reference domain and

$$X : \Omega_\alpha \to \mathcal{M}; \xi \mapsto X(\xi),$$

is the corresponding coordinate map (that is, the parametrization of $\mathcal{M}$ at a given point). We denote the local coordinates Ω_α as (ξ_1, ξ_2).

For a given point $x \in X(\Omega_\alpha) \subset \mathcal{M}$, the tangent space $T_x\mathcal{M}$ at x is spanned by $\{\frac{\partial X(x)}{\partial \xi_1}, \frac{\partial X(x)}{\partial \xi_2}\}$. We use $T\mathcal{M}$ to denote the set of the tangent vector fields. For easy of notation, we from now on drop the subscript α.

Let M be a triangular mesh which is a piecewise-linear approximation

of the smooth manifold $\mathcal{M}$ with arbitrary topology, M is defined by a set T of triangles $T_i, i = 1, \ldots, N_t$, that cover M, and a set X of vertices $X_i, i = 1, \ldots, N_v$. We define an area element A_i around each vertex X_i. More precisely, if $N(i)$ is the set of 1-ring neighbor vertices of vertex X_i, then the associated area A_i, is defined by connecting the midpoints of each 1-ring edge with the barycenter of the elements. Note that, with an abuse of notation, we are defining X as the parametrization of $\mathcal{M}$ and also defining X as a set of vertices in the discrete setting.

1.2 Intrinsic Gradient and divergence discretization

In this section we define a discretization of the differential operators intrinsic gradient ($\nabla_{\mathcal{M}}$) and intrinsic divergence ($div_{\mathcal{M}}$) for the space discretization of $\mathcal{M}$ given by the mesh M.

Let $\mathcal{M} \subset \mathbb{R}^3$ be a surface, we can naturally consider the three coordinate functions characterizing the parametrization X as three scalar functions on the surface $\mathcal{M}$. From now on we consider a generic function $f : \mathcal{M} \to \mathbb{R}$, $f \in C^1(\mathcal{M})$, and we apply $\nabla_{\mathcal{M}} X$ by computing $\nabla_{\mathcal{M}} x$, $\nabla_{\mathcal{M}} y$ and $\nabla_{\mathcal{M}} z$. The intrinsic gradient operator $\nabla_{\mathcal{M}}$ of a function f is a vector field $\in T\mathcal{M}$ that can be written in local coordinates

$$\nabla_{\mathcal{M}} f = \sum_{i,j=1}^{2} g^{ij} \frac{\partial f}{\partial \xi_j} \frac{\partial}{\partial \xi_i}, \tag{1.1}$$

where $g_{ij} = \frac{\partial}{\partial \xi_i} \cdot \frac{\partial}{\partial \xi_j}$ ($\cdot$ indicates the inner product) are the coefficients of the metric matrix G and $(g^{ij})_{i,j=1,2}$ are the elements of the inverse matrix G^{-1}.

The intrinsic divergence operator $div_{\mathcal{M}}$ on $\mathcal{M}$ of a vector field $V \in T\mathcal{M}$ defined in local coordinates $V = v_1 \frac{\partial}{\partial \xi_1} + v_2 \frac{\partial}{\partial \xi_2}$ can be written in local coordinates

$$div_{\mathcal{M}} V = \frac{1}{\sqrt{det(G)}} \sum_{i=1}^{2} \frac{\partial}{\partial \xi_i} (\sqrt{det(G)} v_i). \tag{1.2}$$

where

$$\begin{array}{rcl} \frac{\partial}{\partial\xi_1}v_1 & = & g^{11}(V(X_{j_1})-V(X_i))\cdot\frac{\partial}{\partial\xi_1}+g^{12}(V(X_{j_1})-V(X_i))\cdot\frac{\partial}{\partial\xi_2}, \\ \frac{\partial}{\partial\xi_2}v_2 & = & g^{21}(V(X_{j_2})-V(X_i))\cdot\frac{\partial}{\partial\xi_2}+g^{22}(V(X_{j_2})-V(X_i))\cdot\frac{\partial}{\partial\xi_2}. \end{array} \tag{1.3}$$

Since $\sqrt{det(G)}$ is constant on each triangle, on a given triangle T we have

$$div_T V = \sum_{i=1}^{2}\frac{\partial}{\partial\xi_i}v_i. \tag{1.4}$$

We consider a weighted average in the first ring neighbors, in terms of the triangle area, thus we use the following discretization,[60]

$$\nabla_{\mathcal{M}} f(X_i) = \frac{1}{\sum_{j\in N(i)} A_j}\sum_{j\in N(i)} A_j \nabla_{T_j} f(X_i) \tag{1.5}$$

with A_j area of the triangle jth in the first ring neighbor of vertex X_i, and

$$\nabla_{T_j} f(X_i) \;=\; (f(X_{j_1})-f(X_i), f(X_{j_2})-f(X_i))G^{-1}\begin{pmatrix} X_{j_1}-X_i \\ X_{j_2}-X_i \end{pmatrix}. \tag{1.6}$$

For a triangle T of vertices X_i, X_{j_1}, X_{j_2}, with $j_1 \in N(i), j_2 \in N(i)$ we define

$$\begin{array}{rcl} \overrightarrow{\omega}_T^{X_{j_1}} & = & g^{11}(X_{j_1}-X_i)+g^{12}(X_{j_2}-X_i) \\ \overrightarrow{\omega}_T^{X_{j_2}} & = & g^{21}(X_{j_1}-X_i)+g^{22}(X_{j_2}-X_i) \\ \overrightarrow{\omega}_T^{X_i} & = & -(\overrightarrow{\omega}_T^{X_{j_1}}+\overrightarrow{\omega}_T^{X_{j_2}}). \end{array} \tag{1.7}$$

Then (1.6) can be rewritten as

$$\nabla_{T_j} f(X_i) \;=\; \textstyle\sum_{k\in i,j_1,j_2\prec T_j} \overrightarrow{\omega}_{T_j}^{X_k} f(X_k). \tag{1.8}$$

In matrix form, the discretization of the intrinsic gradient operator $\nabla_{\mathcal{M}}$ is given by:

$$\nabla_{\mathcal{M}} f \approx \overrightarrow{W} f \tag{1.9}$$

with $\overrightarrow{W} = D\overrightarrow{Ws}$, D is a diagonal matrix $D_{ii} = \frac{1}{\sum_{j\in N(i)} A_j}$, and

$$\overrightarrow{Ws}_{ij} = \begin{cases} -\sum_{j\in N(i)} A_j \overrightarrow{\omega}_{T_j}^{X_i} & i = j \\ (A_R \overrightarrow{\omega}_{T_R}^{X_j} + A_L \overrightarrow{\omega}_{T_L}^{X_j}) & i \neq j, j \in N(i) \\ 0 & otherwise \end{cases} \tag{1.10}$$

where T_L and T_R are the two triangle which share the edge X_i, X_j (see Fig. 1.1).

We next discretize the divergence operator for a given vector field V, following [60]

$$div_{\mathcal{M}} V(X_i) = \frac{1}{\sum_{j\in N(i)} A_j} \sum_{j\in N(i)} A_j div_{T_j} V(X_i). \tag{1.11}$$

In matrix form

$$div_{\mathcal{M}} V \approx \overrightarrow{W} \cdot V \tag{1.12}$$

where in right-hand side, since $\overrightarrow{W}$ consists of three matrices (W^x, W^y, W^z), and V consists of three vectors (V^x, V^y, V^z), by $\overrightarrow{W} \cdot V$ we denote the sum of three matrix-vector products $W^x V^x + W^y V^y + W^z V^z$.

1.3 Laplacian discretization

The Laplace operator or Laplacian is a differential operator given by the divergence of the gradient of a function f on Euclidean space Thus if f is a twice-differentiable real-valued function, then the Laplacian of f is defined by

$$\triangle f = \operatorname{div} \nabla f. \tag{1.13}$$

In a Cartesian coordinate system, the Laplacian is given by sum of second partial derivatives of the function with respect to each independent variable.

The Laplace-Beltrami operator extends this concept to functions defined on manifolds:

$$\triangle_{\mathcal{M}} f = \operatorname{div}_{\mathcal{M}} \nabla_{\mathcal{M}} f. \tag{1.14}$$

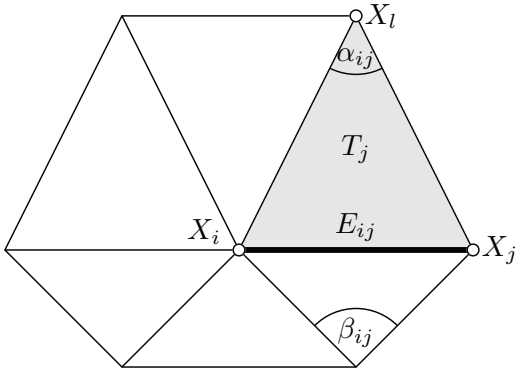


Figure 1.1: Stencil of the first ring neighborhood of the vertex X_i. The triangle T_j is defined by the vertices X_i,X_j, and X_ℓ.

Like the Laplacian, the Laplace-Beltrami operator is defined as the divergence of the gradient, and is a linear operator taking functions into functions.

Let us focus on the discretization of $\triangle_{\mathcal{M}} f$ on a spatial approximation of $\mathcal{M}$ represented by the mesh M with vertex set X.

A vertex $X_i \in X$, usually defined by Cartesian coordinates $X_i = (x_i, y_i, z_i)$, can be represented in differential coordinates (δ-coordinates) as $\delta_i = (\delta_i^x, \delta_i^y, \delta_i^z)$[79], where

$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} X_j - X_i = \frac{1}{d_i} \sum_{j \in N(i)} (X_j - X_i), \tag{1.15}$$

and d_i is the number of one-ring vertex neighbors of X_i, also called the valence of X_i. Note that (1.15) represents the offset vector of a vertex X_i from its gravity center weighted w.r.t. its neighbors X_j.

In a general form

$$\delta_i = \sum_{j \in N(i)} w_{ij}(X_j - X_i), \quad i = 1, \cdots, N_v, \tag{1.16}$$

where the weights w_{ij} are positive numbers and satisfy the normalization condition $\sum_{j \in N(i)} w_{ij} = 1$. For uniform weights $w_{ij} = \frac{1}{d_i}$ we get (1.15).

In matrix-vector form, the linear transformation (1.16) is expressed by

$$Lx = \delta^x, \quad Ly = \delta^y, \quad Lz = \delta^z, \tag{1.17}$$

where $x, y, z \in \mathbb{R}^{N_v}$ are vectors containing the Cartesian coordinates of all the vertices X, while the matrix $L \in \mathbb{R}^{N_v \times N_v}$ represents the connectivity of the mesh and it is called the topological Laplacian of the mesh. It is more convenient to consider the decomposition $L = D\,\overline{\mathrm{L}}$, where $D \in \mathbb{R}^{N_v \times N_v}$ is a diagonal matrix with $D_{ii} = 1/A_i$, which represents the inverse area term.

The elements of the $\overline{\mathrm{L}}$ matrix depend on the choices of the weights in Eq. (1.16) and provide different geometric discretizations of the Laplacian. However, the accuracy of the approximation of the Laplace-Beltrami operator on an arbitrary triangulated surface depends on the quality of the mesh.

In the following we present the common choices for weights in Eq. (1.16).

Cotangent weights[69]

$$\overline{\mathrm{L}}_{ij} = \begin{cases} -\sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) & i = j \\ +(\cot \alpha_{ij} + \cot \beta_{ij}) & i \neq j, j \in N(i) \\ 0 & otherwise \end{cases} \tag{1.18}$$

α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles sharing the edge (X_j, X_i), see Fig. 1.1.
$\overline{\mathrm{L}}$ is a symmetric matrix, and the D matrix used is: $D_{ii} = 1/2A_i$. Imposing the normalization condition we have $D_{ii} = 1/(2A_i \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij}))$. For a scalar function η defined on $\mathcal{M}$, Meyer et al. discretization [69] leads to

$$\triangle_{\mathcal{M}} \eta(X_i) \approx \frac{1}{2A_i} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\eta(X_j) - \eta(X_i)). \tag{1.19}$$

Umbrella Uniform[99]

$$\overline{\mathrm{L}}_{ij} = \begin{cases} -d_i & i = j \\ 1 & i \neq j, j \in N(i) \\ 0 & otherwise \end{cases} \tag{1.20}$$

For the *umbrella* discretization $D_{ii} = 1/d_i$, and $\overline{\mathrm{L}}$ is a symmetric and semi-def. pos. matrix.

Umbrella Scaled[40]

Improved umbrella operator proposed in [99] using the relaxation that the edge length are not supposed to be constant.

$$\overline{\mathrm{L}}_{ij} = \begin{cases} -\sum_{j \in N(i)} \frac{1}{\|X_i - X_j\|} & i = j \\ \frac{1}{\|X_i - X_j\|} & i \neq j, j \in N(i) \\ 0 & otherwise \end{cases} \tag{1.21}$$

In the $scaled - umbrella$ discretization $D_{ii} = 2/\sum_{j \in N(i)} \|X_i - X_j\|$, and $\overline{\mathrm{L}}$ is symmetric and semi-def. pos.

Mean Value[39]

$$\overline{\mathrm{L}}_{ij} = \begin{cases} -\sum_{j \in N(i)} \frac{(\tan \gamma_{ij}/2 + \tan \eta_{ij}/2)}{\|X_i - X_j\|} & i = j \\ \frac{(\tan \gamma_{ij}/2 + \tan \eta_{ij}/2)}{\|X_i - X_j\|} & i \neq j, j \in N(i) \\ 0 & otherwise \end{cases} \tag{1.22}$$

where γ_{ij} and η_{ij} are the two angles at X_i in the two triangles sharing the edge (X_j, X_i).

In $mean - value$ discretization $D = I$, and $\overline{\mathrm{L}}$ is not symmetric.

Given any choice of weights, for a given function η on $\mathcal{M}$ the discretization of k-th order Laplacians $\triangle^k_{\mathcal{M}}$ can be obtained recursively with:

$$\triangle^k_{\mathcal{M}} \eta(X_i) \approx L^k(X) = w_i \sum_{j \in N(i)} w_{ij} (\triangle^{k-1}_{\mathcal{M}} \eta(X_j) - \triangle^{k-1}_{\mathcal{M}} \eta X_i) \tag{1.23}$$

and the corresponding matrix representation L^k is simply the k-th power of the Laplacian matrix L.

Note that every row of $\overline{\mathrm{L}}$ sums up to zero, and

$$rank(\overline{\mathrm{L}}) = N_v - k, \tag{1.24}$$

with k number of connected components of the mesh. A connected component of a mesh consists of a submesh with a number of vertices greater than one where each vertex is connected to the others by a path of edges. Therefore a connected mesh, with or without boundaries, has

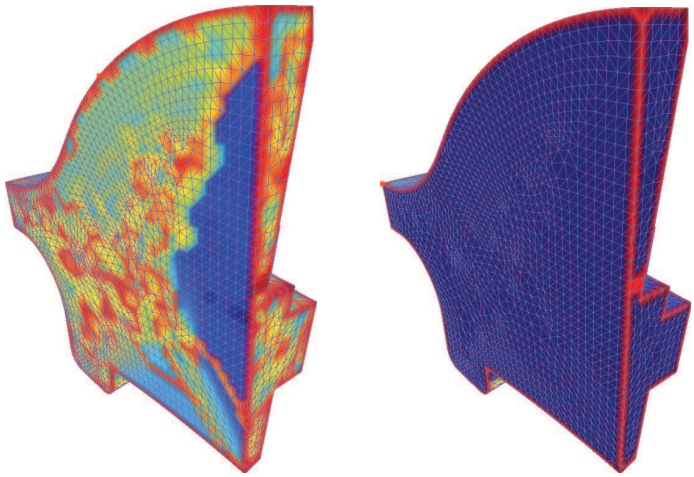


Figure 1.2: Results of applying umbrella weights (1.20) on the left and cotangent weights (1.18) on the right.

$rank(\overline{\mathrm{L}}) = N_v - 1$. The connectivity matrix L representing a mesh with N connected components, consists of a block matrix with N blocks.

The kernel of $\overline{\mathrm{L}}$ consists of constant vectors, that is $Ker(\overline{\mathrm{L}}) = \{cI \in \mathbb{R}^{N_v \times 1}\}$, c scalar value, thus we have

$$dim(Ker(\overline{\mathrm{L}})) + rank(\overline{\mathrm{L}}) = N_v, \tag{1.25}$$

which is the dimension of the matrix $\overline{\mathrm{L}}$.

Unfortunately, $\overline{\mathrm{L}}$ is symmetric and semi-definite positive only for some choices of geometric discretizations, however, the matrix $\overline{\mathrm{L}}\, D\, \overline{\mathrm{L}}$ is a symmetric positive definite matrix and $ker(\overline{\mathrm{L}}\, D\, \overline{\mathrm{L}}) = ker(\overline{\mathrm{L}})$, and we can use this result for further developments.

In [107] has been shown that all the discretizations L presented above are not convergent in the general cases to the Laplace-Beltrami operator $\triangle_{\mathcal{M}}$ applied to $f \in C^2(\mathcal{M})$. Only Meyer et al. [69] is convergent for some special cases.

The cotangent discretization (1.19) leads to vectors δ_i with normal components only, unlike the other discretizations which also have tangential components and may be nonzero on planar one-ring neighbors (see Fig. 1.2).

From differential geometry we know [33] that the Laplace-Beltrami operator is strictly related to the concept of curvature. Let X_i be a point on a differentiable manifold $\mathcal{M}$. Every curves on $\mathcal{M}$ passing through X_i has an associated curvature value κ_j. The maximum and minimum values of curvature, κ_1 and κ_2 respectively, are known as the principal curvatures of $\mathcal{M}$ at X_i. The mean curvature at X_i is then the sum of the principal curvatures:

$$H(X_i) = \kappa_1 + \kappa_2. \tag{1.26}$$

while the Gaussian curvature is the product of the principal curvatures:

$$K_G(X_i) = \kappa_1 \kappa_2. \tag{1.27}$$

The mean curvature normal vector, denoted by $\overrightarrow{H}(X)$, equals the Laplace-Beltrami operator ($\triangle_{\mathcal{M}}$) applied to the identity id on a surface $\mathcal{M}$:

$$\overrightarrow{H}(X) = H(X)\overrightarrow{N}(X) = -\triangle_{\mathcal{M}} X, \tag{1.28}$$

where $H(X)$ is the corresponding mean curvature scalar field and $\overrightarrow{N}(X)$ is the unit outward normal of the surface at point X. Thus from a differential geometry point of view the δ-coordinates in (1.17) can be considered as a discretization of the continuous Laplace-Beltrami operator.

Therefore the differential coordinate vector δ is characterized by a direction which approximates the local normal direction and by a magnitude which is proportional to the local mean curvature.

The mean curvature $\overrightarrow{H}(X)$ can be also expressed by

$$\overrightarrow{H}(X) = div_{\mathcal{M}}(\nabla_{\mathcal{M}}(X)) \tag{1.29}$$

using the discretized operators (1.5) and (1.11) described in Section 1.2. The discretization of the Laplace, gradient and divergence operators at the vertex X_i depends on the elements of its first ring neighborhood while the measure of mean curvature at the vertex X_i depends on its 2-ring neighbors. The discretized mean curvature (1.29) is more reliable than the discretized mean curvature obtained through the relation (1.28), mostly because it is computed on a wider stencil which allows to better identify the features of the mesh.

Given the δ coordinates, in order to recover the Cartesian coordinates of the vertices of the mesh M we solve the linear systems (1.17). However the matrix L is singular, since $rank(L)$ is given by (1.24).Thus for a connected mesh M the linear systems (1.17) are not full-rank.

In order for the L matrix to be non-singular (and for the geometric flow to have a well-defined solution), suitable boundary constraints have to be employed. At this aim the values at a set of constrained vertices X_i are prescribed (so-called Dirichlet constraints) and their corresponding columns are moved to the right hand side[13].

Remark

The volume V of the mesh M can be approximated by the formula

$$V = \frac{1}{6} \sum_{k=1}^{N_t} \gamma_k \cdot \overrightarrow{N}_k, \tag{1.30}$$

where $\overrightarrow{N}_k$ is the normal vector to the face k characterized by the vertices X_1^k, X_2^k, X_3^k, and $\gamma_k = (X_1^k, X_2^k, X_3^k)/3$.

Two essentially different models that have found wide recognition in surface processing are the variational approach according to some functionals and the approach via nonlinear diffusion PDEs.

1.4 Geometric flow based on energy minimization

The basic idea of the variational design approach is to measure the quality of a surface in terms of its bending energy. A pioneer work in this field has been introduced by Moreton in [71]. Let us denote by $\mathcal{M}$ a two-manifold surface, parametrized by a function $X : \Omega \subset \mathbb{R}^2 \rightarrow \mathcal{M} \subset \mathbb{R}^3$. The most common functional, which approximates the bending energy of a thin plate, is the **total curvature energy**

$$E(\mathcal{M}) := \frac{1}{2} \int_{\mathcal{M}} k_1^2 + k_2^2 d\mathcal{M}, \tag{1.31}$$

where k_1 and k_2 are the principal curvatures which depend non-linearly on the surface $\mathcal{M}$. Let us call the surfaces minimizing (1.31) elastica surfaces because they generalize the famous Euler's elastica curves.

However, since the principal curvatures and the area element depend non-linearly on the surface $\mathcal{M}$, this functional is difficult to minimize. The total curvature is therefore replaced, that is "linearized", by the so-called **thin plate energy**:

$$E_{BEND}(X) = \frac{1}{2}\int_{\Omega} X_{uu}^2 + 2X_{uv}^2 + X_{vv}^2 dudv, \tag{1.32}$$

which is a standard measure used for the global surface quality in geometric modeling. When the parametrization is isometric, then (1.32) turns out to be equal to (1.31).

The **membrane energy** is defined by the functional

$$E_{STRETCH}(X) = \frac{1}{2}\int_{\Omega} X_u^2 + X_v^2 dudv = \frac{1}{2}\int_{\Omega} |\nabla X|^2 dudv. \tag{1.33}$$

In order to keep the parametrization of the surface $\mathcal{M}$ as close to isometric as possible, Ω is typically chosen equal to the initial surface $\mathcal{M}$, As a consequence, the gradient turns into the intrinsic gradient, the Laplace operator $\triangle$ w.r.t. the parametrization X turns into the Laplace-Beltrami operator $\triangle_{\mathcal{M}}$ w.r.t. the manifold $\mathcal{M}$, and the functionals (1.32) and (1.33) becomes

$$E_{BEND}(\mathcal{M}) = \frac{1}{4}\int_{\mathcal{M}} (k_1 + k_2)^2 d\mathcal{M} = \int_{\mathcal{M}} H^2 d\mathcal{M} = \int_{\mathcal{M}} |\triangle_{\mathcal{M}} X|^2 d\mathcal{M}, \tag{1.34}$$

and

$$E_{STRETCH}(\mathcal{M}) = \frac{1}{2}\int_{\mathcal{M}} |\nabla_{\mathcal{M}} X|^2 d\mathcal{M}, \tag{1.35}$$

respectively. The minimization of these functionals can be performed efficiently by applying variational calculus, which yields their Euler-Lagrange PDEs, which are

$$-\triangle_{\mathcal{M}} X = 0, \tag{1.36}$$

for (1.35), so-called *mean curvature*, and

$$\triangle_{\mathcal{M}} \circ \triangle_{\mathcal{M}} X = \triangle_{\mathcal{M}}^2 X = 0, \tag{1.37}$$

for (1.34), so-called *bi-Laplacian*, which involve Laplacian and bi-Laplacian operators, respectively, and

$$\triangle_{\mathcal{M}} H + 2H(H^2 - K_G) = 0, \tag{1.38}$$

for (1.31), so-called *discrete elastica* where K_G is the Gaussian curvature. Considering an artificial time evolution variable t, PDEs (1.36), (1.37) and (1.38) turn into the geometric flow defined as *mean curvature flow*, *bi-Laplacian flow* and *Euler discrete elastica flow*, respectively.

Solving ${\triangle_{\mathcal{M}}}^2 X = 0$, with suitable boundary conditions securing its non-trivial solution, produces the so-called thin-plate surface while solving $\triangle_{\mathcal{M}} X = 0$ with suitable boundary conditions produces the so-called minimal area surface.

1.5 Discrete Geometric Flows

Let $\mathcal{M}_0 = Image(X_0) := \{X_0(u), u \in [0,1] \times [0,1]\}$ be a compact, closed immersed orientable surface in $\mathbb{R}^3$. Here X_0 denotes the corresponding parameter map. A curvature-driven geometric evolution consists of finding a family $\mathcal{M}(t) = Image(X(\cdot,t)), t \in [0,T), T > 0$ of smooth, closed, immersed orientable surfaces in $\mathbb{R}^3$ which evolve according to the flow equation (**Geometric flow**)

$$\begin{array}{lcll} \frac{\partial X}{\partial t} & = & -\beta \overrightarrow{N}, & \text{with initial condition} \\ X(0) & = & X_0, & \end{array} \tag{1.39}$$

where $\overrightarrow{N}$ is the unit normal vector to the surface, and β is a velocity applied along the normal direction. The parameter t can be considered as the time duration of the evolution.

The family of manifolds $\mathcal{M}(t) \in \mathbb{R}^4$ moves along the normal direction driven by a normal velocity β which may be a function, for example, of the curvature and spatial position. Eq. (1.39) represents the evolution of the surface $X(t)$ along its normals with speed equal to β. Note that in Eq. (1.39) the tangential contribution is not considered. The normal motion controls the geometry of the surface while the role of the tangential velocity is a sort of redistribution of the nodes which improves the accu-

racy of the surface representation. Considering a tangent movement, the geometric flow becomes

$$\frac{\partial X}{\partial t} = -\beta \overrightarrow{N} + \alpha \overrightarrow{T}, \tag{1.40}$$

where α is the velocity in the tangent direction $\overrightarrow{T}$. The vertex tangent movement is used to improve the regularity of the mesh, that is the mesh vertex distribution.

Considering a uniform discretization of the time interval $[0, T], T > 0$, and using a temporal time step $\tau = T/n$, the approximation of an evolving surface at the n-th time step is denoted by a spatial position vector X^n. This system of PDEs can be discretized in time using a variety of explicit or implicit time integration schemes. In our computational framework, we use a simple time integration method that is the forward Euler scheme

$$\frac{\partial X}{\partial t} \approx \frac{X^{n+1} - X^n}{\tau}, \tag{1.41}$$

which yields a first order scheme in time. For solving (1.39) we can apply explicit schemes:

$$X^{n+1} = X^n + \tau\beta \overrightarrow{N}^n, \tag{1.42}$$

or implicit schemes

$$(I - \tau\beta \overrightarrow{N}^{n+1}) X^{n+1} = X^n. \tag{1.43}$$

In the explicit scheme the numerical stability is conditioned by a time step bounded by

$$\tau \leq \frac{(min\|e\|)^2}{2}, \tag{1.44}$$

where $\|e\|$ is the edge length. The implicit scheme is unconditionally stable, but the time step is chosen according to the geometric criterium

$$\tau = \frac{\bar{e}}{\max \|\delta_i\|}, \tag{1.45}$$

where $\bar{e}$ is the average edge length and the denominator represents the maximum norm value of the Laplacian vectors.

1.6 Linear system solvers for discrete geometric flows

The discretization of PDE models mainly leads to the solution of one or several linear systems. These linear systems are always of very large dimensions since the size of the linear systems corresponds at least to the number of vertices in the mesh. The coefficient matrix is characterized by a band structure, sparsity and can be symmetric positive definite.

Since the differential operators in the geometric flow are locally defined, the discretization of PDEs typically leads to sparse linear systems, in which the ith row contains non-zero values only in those entries corresponding to the topological neighborhood of a vertex X_i. We are interested in iterative solvers that exploit this sparsity in order to minimize both memory consumption and computation time.

In particular, for the class of sparse linear systems that are also symmetric def. pos. we apply a preconditioned conjugate gradient solver with incomplete Cholesky as preconditioner. Such systems frequently occur when we deal with a discrete Laplace-Beltrami operator, Bi-Laplacian, etc., discretized as in section 1.3. If this is not the case we use the GMRES iterative solver[92]. Finally, all linear problems $Ax = b$ that require to be approximated in the least-squares sense, can be solved using the normal equations $A^T Ax = A^T b$ which result again in a sym. def. pos. linear system that we solve by the LSQR iterative method [87]. A very popular source of this systems are the minimizations of energy functional for surface deformation. The stopping criteria used in our experimental work is 10^{-6}.

Furthermore, we observe that most discrete geometric flows are separable w.r.t. the coordinate components, that is, they can be solved componentwise for x, y and z with the same system matrix.

More elaborate surveys on how to efficiently solve general large linear systems can be found in[11].

REMARK: In the evolution flow the geometry of the mesh changes, thus also L should change accordingly, apart when L does not depend on the geometry of the mesh (e.g. umbrella discretization). Following the literature, when the measures involved (edge length, cotangent, ...) do

not change dramatically during a typical iteration, the same initial L can be maintained.

Chapter 2

Differential Models for Computer Graphics and Geometric Modeling

The 3D geometry commonly used for shape representation in geometric modeling, physical simulation and scientific visualization is mainly based on meshes. The 3D scanning devices, medical equipments and computer vision systems often perform a dense uniform acquisition of points on the surface without any a priori knowledge of the surface structure. This may lead to raw meshes with a sampling quality usually far away from the desired sampling distribution needed for subsequent processing. Algorithms for mesh simplification, denoising (fairing), decimation and remeshing represent fundamental preliminary steps in mesh processing. In the following, we present original proposals for these problems based on geometric differential flows.

2.1 Smoothing

A surface smoothing method, in the following named fairing, removes undesirable noise and uneven edges from discrete surfaces. The fairing problem arises mainly when creating high-fidelity computer graphics objects using imperfectly-measured data from the real world, captured for example from 3D laser scanner devices. Fairing can be applied either be-

fore or after generating the mesh from sampled data. The advantage of denoising a mesh rather than a point-cloud, is that the connectivity information implicitly defines the surface topology and can be exploited as a means for fast access to neighboring samples.

The goal is to remove noise from a surface while keeping features, e.g. sharp edges, corners and ridges. Explicit surface representations such as meshes offer an easy way to discretize differential operators, but topological changes are harder to handle. However, for smoothing processing driven by PDE models, the evolution is sufficiently slow to avoid both topological modifications and triangle flips in triangular meshes.

The most common surface degradation model, when the observed data $X_0 \in \mathbb{R}^{3\times N_v}$ are corrupted by a random variation of the vector field, is

$$X = X_0 + \overrightarrow{E}, \tag{2.1}$$

where $\overrightarrow{E} \in \mathbb{R}^{n^3}$ accounts for the vector perturbations with Gaussian distribution.

Variational and PDE-based surface denoising models have had great success in the past ten years. Several authors presented isotropic/anisotropic denoising of surfaces applying image processing methodology based on linear/nonlinear diffusion equations [25][108][98][84][74].

We first introduce some of the most common PDE based smoothing models, like the mean curvature, the bi-Laplacian flows and the anisotropic mean curvature flow, then we discuss an original two step approach which implements a nonlocal surface diffusion flow on meshes. First, we smooth the mean curvature normal map of a surface, and next we manipulate the surface to fit the processed smoothed curvature normal vector field. We show that we can efficiently implement geometric fourth-order flow by solving a set of second order PDEs discretized on the mesh M. Inspired by [42] we integrate a nonlocal approach into this framework driven by a mean curvature based local geometric descriptor. Our proposal and its results have been published in [78].

2.1.1 Mean curvature flow (MCF)

According to the geometric flow (1.39), considering (1.28) and $\beta = H$ in (1.39), an initial surface represented by the position vector X_0, evolves with speed

$$\frac{\partial X}{\partial t} = \triangle_{\mathcal{M}} X \overrightarrow{N}(X), \quad X(0) = X_0, \tag{2.2}$$

which is a second order PDE. Applying an explicit scheme and Laplacian discretization, we get:

$$X^{n+1} = X^n + \tau L X^n. \tag{2.3}$$

Implicit scheme leads to

$$(I - \tau L) X^{n+1} = X^n, \tag{2.4}$$

where I is the identity matrix.

The mean curvature flow is known to have a strong regularization effect, because it is the gradient flow for the area functional. In a discrete setting, the mean curvature flow moves every vertex in the normal direction with the speed equal to a discrete approximation of the mean curvature at the vertex. It is also well known that the mean curvature flow performs well in smoothing (fairing) but produces uneven distribution of vertices.

In [35] the authors present finite element schemes for MCF on triangulated surfaces, similarly in [31] implicit and explicit discretizations are considered.

Unfortunately MCF not only decreases the geometric noise due to imprecise measurements, but also smooths out geometric features such as edges and corners of the surfaces. Recently, several authors presented anisotropic denoising of surfaces applying image processing methodology based on nonlinear diffusion equations [25][108][98][84].

We want to consider novel or different strategies for introducing anisotropy to the diffusion process.

2.1.2 Bi-Laplacian flow (BLF)

According to the geometric flow (1.39), considering (1.28) and the bi-Laplacian $\beta = \triangle^2$ in (1.39), an initial surface represented by the position vector X_0, evolves with speed

$$\frac{\partial X}{\partial t} = -\triangle^2_{\mathcal{M}} X \overrightarrow{N}(X), \quad X(0) = X_0. \tag{2.5}$$

Applying an explicit scheme and the discretization $\triangle^2_{\mathcal{M}} X \approx L(L(X))$, we get:

$$X^{n+1} = X^n - \tau L(LX^n). \tag{2.6}$$

Implicit scheme gives

$$(I + \tau L^2)X^{n+1} = X^n. \tag{2.7}$$

Since $L^2X = D\,\overline{\mathrm{L}}(D\,\overline{\mathrm{L}}\,X)$ with $\overline{\mathrm{L}}\,D\,\overline{\mathrm{L}}$ a matrix s.d.p., in order to obtain a symmetric linear system we rewrite (2.7) as:

$$(D^{-1} + \tau\,\overline{\mathrm{L}}\,D\,\overline{\mathrm{L}})X^{n+1} = D^{-1}X^n. \tag{2.8}$$

and choose τ such as the matrix is def.pos.

2.1.3 Anisotropic MCF (AMCF)

The simple model (2.2) can be extended considering an isotropic scalar function $f : X \to R^+$ for the motion in the normal direction. Thus each point of the surface mesh is moved, at each iteration, proportionally to its local curvatures. Using, for example, the monotonic decreasing function

$$f(s) = \frac{1}{1 + (\nu/s)^2}, \tag{2.9}$$

where s represents the local value of curvature of the mesh, points with large curvature (compared to ν) move faster than points on locally flat area.

According to the geometric flow (1.39), considering (1.28) and $\beta = -div_{\mathcal{M}}(f\nabla_{\mathcal{M}}X))$

in (1.39), an initial surface represented by the position vector X_0, evolves with speed

$$\frac{\partial X}{\partial t} = (div_{\mathcal{M}}(f\nabla_{\mathcal{M}} X)\cdot \overrightarrow{N})\overrightarrow{N}, \quad X(0) = X_0. \tag{2.10}$$

To avoid tangential velocity components in the evolution we projected the velocity in equation (1.39) in normal direction. Equation (2.10) can be rewritten as

$$\frac{\partial X}{\partial t} = ((f \triangle_{\mathcal{M}} X - \nabla_{\mathcal{M}} f \cdot \nabla_{\mathcal{M}} X)\cdot \overrightarrow{N})\overrightarrow{N}, \quad X(0) = X_0. \tag{2.11}$$

An anisotropic version of the model (2.10) can be obtained by replacing the feature-preserving geometric function f with an anisotropic diffusion tensor in terms of the principal curvatures, see for a detailed analysis [25], [26] and [27]. Let's introduce an anisotropic tensor

$$F = \begin{pmatrix} f(k_1) & 0 \\ 0 & f(k_2) \end{pmatrix}$$

in equation (2.10), where k_1 and k_2 are the principal curvature directions. This tensor leads to a surface classification as follows: smooth parts of the surface can be characterized by $F = diag[1,1]$.An edge can be defined via the relation $F = diag[1,0]$. In this case, the direction along the edge is given by e_2 where we assume here $|k_1| >> |k_2|$. In this setting, corners are given by the relation $F = diag[0,0]$.We may introduce as an edge-indicator the function $\eta(x) = tr(F)$. Depending on the parameter η edges and corners are given by $\eta < 1$.

Let us construct a matrix D_0 from the system of orthonormal eigenvectors,

$$\mathbf{v_1} \| \nabla_{\mathcal{M}} \mathbf{X}, \quad \mathbf{v_2} \perp \nabla_{\mathcal{M}} \mathbf{X}, \quad \mathbf{v_3} \perp \nabla_{\mathcal{M}} \mathbf{X} \quad \textbf{and} \quad \mathbf{v_3} \perp \mathbf{v_2} \tag{2.12}$$

which has the following form

$$D_0 = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & (\mathbf{v}_1 \times \mathbf{v}_2) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ (\mathbf{v}_1 \times \mathbf{v}_2)^T \end{bmatrix}. \tag{2.13}$$

Here $\mathbf{v}_1$ and $\mathbf{v}_2$ denote the embedded tangent vectors corresponding to $X_j - X_i$ and $X_k - X_i$ for the triangle $X_i X_j X_k$.

The corresponding eigenvalues are computed so that diffusion along edges is preferred over diffusion across them, that is, by

$$\begin{aligned} \lambda_1 &= f(k_1) \\ \lambda_2 &= f(k_2) \end{aligned} \tag{2.14}$$

where $f(\cdot)$ is defined in (2.9) and suitably adapts its values to the anisotropy, and ν has the role of a threshold parameter. The diffusion tensor is a smooth, symmetric and positive definite matrix. The positive definiteness of the matrix D_0 follows easily from (2.14) and from the positivity of $f(\cdot)$.

In local homogeneous areas the diffusion is reduced to be the isotropic mean curvature motion driven by (2.2), in fact, following (2.14), D_0 is replaced by f.

2.1.4 Total variation diffusion flow (TVF)

For a given surface $\mathcal{M}$ the total variation of a C^1 function $\phi : \mathcal{M} \to R$ is defined as

$$TV(\phi) := \int_{\mathcal{M}} |\nabla_{\mathcal{M}} \phi| ds, \tag{2.15}$$

and it is equivalent to the L_1 norm of the derivative and hence it is some measure of the amount of oscillation found in the function ϕ. The Rudin Osher Fatemi (ROF)[91] image denoising model can be generalized to surfaces. In this case, the surface analogous form of the total variational ROF model is represented as unconstrained optimization problem as follows

$$\min_{\phi \in L_2(\mathcal{M})} \int_{\mathcal{M}} |\nabla_{\mathcal{M}} \phi| ds + \frac{\mu}{2} \int_{\mathcal{M}} (\phi - f)^2 ds, \tag{2.16}$$

where $f : \mathcal{M} \to R$ is a function on the surface $\mathcal{M}$ and $\mu \geq 0$ is a Lagrange multiplier. The first term in (2.16) is a regularization term, and the second one is the L_2 fitting term which could be replaced with the L_1 fitting term $\int_{\mathcal{M}} |\phi - f| ds$.

The equation that minimizes energy (2.16) is its gradient descent flow:

$$\frac{\partial \phi}{\partial t} = (div_{\mathcal{M}}(\frac{\nabla_{\mathcal{M}}\phi}{\|\nabla_{\mathcal{M}}\phi\|}) \cdot \overrightarrow{N})\overrightarrow{N} + \mu(f-\phi), \quad \phi(u,0) = f. \tag{2.17}$$

According to the geometric flow (1.39), considering (1.28) and $\beta = -div_{\mathcal{M}}(\frac{\nabla_{\mathcal{M}}X}{\|\nabla_{\mathcal{M}}X\|})$ in (1.39), an initial surface represented by the position vector X_0, evolves with speed

$$\frac{\partial X}{\partial t} = (div_{\mathcal{M}}(\frac{\nabla_{\mathcal{M}}X}{\|\nabla_{\mathcal{M}}X\|}) \cdot \overrightarrow{N})\overrightarrow{N}, \quad X(0) = X_0. \tag{2.18}$$

here we considered $\mu = 0$.

Since (2.17) is difficult to solve numerically and it is slow, the dual method (Chambolle's projection method)[20] to solve the total variational problem (2.16) on surface can be applied. The procedure iterate the following steps. Set $V^0 = 0$.
Solve for V:

$$V^{n+1} = \frac{V^n + \tau\nabla_{\mathcal{M}}(div_{\mathcal{M}}V^n - \mu f)}{1 + \tau|\nabla_{\mathcal{M}}(div_{\mathcal{M}}V^n - \mu f)|}$$

where $\tau < 1/\|div_{\mathcal{M}}\|$ is the time step which guaranties the convergence of the iterative scheme.

For a more sophisticated version of TV, see [37], where the authors propose to minimize the energy

$$\int_{\mathcal{M}} |K_G| ds \tag{2.19}$$

where K_G is the Gaussian curvature of $\mathcal{M}$.

2.1.5 The nonlocal surface diffusion flow (NL-SDF)

We propose a new variational model for surface fairing. We extend nonlocal smoothing techniques for image regularization to surface smoothing or fairing, with surfaces represented by triangular meshes. Our method is able to smooth the surfaces and preserve features due to geometric

similarities using a mean curvature based local geometric descriptor. We present an efficient two step approach that first smooths the mean curvature normal map, and then corrects the surface to fit the smoothed normal field. This leads to a fast implementation of a feature preserving fourth order geometric flow. We demonstrate the efficacy of the model with several surface fairing examples.

In [32] the authors propose a point cloud nonlocal denoising using the signed distance function as local surface descriptor in a point-wise process. Similar descriptors are used in the nonlocal denoising method proposed in [111] where instead of the moving least square representation, the authors used local radial basis functions. In [42] a nonlocal diffusion process is derived as steepest descent of a nonlocal quadratic functional of weighted differences. This formulation is an excellent framework for nonlocal variational image denoise, Bregman iterations, and segmentation. A nonlocal heat equation for denoising surfaces has been introduced in [34], where the signed distance function is used to define the similarity weights, and the PDE evolution is solved using a level set formulation on an implicitly defined surface.

Nonlocal means image denoising

Nonlocal denoising is an algorithm for image denoising introduced in [18]. The algorithm aims to denoise a gray-scale image I, defined over a rectangular bounded domain Ω, by replacing each pixel with a weighted mean of the neighborhoods. The new value of the image pixel is

$$NL[I](x) = \int_\Omega W(x,y)I(y)dy, \tag{2.20}$$

where the convolution kernel $W(x,y)$ is given by

$$\begin{aligned} W(x,y) = &\quad \tfrac{1}{C(x)} e^{-D(I(x),I(y))/c} \\ D(I(x),I(y)) = &\quad \|I(x) - I(y)\|_2^2, \quad y \in N(x) \end{aligned} \tag{2.21}$$

with a normalization factor $C(x) = \int_\Omega W(x,y)$, $N(x)$ represents a neighborhood of x, and c is a filtering parameter which is related to the noise level. The similarity between pixels is measured by the similarity ker-

nel D in (2.21) and depends on the similarity of gray-level intensities in the neighborhood of x and y, that is, the algorithm not only compares the (color) value at a single pixel but the geometrical configuration in a whole neighborhood. The algorithm gives excellent results in image denoising (see [52],[18]). For a more detailed analysis on the NL-means algorithm see [18].

The nonlocal variational fairing

We propose a variational formulation in order to derive our nonlocal approach to surface fairing.

For a surface parameterization X of $\mathcal{M}$ on a domain Ω, and a given vector field $f \in \mathbb{R}^{N_v \times 3}$, we consider the minimization of the following functional

$$min_X \int_\Omega |\nabla_{w\mathcal{M}} X|^2 + \frac{\lambda}{2}(X - f)^2 d\omega, \tag{2.22}$$

where $\lambda > 0$ is a regularization parameter and $\nabla_{w\mathcal{M}}$ is a weighted gradient operator. The corresponding Euler-Lagrange descent flow can be written as

$$\frac{\partial X}{\partial t} = \int_\Omega (X(y) - X(x))W(x,y)d\omega + \lambda(f - X), \tag{2.23}$$

with $x, y \in \Omega$, (see [34] for a similar definition). Here $W(x,y)$ is the weight function, which satisfies $W(x,y) \geq 0$, and is symmetric $W(x,y) = W(y,x)$. For image processing the weight function can be defined as in (2.21). The spatial discretization of (2.23) on the mesh M, is

$$\frac{\partial X_i}{\partial t} = \sum_{j \in N(i)} W_{ij}(X_j - X_i) + \lambda(f_i - X_i), \tag{2.24}$$

where X_i denotes the value of X at the ith vertex, $i = 1, \ldots, N_v$, and $N(i)$ is the set of 1-ring neighbor vertices of the ith vertex.

Let $f(x) := (f^1, f^2, f^3)(x)$ be a vector field on $\mathcal{M}$, $W(x,y)$ is the same for all vector components. Let $X(x) := (X^1, X^2, X^3)(x)$ be the coordinate function vector on M, where X^1 is the scalar function that defines the first coordinate of point $x \in \mathcal{M}$, and analogously for the second and the third

coordinate scalar functions. Then the regularizing formulation (2.24) for each vector component $X^k, k = 1, 2, 3$, is

$$\frac{\partial X_i^k}{\partial t} = \sum_{j \in N(i)} W_{ij}(X_j^k - X_i^k) + \lambda(f_i^k - X_i^k), \tag{2.25}$$

by initializing, e.g., each component k of X as $X^k|_{t=0} = f^k$.

If we let $W_{ij} = w_{ij}$, with w_{ij} defined by (1.18), then the regularized PDEs (2.25) can be interpreted as the spatial discretization on M of the well know *mean curvature flow* (MCF)

$$\frac{\partial X}{\partial t} = \triangle_{\mathcal{M}} X + \lambda(X_0 - X), \quad X|_{t=0} = X_0, \tag{2.26}$$

with initial surface X_0. The first term in (2.26) is the *regularization* term, while the second one is the *fidelity* term.

We propose the following *nonlocal weighted Laplace-Beltrami operator* on M,

$$L_w X_i = \sum_{j \in N(i)} (X_j - X_i) W_{ij} w_{ij}, \tag{2.27}$$

where w_{ij} is defined as in section 1.3, while W_{ij} depends on a similarity measure between i*th* and j*th* vertex. A proposal of similarity weight functions in surface processing is discussed in the next Section.

By initializing $X|_{t=0} = X_0$ and using the nonlocal operator (2.27), then (2.24) can be rewritten as

$$\frac{\partial X_i}{\partial t} = L_w X_i + \lambda(X_{0_i} - X_i). \tag{2.28}$$

In the next Section, we apply the nonlocal variational approach to mesh fairing and develop a new mesh smoothing method which solves a fourth order surface diffusion equation on $\mathcal{M}$.

Nonlocal surface diffusion flow (NL-SDF)

Replacing X with the mean curvature normal vector $\overrightarrow{H}$ in (2.25), and considering a uniform discretization of the time interval $[0, T]$, $T > 0$, with

a temporal time step τ, then (2.28) can be fully discretized using a variety of explicit or implicit time integration schemes. In our computational method, we used the forward Euler scheme which yields a first order scheme in time. Therefore, applying an implicit scheme to (2.28), without the fidelity term, we get the iterative scheme

$$(I - \tau L_w)\overrightarrow{H}_i^{n+1} = \overrightarrow{H}_i^n, \quad \overrightarrow{H}|_{t=0} = \overrightarrow{H}^0, \tag{2.29}$$

where L_w is computed as given in (2.27), with initial condition $\overrightarrow{H}^0$ determined from X_0.

The number of time iterations n is chosen by the user; from our experimental work we tuned up $n_{MAX} \leq 20$.

For image processing the weight function is defined by image features and represents the similarity between two pixels, based on features in their neighborhood, see [18]. Working with surfaces, the way of choosing weight W_{ij} in (2.27) should characterize the similarities between two local surface patches. We propose to use the mean curvature values. Therefore, according to (2.21), we define the weights as follows

$$\begin{array}{rl} W_{ij} = & \frac{1}{\sum_{j\in N(i)} W_{ij}} e^{-D(X_i,X_j)/\sigma}, \\ D(X_i, X_j) = & \|H(X_i) - H(X_j)\|_2^2, \quad j \in N(i). \end{array} \tag{2.30}$$

The parameter σ controls the decay of the exponential function and therefore the decay of the weights as function of the Euclidean distance between mean curvature values. Since $H(X)$ is normalized to one, we can fix the value for σ in order to identify a significant change in the curvature between vertices X_i and X_j. For example, for $\sigma = 0.1$ we identify as a curvature change when the mean curvature values in the two vertices differ more than 10%. The use of smaller σ leads to the detection of sharper features.

The two-step strategy first smooths the normal vectors allowing the mean curvature normals to diffuse on M, then the second step refits the parameterization X according to a given mean curvature distribution. The mean curvature smoothing (2.29) is "nonlocal". By this we mean that a "nonlocal" operator is used which includes weights that penalize the similarity between patches.

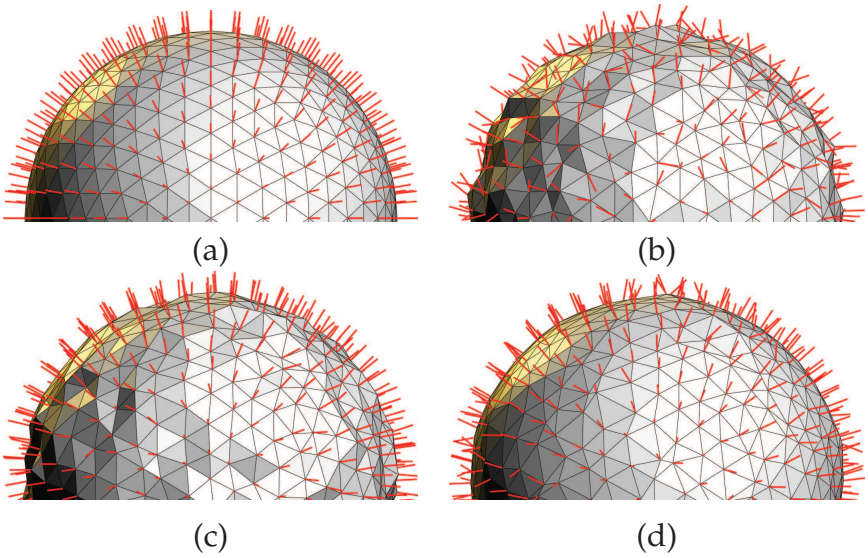


Figure 2.1: (a) The noise-free sphere mesh; (b) the perturbed sphere; (c) the smoothed mean curvature vector field obtained by step 1; (d) reconstructed sphere by step 2.

The nonlocal approach is described by the following algorithm, where in step 1 we solve (2.29) by a sequence of linear systems, the smoothed mean curvature normal vector field is then plugged into the constrained least square problem in step 2. Here L is defined by (1.16) and L_w as in (2.27).

Non Local SDF Algorithm
Given an initial position vector X_0,
STEP 1: SOLVE FOR H:
For each $n = 1, \cdots, n_{MAX}$
$(I - \tau L_w)\overrightarrow{H}^{n+1} = \overrightarrow{H}^n$
end for
STEP 2: PLUG IN $\overrightarrow{H}$ AND SOLVE FOR X:
$min_X \|LX - \overrightarrow{H}\|_2^2 + \lambda \|X_0 - X\|_2^2$

Fig. 2.1 shows how the two step NL-SDF algorithm works. A noise-free sphere mesh together with the associated mean curvature normal field is shown in (Fig. 2.1 (a)). The mesh is perturbed by a randomly chosen noise vector field. The perturbed sphere is illustrated in Fig. 2.1(b). The smoothed mean curvature vector field obtained by applying 10 iterations of step 1 is shown in Fig. 2.1 (c), while the recovered sphere resulting

from applying step 2 using the smoothed normal vector field, is shown in Fig. 2.1 (d).

We shall assume a certain level of connectivity in the mesh such that there will not be any disjoint regions where no information is exchanged between them throughout the evolution. Thus we assume that M consists of only one connected mesh. The matrix L has $rank(L) = N_v - k$, where k is the number of connected components of M, and it is positive semi-definite. Since we imposed that M is connected, that is $k = 1$, then L has a zero eigenvalue with multiplicity 1. The linear system derived from solving step 2 is uniquely solvable by fixing a vertex to have an assigned value.

When the perturbation on the initial mesh affects only the magnitude of the normal field, that is $\overrightarrow{E}$ in (2.1) are in the normal directions, we can replace $\overrightarrow{H}$ with H in step 1 and step 2, thus processing the mean curvature scalar field instead of the mean curvature normal vector field.

In the following we theoretically justify the NL-SDF algorithm, which approaches to the solution of a fourth-order PDE representing a nonlocal surface diffusion flow on $\mathcal{M}$.

Let us suppose that the weight functions $W(x, y)$ are defined as in (2.30), and $\lambda = 0$. Then the sequence $\{X^{(n)}\}$, generated by the NL-SDF algorithm is convergent to the solution X^* of the fourth order **Non Local Surface Diffusion Flow (NL-SDF)** on $\mathcal{M}$

$$\frac{\partial X}{\partial t} = \triangle_{w\mathcal{M}} H(X), \quad X(0) = X_0, \tag{2.31}$$

where $\triangle_{w\mathcal{M}}$ is a nonlocal Laplace Beltrami operator, and M is the piecewise linear representation of $\mathcal{M}$.

Theorem 1 *Let* $\overrightarrow{V}(X) := \Delta_{\mathcal{M}} H(X) N(X)$. *Discretizing (2.31) in time, with time-step* τ_n, *we get*

$$X^{n+1} = X^n + \tau_n \overrightarrow{V}^{n+1}(X). \tag{2.32}$$

Solving the system of two second order PDE

$$\begin{aligned} &\tfrac{\partial \overrightarrow{H}}{\partial t} = \triangle_{\mathcal{M}} \overrightarrow{H}(X), \\ &\triangle_{\mathcal{M}} X = \overrightarrow{H}, \end{aligned} \tag{2.33}$$

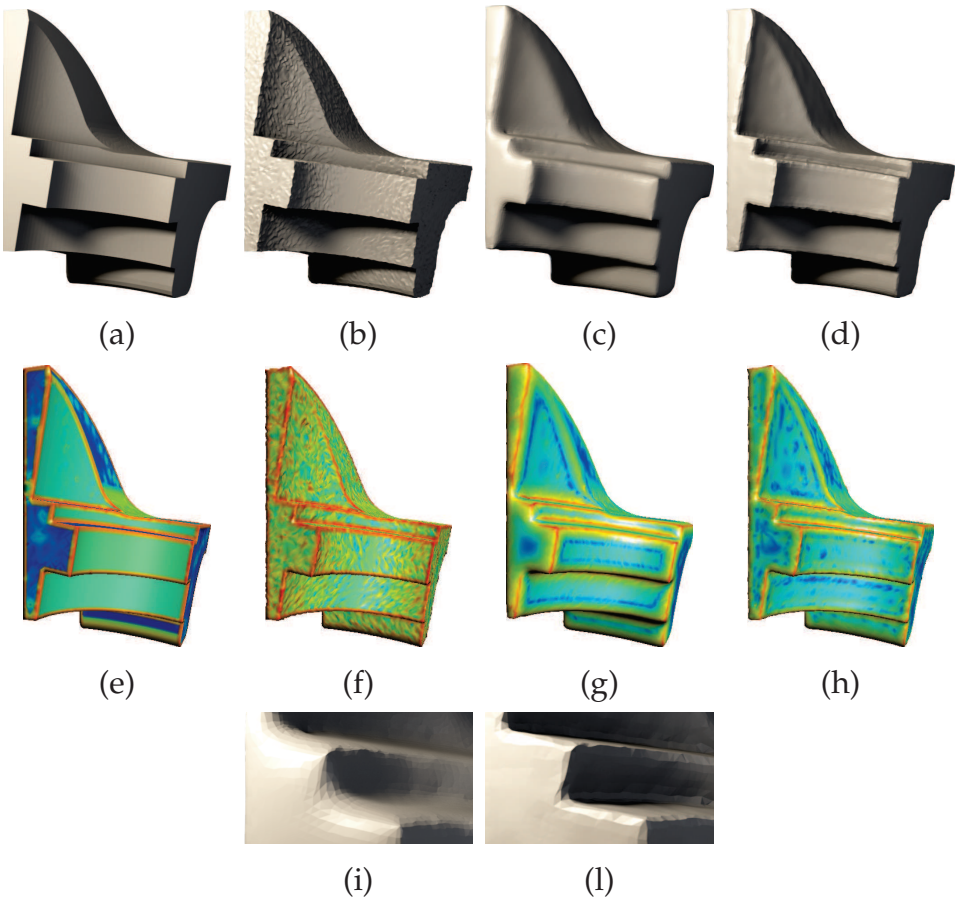
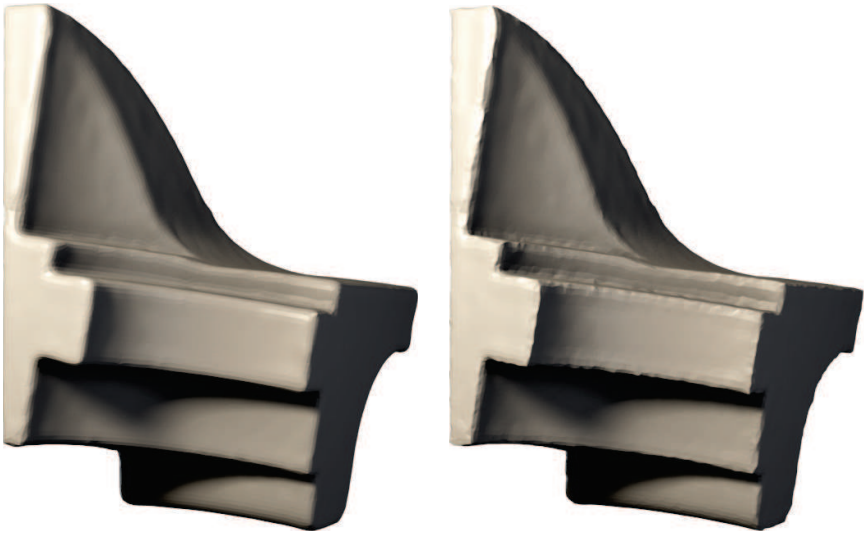


Figure 2.2: `fandisk` mesh: (a) Noise-free mesh and its curvature map (e); (b) noisy mesh and its curvature map (f); (c) restored mesh by the two step SDF and its curvature map (g); (d) restored mesh by NL-SDF algorithm and its curvature map (h); (i) and (l) zoomed details from (c) and (d), respectively.

that is, evaluating the corresponding discretizations

$$\begin{aligned} \frac{\overrightarrow{H}^{n+1}-\overrightarrow{H}^{n}}{\tau_n} &= \triangle_{\mathcal{M}} \overrightarrow{H}^{n+1} \\ \triangle_{\mathcal{M}} X^{n+1} &= \overrightarrow{H}^{n+1}, \end{aligned} \tag{2.34}$$

produces a sequence of iterates $\{X^{n+1}\}$ *which converges to the solution of (2.32), that is to the solution* X^* *of the fourth order* **surface diffusion flow (SDF)** *on* $\mathcal{M}$ *(2.31).*



(a) (b)

Figure 2.3: `fandisk` mesh: (a) restored mesh by BLF; (b) restored mesh by NL-SDF.

Proof 1 *Replacing X given by (2.32) in $\overrightarrow{H} = \triangle_{\mathcal{M}} X$ at time step n, we have*

$$\overrightarrow{H}^{n+1} = \triangle_{\mathcal{M}}(X^n + \tau_n \overrightarrow{V}^{n+1}(X)),$$

$$\overrightarrow{H}^{n+1} = \triangle_{\mathcal{M}} X^n + \tau_n \triangle_{\mathcal{M}} \overrightarrow{V}^{n+1}(X),$$

that is

$$\overrightarrow{H}^{n+1} - \tau_n \triangle_{\mathcal{M}} \overrightarrow{V}^{n+1}(X) = -\triangle_{\mathcal{M}} X^n$$

which leads to the first equation in (2.34). To relate position X and curvature $\overrightarrow{H}$, we recall from basic differential geometry the relation

$$\overrightarrow{H}^{n+1} = -\triangle_{\mathcal{M}} X^{n+1}, \tag{2.35}$$

which is the second equation in (2.34). □

The nonlinear parabolic PDE (2.33) can be interpreted as a diffusion flow for the vectors $\overrightarrow{H}_i$. The unknown mean curvature vectors at the vertices are determined by an implicit scheme which leads to a non-singular linear system $(I - dtL_w)$, with the matrix L_w defined as in (2.27) that discretizes

Mesh	Faces	Vertices	Volume	c_1	c_2
fandisk	51784	25894	0.234024	-0.6	-0.6
oilpump	82176	41090	0.184494	-0.8	-0.8
igea	268686	134345	0.376882	0.8	0.8

Table 2.1: Data for the meshes used in the examples.

$\triangle_{w\mathcal{M}}$. The computed mean curvature normals $\overrightarrow{H}_i, i = 1, \ldots, N_v$ are then used to move each vertex ith, according to the well known relation (2.35).

On the other hands, considering the similarity weights $W_{ij} = 1, \forall i, j$, and $\lambda = 0$, then the NL-SDF algorithm approaches to the solution of the Surface Diffusion Flow (SDF): $\frac{\partial X}{\partial t} = \triangle_{\mathcal{M}} H(X)$.

Moreover, if $\mathcal{M}(t)$ is a closed surface then the volume of the bounded domain computed by both NL-SDF and SDF is preserved.

In [94] the two step method is applied to solve the elliptic fourth order PDE $\triangle_{\mathcal{M}} H = 0$. A pioneer approach to the two-step denoising procedure with a fourth order model is introduced in [66]. A level set formulation of a two step geometric denoising via normal maps is also presented in [98].

Fairing Results

The results of the proposed algorithm are demonstrated applying perturbations to the meshes shown in Fig. 2.2(a), Fig. 2.4(a) and Fig. 2.6(a). The meshes present different characteristics in terms of details, "sharpness", and level of refinement, as summarized in Table 2.1.

The meshes are corrupted by adding a perturbation vector $\overrightarrow{E_i}$ for each vertex i of the mesh according to (2.1). We let $\overrightarrow{E_i}$ be a weighted sum of the normal vector $\overrightarrow{N(X_i)}$, and a random-direction unitary vector $\overrightarrow{v}$,

$$\overrightarrow{E_i} = \frac{c_1 \overrightarrow{N(X_i)} + c_2 \overrightarrow{v}}{\bar{e}}, \quad c_1, c_2 \in [-1, 1], \tag{2.36}$$

where $\bar{e}$ is a scaling factor determined by the edge length average of the mesh, and c_1 and c_2 are assigned scalar parameters that control the max-

Mesh	Algorithm	τ	n_{MAX}	$\Delta V(\%)$	σ
fandisk	MCF	0.013	10	8.11×10^{-4}	-
fandisk	SDF	0.139	10	0.89×10^{-4}	-
fandisk	NL-SDF	0.139	10	0.92×10^{-4}	0.6
oilpump	MCF	0.013	10	5.93×10^{-4}	-
oilpump	SDF	0.077	10	0.32×10^{-4}	-
oilpump	NL-SDF	0.022	10	0.32×10^{-4}	0.5
igea	MCF	0.146	20	3.90×10^{-4}	-
igea	SDF	0.141	20	0.02×10^{-4}	-
igea	NL-SDF	0.141	20	0.02×10^{-4}	0.6

Table 2.2: Data for the examples shown in Fig. 2.2, 2.4, 2.5 and 2.6.

Figure 2.4: `oilpump` mesh: (a) noise-free mesh; (b) noisy mesh

imum length of the corresponding vectors.

The amount of noise added to the meshes is then controlled by parameters c_1 and c_2, whose values are reported in Table 2.1. The perturbed versions of the meshes in the examples are shown in Fig. 2.2(b), Fig. 2.4(b) and Fig. 2.6(b).

Table 2.2 summarizes the experiments illustrated in this section. We compared the performance of the proposed NL-SDF method with MCF and SDF algorithms. The parameter λ for the fidelity term in step 2 of the algorithm is set to be 0.5. In Table 2.2 for each mesh (first column), the algorithm applied is shown in the second column, the corresponding time step used (τ) is provided in the third column, while the number of iteration steps (n_{MAX}) is in the fourth column. The differences in volume are labeled by $\Delta V\%$, and σ is the parameter in the weight functions (2.30).

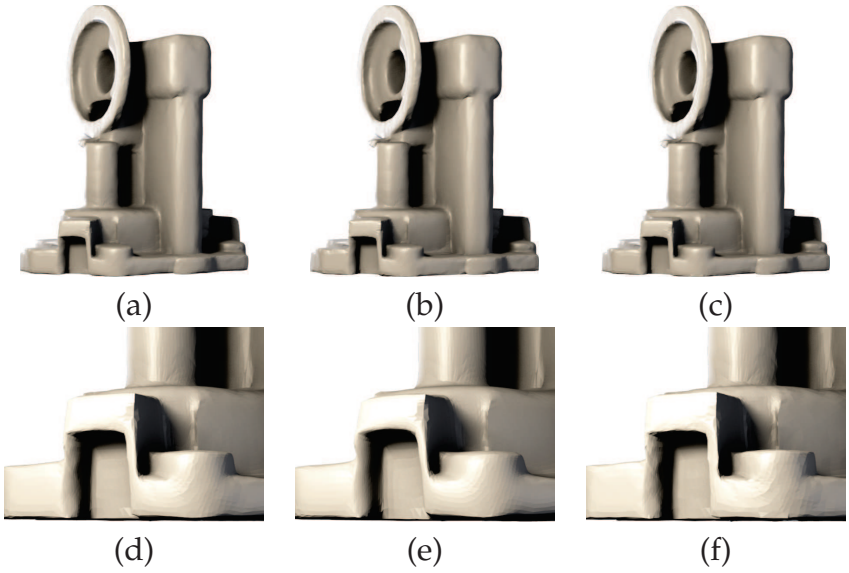


Figure 2.5: `oilpump` mesh: (a) restored mesh by the MCF algorithm; (b) restored mesh by the SDF algorithm; (c) restored mesh by NL-SDF algorithm; (d), (e) and (f) zoomed details from (a), (b) and (c), respectively.

The three models compared NL-SDF, MCF and SDF are all discretized by implicit schemes to avoid stability conditions on the time step. The time step τ for the iterative process is automatically chosen using Eq. 1.45 in order to produce a good quality denoised mesh using about 5 to 20 iterations, independently on the mesh characteristics or the Laplacian weights in (1.16).

In Fig. 2.2 and Fig. 2.5 we compare the recovered `fandisk` and `oilpump` meshes by applying algorithms MCF, SDFand NL-SDF. In Fig. 2.2, second row, by false colors we represented the value of the norm of the mean curvature vector associated to each vertex of the corresponding mesh in the first row. In Fig. 2.6 we compare the recovered `igea` meshes by applying algorithms MCF, BLF and NL-SDF. The example shown in Fig. 2.6 demonstrates that the proposed method can produce better results even on more naturally smooth meshes. From a visual inspection of Fig. 2.2 and Fig. 2.5, we can observe that, while the SDF and MCF algorithms well accomplish the task of denoising the surface, they fail in distinguishing the edges and sharp corners from the noise. The NL-SDF algorithm clearly enhances sharp features of the object while removing the noise in

the flat areas. This is also noticeable if the NL-SDF result is compared with the Bi-Laplacian flow result as in 2.3. The overhead of computational effort for NL-SDF with respect to SDF, is negligible and it consists in computing the weights W_{ij} in (2.30). The superiority of the NL-SDF method can be better appreciated in the more detailed and sharp areas of the mesh, where the features are reconstructed preserving the sharpness of the original noise-free mesh.

In Table 2.2 we labeled by $\Delta V(\%)$ the difference between the volume of the noise-free mesh (see Table 2.1, column marked by *Volume*), and the volume of the restored mesh. The NL-SDF algorithm ensures that the volume of the mesh is preserved after each smoothing iteration.

Numerical experiments seem to confirm that our algorithm is promising. We plan to extend the variational framework to general weighted operators.

2.2 Remeshing

Remeshing refers to the improvement process of the mesh quality in terms of redistribution of the sampling, connectivity of the geometry, and triangle quality, in order to satisfy mesh property requirements while maintaining surface features.

Some remeshing techniques are parameterization-dependent, i.e. they associate the mesh with a planar parameterization, and apply the algorithms on this plane. For arbitrary genus objects, this involves also the creation of an atlas of parametrization, a well known complex process that inevitably introduces some metric distortion and may lead to the loss of important feature information [4],[2].

In contrast, parameterization-free methods avoid these problems by working directly on the surface mesh and performing local modifications on the mesh. A parameterization-free method has been proposed in [12] for directly remeshing using area-equalizing weights in multiresolution modeling, and in [74], several tangential velocity strategies are introduced to regularize geometric surface flows.

Isotropic remeshing methods based on Centroidal Voronoi Tessellation

Figure 2.6: `igea` mesh: (a) Noise-free mesh; (b) noisy mesh; (c), (d) and (e) the MCF, BLF and SDF-NL restoration respectively.

(CVT) require to repeatedly compute a geodesic Voronoi diagram which is a complex and time-consuming step in this approach [2]. Several interesting proposals have been presented for this type of methods, both parameterization-based strategies [3], which compute the CVT on the 2D parametric domain, and parameterization-free methods, based on the intersection between a 3D Voronoi diagram and an input mesh surface, see [109]. An anisotropic remeshing method based on CVT for capturing sharp feature has been proposed in [61].

Another anisotropic remeshing approach for parametric surfaces where

the metric tensor is computed based on the surface curvature has been proposed in [95].

An interesting solution to the remeshing problem has been presented in [58] where polygonal surfaces are converted into meshes with subdivision connectivity through a simulation of the shrink wrapping process.

In [5] the remeshing methods are classified by their end goal rather than by the algorithmic strategy they employ. The techniques are classified into five categories: structured, compatible, high quality, feature and error-driven remeshing. The structured and compatible remeshing methods aim to obtain a given connectivity structure, the main goals for the high quality remeshing methods are the shape of the elements as well as the vertex distribution, while the end goal of feature remeshing is to preserve sharp features when producing the resulting meshes.

The proposed remeshing strategy is an adaptive, parameterization-free technique designed to produce a good compromise of high quality and feature remeshing techniques. High quality remeshing amounts to generating a mesh with well-shaped elements, uniform or isotropic sampling and smooth gradation sampling. Good quality elements mainly lead to minimizing numerical instabilities in subsequent computations. However, we relaxed the uniform sampling property in order to adapt the size of the elements to the underlying surface features.

The proposed remeshing algorithm alternates equalization of edge lengths and vertex valence, which generate a new connectivity, with mesh regularization, which modifies the distribution of the vertices on the surface to satisfy given mesh quality requirements. While the techniques that explicitly modify the connectivity, such as e.g. edge split, collapse, and flip, are widely used, the potential of the regularization step is still not much investigated.

We present a new method to regularize a triangle mesh M, which defines a piecewise linear approximation of a curved surface $\mathcal{M}$, with the purpose of having an accurate representation of $\mathcal{M}$: the density of the vertices should correlate with the regularity of $\mathcal{M}$. We cannot rely on parameterization to quantify regularity of $\mathcal{M}$ because this concept would not be invariant under reparameterization and furthermore we do not assume any parameterization is given. Therefore, we used the mean cur-

vature as a measure of regularity. To improve the regularization of the mesh M, the points $X(t)$ on the surface $\mathcal{M}$ are geometrically evolved using a tangential flow

$$\frac{\partial X}{\partial t} = \gamma \overrightarrow{T}, \tag{2.37}$$

where γ is the velocity in the tangent direction $\overrightarrow{T}$. The new resulting sampling adapts itself to the sharper features of the surface. This motivates us to name the proposed method adaptive remeshing (AR).

In order to satisfy high quality remeshing we investigate the design of tangential velocities that aim to keep all edges on element stars approximately of the same size and all areas proportional to the surface features. To this aim, we use a two-step approach. First, we compute an area distribution function driven by a mean curvature map of the surface mesh. Then the mesh vertices are moved on the tangential plane to satisfy edge equalization and area distribution quality requirements. The process is iterated until a significant improvement in triangle shape is obtained.

2.2.1 Adaptive Mesh Regularization

The AR method alternates equalization of edge lengths and vertex valence, which generate a new connectivity, with adaptive mesh regularization, which modifies the distribution of the vertices on the surface.

In the following we focus on a new adaptive mesh regularization method, while the mesh connectivity regularization is briefly discussed in Section 2.1.5 since it is based on classical tools for meshes.

The mesh regularization method consists of a two-step PDE model. In the first step, the vertex area distribution function $A(X)$ defined on the mesh M with vertex set $X = \{X_i\}_{i=1}^{n_v}$, is diffused over the mesh, constrained by the mean curvature map. In the second step, the vertex position is tangentially relocated to obtain edges on element stars approximately of the same size, and all the vertex areas proportional to the surface features.

Let A_0 be the initial vertex area distribution function computed as the Voronoi area at each vertex on the mesh M, with vertex set X_0. Then in STEP 1, the vertex area distribution function $A(X)$ is diffused on $\mathcal{M}$ by

solving

$$\frac{\partial A}{\partial t} = \triangle_{\mathcal{M}}^{w_H} A(X), \quad A(0) = A_0. \tag{2.38}$$

In (2.38) the operator $\triangle_{\mathcal{M}}^{w_H}$ is the *weighted Laplace-Beltrami operator* discretized on the mesh M by the matrix L_{w_H} with elements

$$L_{w_H}^{ij} = \frac{1}{\sum_{j \in N(i)} w_{ij}} \begin{cases} -\sum_{j \in N(i)} w_{ij} W_{ij} & i = j \\ +w_{ij} W_{ij} & i \neq j, j \in N(i) \\ 0 & otherwise \end{cases} \tag{2.39}$$

The weight W_{ij} defined as in (2.30) depends on a similarity measure between the i^{th} and the j^{th} vertex, and it is defined in terms of mean curvature values H on the mesh M

The mean curvature attribute $H(X)$ in (2.30) tends to be dominated by the maximum curvature and consequently it is visually similar to it. We chose the mean curvature attribute to determine the characteristics of the underlying surface, rather than the Gaussian curvature attribute, since many shapes cannot be differentiated by Gaussian curvature alone.

The weights (2.30) used in (2.38) prevents the area diffusion in high curvature regions. The method tends to adapt the areas to the object features: high curvature regions will be covered by small area elements, while flat regions will be covered by faces with larger areas. Fig. 2.7 shows the benefit of the weights in STEP 1 of the Adaptive Remeshing (AR) algorithm. The result of applying a few time steps of (2.38) without the help of the weights (2.30) on the irregular initial mesh shown in Fig. 2.7(a), is illustrated in Fig. 2.7 (b), while the contribution in (2.38) of the weights (2.30) is shown in Fig. 2.7 (c). The area diffusion function is represented by false colors, red colors for big areas, blue color for small areas. Increasing the number of time steps, the diffusion of (2.38) without weights converges to a constant area all over the entire mesh.

In STEP 2 of the AR algorithm the vertex position X is updated, taking into account the resulting $A(X)$ area distribution obtained in STEP 1, by solving the following constrained curvature diffusion equation

$$\frac{\partial X}{\partial t} = \nabla_{\mathcal{M}}^{w_A} \cdot (g(|H(X)|) \nabla_{\mathcal{M}}^{w_A} X), \quad X(0) = X_0, \tag{2.40}$$

(a) (b) (c)

Figure 2.7: Area diffusion in AR STEP 1: (a) the flat shaded mesh; (b) vertex area distribution function without weights (2.30); (c) vertex area distribution function with weights (2.30).

where the function $g(\cdot)$, referred to as the diffusivity, is defined as

$$g(s) := \frac{1}{(1+s^{\alpha})}, \tag{2.41}$$

where $\alpha > 0$ is a small positive constant value. The geometric evolution driven by (2.40) constrains the movement of vertices with high mean curvature values, that is, belonging to sharp creases and corners.

At each vertex X_i, linearizing (2.40) by evaluating $g(|H(X_i)|)$ with X_i from the previous time-step, the right-hand side of (2.40) reduces to

$$g(|H(X_i^{old})|)\triangle_{\mathcal{M}}^{w_A} X_i^{new}. \tag{2.42}$$

We denote by L_{w_A} the discretization of the weighted Laplace Beltrami operator $\triangle_{\mathcal{M}}^{w_A}$ at vertex X_i defined as in (2.39) with w_{ij} given in (1.18) and the equalization weights W_{ij}^A defined by the sigmoid function

$$W_{ij}^A = \frac{1}{\sum_{j\in N(i)} W_{ij}^A} \frac{1}{1+e^{-f(X_i,X_j)/\sigma}} \tag{2.43}$$

$$f(X_i,X_j) = \lambda_1 \frac{\Delta A(X_j)-\Delta A(X_i)}{\overline{A}} + \lambda_2\left(\frac{E_{ij}-\overline{E_i}}{\overline{E}}\right), \tag{2.44}$$

where $\Delta A(X) = A(X) - A^d(X)$ is the offset between the vertex area $A(X)$

and the ideal vertex area $A^d(X)$ resulting from STEP 1, $\overline{A}$ is the mesh average vertex area, $\overline{E_i}$ is the local average edge length, and $\overline{E}$ is the mean of the mesh edge length. The coefficients $\lambda_i > 0$,$i = 1, 2$, sum up to 1 and determine how much respectively the area gap and the local edge length difference influence the movement of vertex X_i toward the neighborhood vertex X_j. In other words, considering only the area gap influence (i.e. $\lambda_1 = 1$), the vertex X_i is attracted by X_j when the area of X_i needs to grow more than the area of X_j. On the other hand, the vertex X_i does not move towards X_j when the area of both vertices do not need either to shrink or to grow or when they both need to shrink or grow by the same amount.

Finally, the displacement of the vertex X_i is in the tangent plane if we replace (2.40) with

$$\frac{\partial X_i}{\partial t} = (I - \overrightarrow{N}_i \overrightarrow{N}_i^T) g(|H(X_i)|) \triangle_{\mathcal{M}}^{w_A} X_i, \quad X(0) = X_0, \tag{2.45}$$

where $\overrightarrow{N}_i$ is the unit normal to the surface at X_i.

2.2.2 Adaptive remeshing (AR) algorithm

The AR algorithm iterates on the two stages of mesh connectivity regularization and adaptive mesh regularization approaching to a mesh with a smooth gradation of vertex density depending on mean curvature values and represented by well-shaped triangles.

We terminate the process and accept $X^{(i)}$ as the resulting mesh as soon as the difference of area variance between consecutive iterations is sufficiently small; specifically, we accept $X^{(i)}$ when for the first time

$$\Delta Var(A) := |Var(A(X^{(i)})) - Var(A(X^{(i-1)}))| < 1 \cdot 10^{-6}.$$

The regularization of mesh connectivity (named STEP 0) aims to perform an adjustment of edge lengths and vertex valences, and is implemented by applying the following basic tools:

1. specify target edge length $l \in [l_{min}, l_{max}]$
2. split all edges longer than l_{max}

3. collapse all edges shorter than l_{min}
4. flip edges to promote valence 6 (or 4 on boundaries).

These are commonly used tools to obtain mesh connectivity regularization, and we refer the reader to [12] for more details.

The adaptive mesh regularization stage is a two-step process which implements the PDEs (2.38) and (2.45), named in the sequel STEP 1 and STEP 2, to relax the vertex position according to a computed area distribution.

The following algorithm summarizes the computations required by our method.

Adaptive Remeshing Algorithm

Given an initial position vector X_0,
Compute $L_{w_H}(X_0), A(X_0)$, set $X^{(0)} = X_0$, i=1
While $\Delta Var(A) < 1 \cdot 10^{-6}$
  STEP 0: MESH CONNECTIVITY REGULARIZATION
  STEP 1: AREA REGULARIZATION:
  Set $A^{(0)} = A(X^{(i-1)})$
  $(I - \tau L_{w_H})A^{(i+1)} = A^{(i)}$
  Compute $L_{w_A}(X^{(i-1)})$, $g(|H^{(i-1)}|)$
  STEP 2: VERTEX TANGENTIAL UPDATE:
  Set $X^{(0)} = X^{(i-1)}$
  For $n = 1, \cdots, n_{MAX}$
    $X^{(n+1)} = (I + \tau(I - \overrightarrow{N}\overrightarrow{N}^T)g(|H^{(i-1)}|)L_{w_A})X^{(n)}$
  end for
  i=i+1
end while

Considering a uniform discretization of the time interval $[0, T], T > 0$, with a temporal time step τ, then (2.38) and (2.45) can be fully discretized using the forward Euler scheme which yields a first order scheme in time. We applied an implicit time scheme to (2.38), and an explicit time scheme to (2.45) with initial condition $A^{(0)}$ determined from X_0. From our experimental work we tuned up the maximum number of time iterations to be $n_{MAX} \leq 10$.

The tangential smoothing approach used in several remeshing algorithms is a simple Laplacian smoothing discretized as in (1.16) applied to the

three vertex coordinates components $X = (x, y, z)$ and then projected back into the tangent plane, see [108],[84]. Thus the tangential movement $\gamma \overrightarrow{T}$ in (2.37) at the surface vertex X_i is given by $(I - \overrightarrow{N}_i \overrightarrow{N}_i^T) L(X_i)$. In [12] the authors proposed an improvement, considering the tangential movement $\gamma \overrightarrow{T}$ in (2.37) at the surface vertex X_i as

$$(I - \overrightarrow{N}_i \overrightarrow{N}_i^T) L_w(X_i) \tag{2.46}$$

with

$$L_w(X_i) = \frac{1}{\sum_{j \in N(i)} A(X_j)} \sum_{j \in N(i)} A(X_j)(X_j - X_i), \tag{2.47}$$

and $A(X_j)$ represents the Voronoi area of vertex X_j. Vertices with large Voronoi area have a higher weight ('gravity') and attract other vertices, thereby reducing their own area. We call this method the Laplacian Flow (LF) scheme and we compare LF with our proposal in Section 2.2.3.

The LF scheme presented in [12] is integrated into an iterative remeshing procedure similar to the AR algorithm which alternates mesh connectivity regularization (like the STEP 0 in AR method) with the LF mesh regularization given in (2.47) (which is replaced by STEP 1 and STEP 2 in AR method).

Both the LF scheme and the STEP2 of the AR algorithm are discretized in time using explicit integration schemes. If we let τ satisfy the stability criterion for the diffusion PDE in (2.45), then $\tau \leq \frac{min(|e|)^2}{2}$, that is, it depends on the square of the smallest edge length, which is a very restrictive criterion involving an enormous number of integration steps [31].

We instead propose a "geometric" criterion on τ, which is defined by the formula

$$\tau = 0.1 \frac{\overline{E}}{\max_i(\|L_{w_A} X_i\|_2)}. \tag{2.48}$$

Using (2.48) each vertex will never be moved by a distance greater than 10% of the average edge length $\bar{e}$. Even if the time step computed by the geometric criterium is slightly larger than the τ obtained by stability requirements, in our computational experiments we always converged to an acceptable solution.

The implicit integration scheme used in the discretization of STEP 1 of

the AR algorithm does not suffer from numerical instability problems, anyway to avoid triangle flips τ is chosen such that:

$$\tau = 10\frac{\overline{A}}{\max_i(\|L_{w_H}A(X_i)\|_2)}.$$

This choice makes the remeshing procedure independent on the mesh area distribution.

2.2.3 Remeshing results

To our knowledge, there are no standard measures for evaluating and comparing the efficacy of remeshing techniques. Instead, comparison is often based on a qualitative evaluation of the final results and it is strictly related to the specific end goal the remeshing is used for. In [97] the authors measure the quality of a remeshed model by measuring the geometric properties of the resulting triangles, but this approach slightly limits the global overview on the benefits of the remeshing algorithm.

In our work, to assess the quality of the mesh generated by the proposed AR algorithm, we introduce the following measures:

Area Variance,

$$Var(A(X)) := \sum_{i=1}^{N_v}(A(X_i) - \overline{A})^2, \tag{2.49}$$

where $\overline{A}$ is the average vertex area for a mesh with N_v vertices,

Mean curvature variation,

$$\Delta H = \|H - H_0\|_2 / \|H_0\|_2, \tag{2.50}$$

where H and H_0 are the mean curvature maps of the remeshed and the original meshes, respectively, For each vertex X_i on the original mesh, the difference $H(X_j) - H_0(X_i)$ is computed with respect to the nearest vertex X_j on the remeshed mesh.

Variation of local edge length variance,

$$\Delta Var(E) = \|Var(E) - Var(E_0)\|_2 / \|Var(E_0)\|_2, \tag{2.51}$$

Table 2.3: Quality measurements for the remeshing tests.

Mesh	Method	# its	$\Delta H\%$	$\Delta Var(E)\%$	$\Delta Var(A)\%$	$\Delta Var(H)\%$	$\overline{d}_h$
hand	AR	5	17.8	77.2	61.2	5.9	0.002
hand	LF	5	40.3	78.4	51.6	47.1	0.003
gargoyle	AR	10	26.4	63.7	47.5	14.3	0.007
gargoyle	LF	10	69.0	81.6	48.0	59.7	0.016
fandisk	AR	20	4.7	51.7	55.2	0.8	0.005
fandisk	LF	20	30.3	74.4	31.0	5.9	0.013
igea	AR	20	12.9	57.3	41.4	5.2	0.004
igea	LF	20	27.5	64.9	41.5	34.3	0.005
foot	AR	10	13.1	92.8	65.4	3.7	0.007
foot	LF	10	28.2	93.0	62.7	22.5	0.008

where the local edge length variance of the vertex X_i is computed as
$Var(E)_i = \frac{1}{d_i}\sum_{j\in N(i)}(E_{ij} - \overline{E}_i)^2$.

Hausdorff distance,

$$d_{\mathrm{H}}(X_0, X) = \max\{\sup_{x\in X_0}\inf_{y\in X} d(x,y), \sup_{y\in X}\inf_{x\in X_0} d(x,y)\}, \tag{2.52}$$

is a measure of distance between the remeshed and the original meshes, see [23].

The goal of our remeshing strategy is to minimize the area variance ($Var(A(X))$), while preserving the mean curvature values and the shape of the original mesh, that is minimizing (ΔH) and the average one-sided Hausdorff distance $\overline{d}_h$. Moreover, the variation of mean curvature variance ($\Delta Var(H)$) is a value that should be preserved, and the variation of area variance ($\Delta Var(A)$) and the variation of edge variance ($\Delta Var(E)$) provide a measure of the quality of the resulting mesh.

The parameters α in the diffusion function $g(\cdot)$ in (2.41), and σ in (2.30) are chosen to be 0.05, while the parameter σ in (2.43) is 0.1.

We compare the AR algorithm with the LF method defined in [12], when applied to the meshes shown in Fig. 2.8(a), Fig. 2.10(a), Fig. 2.11(a), Fig. 2.12(a) and Fig. 2.13(a). In the reported experiments we omit the mesh

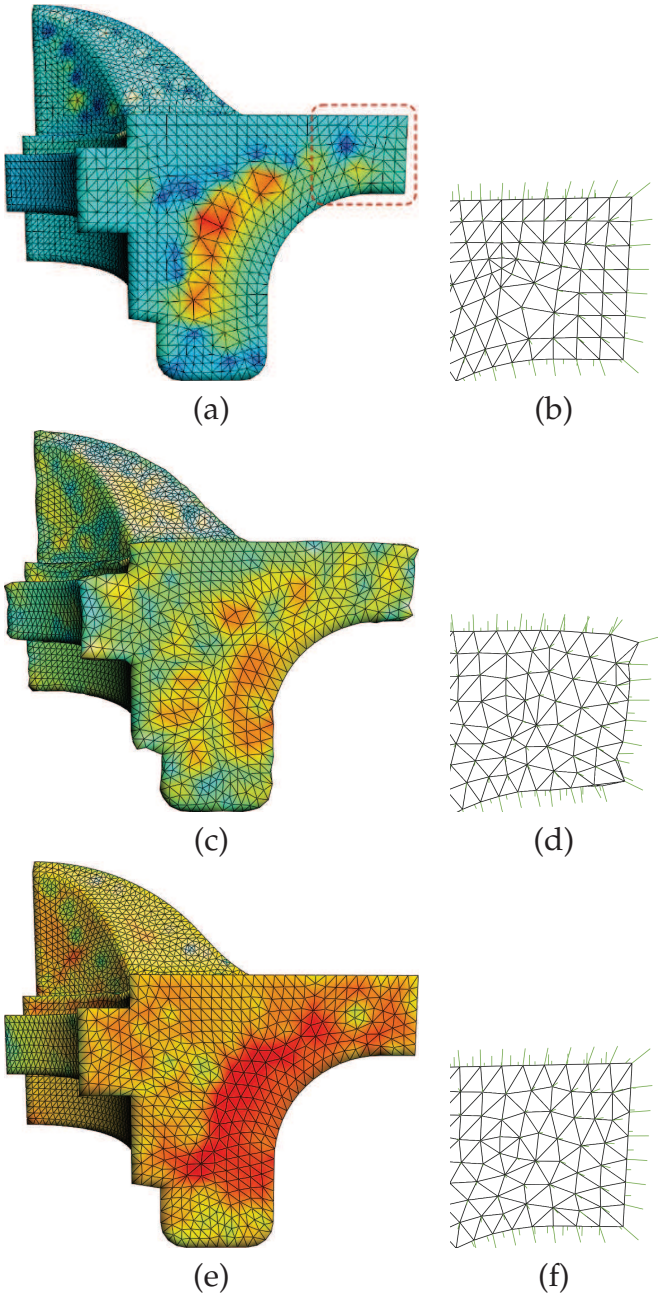


Figure 2.8: (a)-(b) the original mesh and a zoomed detail; (c)-(d) the result of LF and a zoomed detail; (e)-(f) the result of AR and a zoomed detail from the rightmost part of `fandisk` mesh.

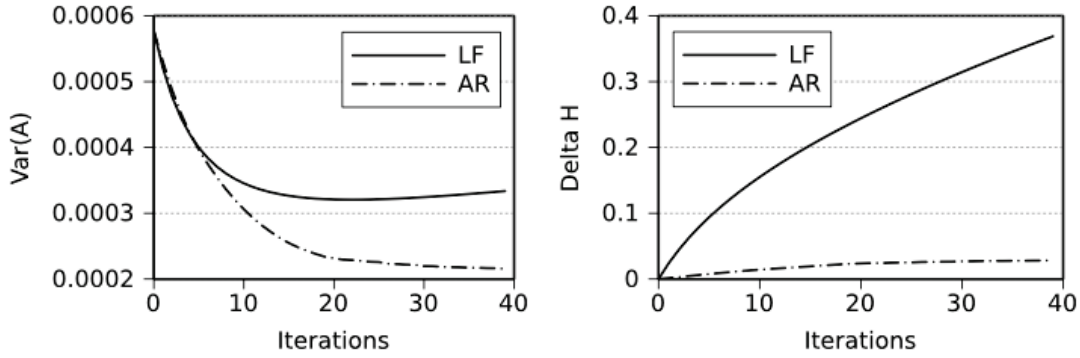
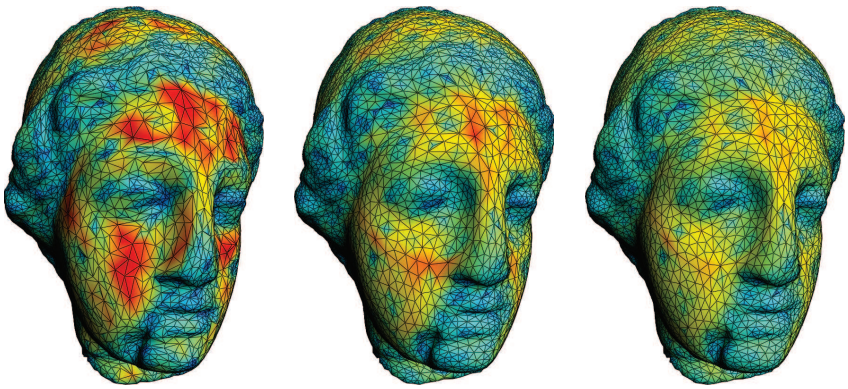
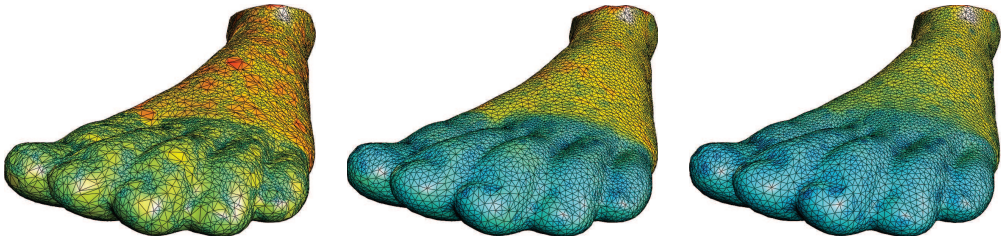


Figure 2.9: The area variance $Var(A)$ (left) and the mean curvature changing ΔH (right) as function of the remeshing iterations.



(a) (b) (c)

Figure 2.10: (a) the original mesh with area distribution function superimposed using false color; (b) LF remeshing (c) AR remeshing.



(a) (b) (c)

Figure 2.11: (a) the original mesh with area distribution function superimposed in false color; (b) LF remeshing and (c) AR remeshing.

connectivity regularization stage integrated in both AR (STEP 0) and LF methods in order to effectively point out the performances of the mesh regularization stage in the AR algorithm. Moreover, we noticed that the features of the mesh get easily compromised if the operations involved in the mesh connectivity stage are not performed adaptively, as we do in STEP 0 of the AR algorithm. In particular, in STEP 0 of the AR method the flip tool is applied only when the deviation in curvature normals is acceptable. This improves further on the results obtained by the AR method, but it makes more difficult to distinguish the benefit of the mesh regularization stage inside the entire iterative process.

A qualitative comparison is illustrated in Fig. 2.8, Fig. 2.10, Fig. 2.11, Fig. 2.12 and Fig. 2.13. In Fig. 2.8, Fig. 2.10, Fig. 2.11, and Fig. 2.12 the area distribution function A is visualized using false colors superimposed onto the meshes. In Fig. 2.13 the curvature map is superimposed on the `gargoyle` mesh using false colors where red colors represent high curvatures and blue colors low curvature values. Fig. 2.8 and Fig. 2.13 show two examples of applying our AR method to meshes with sharp features to highlight the weaknesses of the LF remeshing when applied to regions with high curvature values and high triangle density.

The `fandisk` mesh, illustrated in Fig. 2.8(a), presents a medium quality vertex area distribution, sharp edges and corners which allow us to demonstrate the capacity of our AR algorithm to adaptively distribute the vertex areas in zones of high curvature, preserving the sharp features of the mesh. The superiority of the AR approach w.r.t. the LF method can be visually appreciated in Fig. 2.8 comparing the resulting mesh by 20 remeshing iterations of LF algorithm (Fig. 2.8(c)) and the mesh obtained by 20 remeshing iterations of AR (Fig. 2.8(e)). A detail of the rightmost part (see the dashed rectangular box in Fig. 2.8(a)) is shown to enhance the area equalization and preservation of features obtained by the AR method.

In the `gargoyle` model of Fig. 2.13 the wings of the gargoyle are completely ruined by the LF method (see Fig. 2.13(b)) while they are well preserved by the AR method (Fig. 2.13(c)).

The irregular models illustrated in Fig. 2.10, Fig. 2.11 and 2.12 are characterized by a vertex area distribution particularly corrupted and many badly shaped triangles. In all the examples AR performs better than LF

in both the task of area distribution and mean curvature preservation.

Table 2.3 summarizes the quality measurements obtained by the illustrated experiments. In Table 2.3 for each mesh (first column) the applied algorithm is shown in the second column and the corresponding number of remeshing iterations (*its*) is in the third column. The results reported in Table 2.3 show that the AR method successfully produces well-shaped triangles while preserving the mean curvature map of the original mesh better than the LF method.

The plots in Fig. 2.9(left) and Fig. 2.9(right) show the area variance $Var(A)$ and the mean curvature variation ΔH as functions of the number of remeshing iterations when the LF method and the AR method are applied to the `fandisk` mesh. When the AR algorithm is applied, the area variance $Var(A)$ rapidly decreases, while the mean curvature of the mesh is preserved. The comparison with the LF method highlights a strong effect on the mean curvature of the resulting mesh and a non-convergent behavior in the minimization of the area variance. This aspect requires further theoretical investigation.

2.3 Deformation

The classical mathematical foundations of deformable models represent the confluence of geometry, physics, and approximation theory. Geometry serves to represent object shape, physics imposes constraints on how the shape may vary over space and time, and optimal approximation theory provides the mechanisms for fitting the models to measured data[67].

Deformable curve, surface, and solid models gained popularity after they were proposed for use in computer vision and computer graphics in the mid 1980s[101][102]. A deformable model that has attracted the most attention to date is popularly known as snakes[56]. Snakes are planar deformable contours that are useful in several image analysis tasks. In its basic form, the mathematical formulation of snakes draws from the theory of optimal approximation involving functionals.

We start from a similar mathematical formulation to derive an interactive deformation framework for 3D Meshes. For a more comprehensive

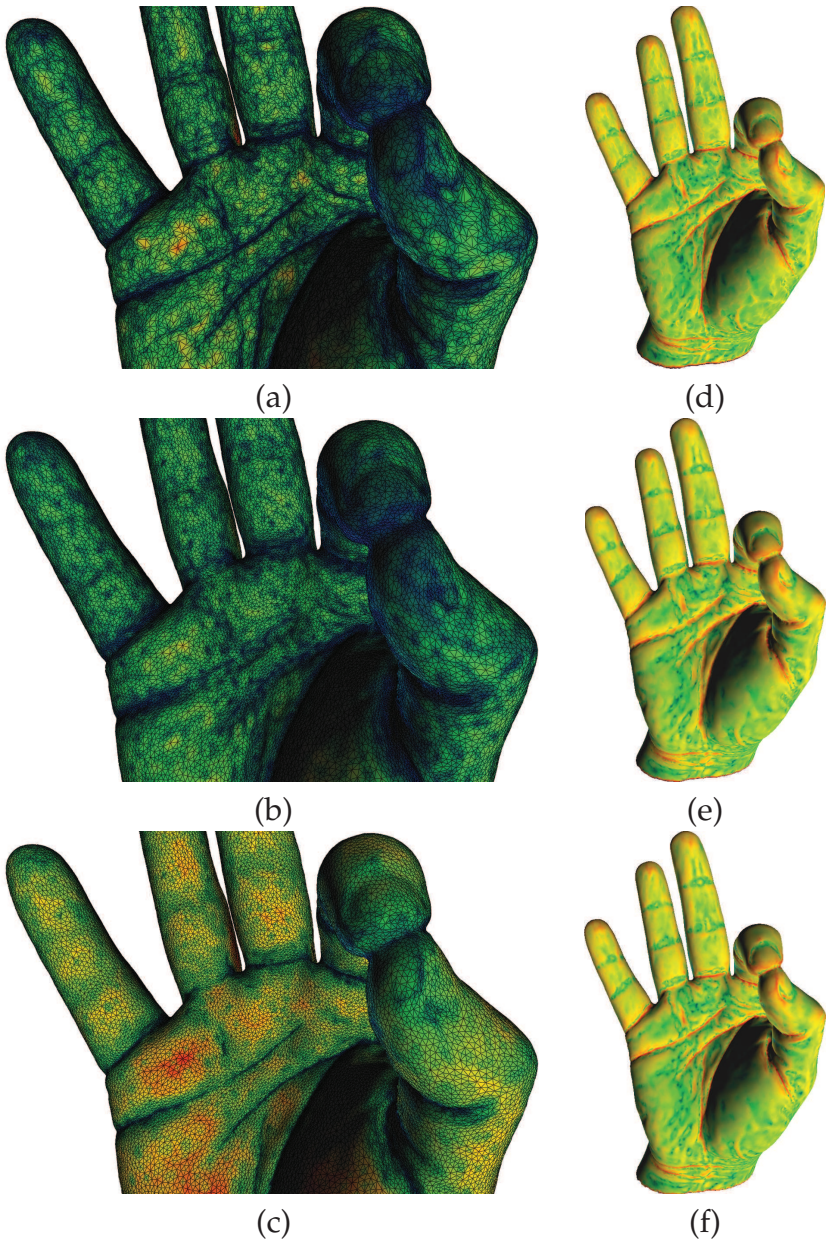
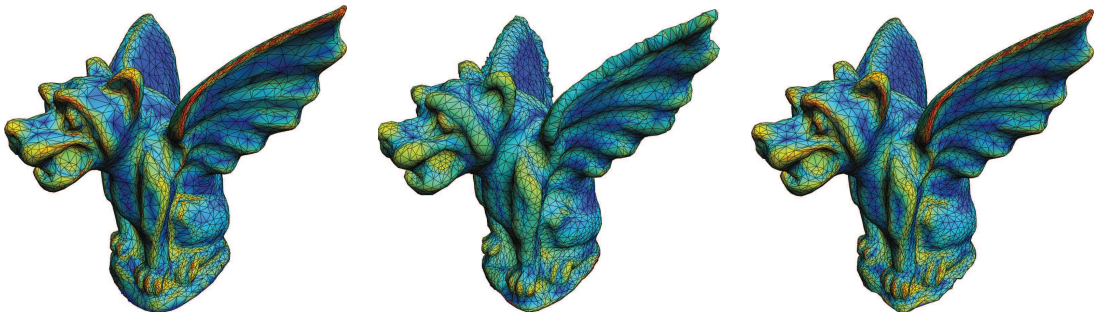


Figure 2.12: First column: area distribution function superimposed on the original mesh (a), the LF (b) and AR (c) remeshing respectively. Second column: the mean curvature map on the same meshes.



(a) (b) (c)

Figure 2.13: (a) the original mesh, (b) LF remeshing and (c) AR remeshing. Red represents high curvature, blue represents low curvature.

review on Lagrangian deformation models see [100] and [15].

The main requirement for physically based surface deformation is an elastic energy that measures how much an object has been deformed from its initial configuration. While for solid objects this energy basically considers local stretching within the object, for two-manifold surfaces (so called *thin-shells*) an additional bending term is required to approximate the change of surface curvature.

Let us denote by $\mathcal{M}$ a two-manifold surface, parametrized by a function $X : \Omega \subset \mathbb{R}^2 \to \mathcal{M} \subset \mathbb{R}^3$. This surface is to be deformed to $\mathcal{M}'$ by adding to each point $X(u, v)$ a displacement vector $d(u, v)$, such that $\mathcal{M}' = X'(\Omega)$, $X' = X + d$.

A standard measure used for the global surface quality in geometric modeling is the **thin plate energy**:

$$E_{BEND}(d) = \frac{1}{2} \int_{\Omega} d_{uu}^2 + 2d_{uv}^2 + d_{vv}^2 du dv, \tag{2.53}$$

which approximates, that is, it is a "linearized" version, of the **total curvature energy** of a surface $\mathcal{M}$ defined by (1.31), which is equal to (2.53) when the parametrization is isometric. The total curvature (1.31) approximates the elastic bending energy of a thin plate manifold.

The **membrane energy** is defined by the functional

$$E_{STRETCH}(d) = \frac{1}{2}\int_{\Omega} d_u^2 + d_v^2 du dv. \tag{2.54}$$

The minimization of these functionals can be performed efficiently by applying variational calculus, which yields their Euler-Lagrange which are

$$-(d_{uu}^2 + d_{vv}^2) = -\triangle d = 0, \tag{2.55}$$

for (2.54), and

$$d_{uuuu} + 2d_{uuvv} + d_{vvvv} = \triangle \circ \triangle d = \triangle^2 d = 0, \tag{2.56}$$

for (2.53), which involve Laplacian and bi-Laplacian operators.

In order to keep the parametrization of the surface $\mathcal{M}$ as close to isometric as possible, Ω is typically chosen to be equal to the initial surface $\mathcal{M}$, such that $d : \mathcal{M} \to \mathbb{R}^3$ is defined on the manifold $\mathcal{M}$ itself. As a consequence, the Laplace operator $\triangle$ w.r.t. the parametrization X turns into the Laplace-Beltrami operator $\triangle_{\mathcal{M}}$ w.r.t. the manifold $\mathcal{M}$, and the PDEs (2.55) and (2.56) becomes:

$$-\triangle_{\mathcal{M}} d = 0, \tag{2.57}$$

and

$$\triangle_{\mathcal{M}}^2 d = 0, \tag{2.58}$$

respectively.

Combining stretching and bending together, The variational deformation on the thin plate is then given by

$$-k_s \triangle_{\mathcal{M}} d + k_b \triangle_{\mathcal{M}}^2 d = 0, \tag{2.59}$$

where the parameters k_s and k_b are the stretching and bending factors, respectively. The order m of partial derivatives in the energy or in the corresponding PDE $(-1)^m \triangle_{\mathcal{M}}^m d = 0$ defines the maximum continuity C^{m-1} for interpolating displacement constraints. Hence, solving (2.59) provides C^1 continuous surface deformations.

Note that in surface smoothing similar functionals are applied to X itself

instead of their displacements.

The minimization of the total curvature energy

$$E(\mathcal{M}) := \frac{1}{2}\int_{\mathcal{M}} k_1^2 + k_2^2 d\mathcal{M}, \tag{2.60}$$

leads to the Euler-Lagrange equation

$$-\triangle_{\mathcal{M}} H(d) - 2H(d)(H^2(d) - K_G(d)) = 0, \tag{2.61}$$

which is a fourth-order partial differential equation, (the term $\triangle_{\mathcal{M}} H(X)$ involves fourth-order surface derivatives) satisfied for an elastica surface. The elastica flow has been proposed for surface fairing and repairing in [110].

In a modeling application, one can be interested either to a dynamic time dependent simulation, or directly to solve the rest state of the deformation process. The latter means solving the PDE subject to user-defined boundary constraints. This typically means to fix certain surface parts $\mathcal{F} \subset \mathcal{M}$, and to define displacements for the so-called handle (target) regions $\mathcal{H} \subset \mathcal{M}$. In an interactive application $\mathcal{M}'$ has to be recomputed by solving the PDE each time the user manipulates the boundary constraints, for instance by moving the handle region $\mathcal{H}$.

Considering both the energy and linear or nonlinear constraints represented by a generic $\Phi(X)$, the constrained deformation can be modeled by

$$min_d E(d) \quad \text{subject to} \quad \Phi(X). \tag{2.62}$$

2.3.1 Discretization

Considering that the displacement vector is $d = X' - X$ and the discretizations L of the Laplace-Beltrami operator, then the deformation models on the surface represented by the mesh M lead to the following linear systems

$E_{STRETCH}$ (2.57)	$Ld = 0$	$LX' = LX$
E_{BEND} (2.58)	$L^2d = 0$	$L^TLX' = L^TLX$
E_{COMBO} (2.59)	$k_sLd + k_bL^2d = 0$	$(k_sL + k_bL^2)X' = (k_sL + k_bL^2)X$
E_{TOTAL} (2.61)	$L^2d - 2LGd = 0$	$(L^2 + 2LG)X' = (L^2 + 2LG)X$
	$G = H^2 - K$	

Each of these models leads to a generic linear system $AX = b$ with a sparse $N_v \times N_v$ coefficient matrix.

The positional constraints may be either incorporated as **hard** or **soft** constraints.

Hard constraints are incorporated into the system by moving each column corresponding to a constraints vertex $X_i \in \mathcal{F} \bigcup \mathcal{H}$ to the right-hand side, and removing the corresponding rows from the system. This yields a non-zero right-hand side $b \in \mathbb{R}^{N_v - n}$, for n constraints, and leads to an $N_v - n \times N_v - n$ system $A_rX = b$ that is solved for the x,y,z components.

We should remark that the original mesh without the constrained vertices $\mathcal{F}$ and $\mathcal{H}$ is a reduced mesh with, in general, more than one connected components. The associated connectivity matrix should be a reduced rank matrix. However, the A_r matrix is not the connectivity matrix of such a reduced mesh since the elements of A_r are the same of the original connectivity matrix, that is computed on the entire mesh.

Note that hard positional constraints are preferred in classical editing tools where the exact position should be achieved, while in a sketch-based system soft constraints are actually advantageous, since they allow the user to place imprecise locations to hint the desire shape, but not specify it exactly.

In this case we forced the constrained vertices to lie in the exact prescribed location, thus eliminating them from the system, if instead we want to consider soft constraints, they are added as additional terms to the discrete energy functional

$$min_{X'} E(X') + \frac{\lambda}{2}||X' - C||^2 \tag{2.63}$$

with the Lagrangian parameter $\lambda \in \mathbb{R}^n$, and C is the vector of prescribed vertex positions.

Replacing the bending energy (1.32) in (2.63) and solving the minimization problem lead to

$$\begin{aligned} L^T L X' - L^T \delta + \lambda (X' - C) = 0 \\ (L^T L + \lambda I) X' = L^T \delta + \lambda C \end{aligned} \tag{2.64}$$

In matrix-vector form, using $L = D\bar{\mathrm{L}}$ and $D_{ii} = 1/A_i$, the solution of (2.64) for the new mesh vertices X', is given by solving the overdetermined system

$$\begin{bmatrix} \sqrt{D^{-1}} L \\ \mathbf{0} \quad \sqrt{\lambda} I_n \end{bmatrix} X' = \begin{bmatrix} \sqrt{D^{-1}} \delta \\ \sqrt{\lambda} C \end{bmatrix} \tag{2.65}$$

where $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix which requires a resorting of the rows of L, and $C \in \mathbb{R}^n$ is a vector of elements c_i for each of the n positional constraint. The system has dimension $(N_v + n) \times N_v$ and it is full rank, thus it has a unique solution in the least-squares sense

$$X' = (L^T D^{-1} L + \lambda I)^{-1} (L^T D^{-1} \delta + \lambda C). \tag{2.66}$$

In order to approach to interpolation of the constraints c_i, the parameter λ has to be chosen sufficiently large. However, the condition number of the matrix grows with λ, then a higher weight can cause numerical problems.

2.3.2 Nonlinear deformation

This deformation framework tries to preserve the orientation of the Laplacian vectors w.r.t. the global coordinate system, whereas in reality they should rotate with the deformed surface. A correct deformation should retains the local surface features, that is their relative orientation and possibly their size. Therefore, the local transformation T_i should be restricted to rotation and isotropic scaling which transform the differential representations of the input mesh X: $\widehat{\delta_i} = T_i \delta_i$, where δ_i is the Laplacian coordinate in the rest pose, and T_i transforms it into the deformed pose.

Introducing the local transformations, the deformed positions of the mesh vertices X' are then obtained by solving the following minimization problem:

$$min_{X'} E(X') + \frac{\lambda_1}{2} ||X' - C||^2 + \frac{\lambda_2}{2} ||LX' - \hat{\delta}(X')||^2 \tag{2.67}$$

The term $\hat{\delta}(X')$ is a nonlinear function of the vertex positions because it includes the effects of local rotations, thus (2.67) is a nonlinear least-squares problem, while (2.63) is a linear least-squares problem. We impose the nonlinear constraints on the set $X \setminus \mathcal{F} \cup \mathcal{H}$.

2.3.3 Solution method 1: alternating

Solution for (2.67) can be obtained by alternating two simple and more efficient least-squares steps which are respectively responsible for improving the estimation of the local transformations, and vertex positions. That is, we treat X' and T_i as separate variables and allows for alternating optimization:

- STEP 1: Fix X', find T_i using local shape matching,
- STEP 2: Fix T_i, find X' solving linear least-squares (2.67) problem.

2.3.4 Solution method 2: Inexact Gauss-Newton

We want to solve the nonlinear deformation problem using the inexact Gauss-Newton method.

Given a non-linear equation $r(x) = b(x) - Ax$, the unconstrained nonlinear least squares problem

$$min_x \frac{1}{2}\|r(x)\|_2^2,$$

is solved by the Gauss-Newton method as follows. First, it approximates $r(x)$ by a linear model in a neighborhood of a given point x_c:

$$\overline{r_c}(x) = r(x_c) + J(x_c)(x - x_c), \tag{2.68}$$

where J represents the Jacobian, then it solves the linear least-squares problem

$$min_x \|r(x_c) + J(x_c)(x - x_c)\|_2^2.$$

The Gauss-Newton algorithm at each iteration k, computes a correction

s^k to the current approximation x^k, as a solution of the LS problem

$$min_s \|r(x^k) + J(x^k)s^k\|_2^2, \quad x^{k+1} = x^k + s^k. \tag{2.69}$$

The Gauss-Newton algorithm iterates until convergence.

The inexact Gauss-Newton method for nonlinear least squares (Steinhaug 1985) solve by an iterative method the LS system of equations

$$J^T J s^k = -J^T r^k$$

derived from (2.69).

Let us consider first the nonlinear deformation constraints,

$$min_{X'} \frac{1}{2} \|LX' - \widehat{\delta}(X')\|_2^2. \tag{2.70}$$

Similarly to (2.68), we can linearize $r(x) = LX' - \widehat{\delta}(X')$, at a given iteration k, as follows

$$LX'^{(k+1)} - \widehat{\delta}(X'^{(k+1)}) \approx LX'^{(k)} - \widehat{\delta}(X'^{(k)}) + (J(X'^{(k)}) - L)(X'^{(k+1)} - X'^{(k)}) \tag{2.71}$$

with J Jacobian of $\widehat{\delta}$.

In [48] an approximation for (2.71) is proposed which uses the simplification $J = 0$ in (2.71) at each steps:

$$\begin{aligned} LX'^{(k+1)} - \widehat{\delta}(X'^{(k+1)}) &\approx LX'^{(k)} - \widehat{\delta}(X'^{(k)}) + (L - J(X'^{(k)}))(X'^{(k+1)} - X'^{(k)}) \\ &\approx LX'^{(k)} - \widehat{\delta}(X'^{(k)}) + L(X'^{(k+1)} - X'^{(k)}) \\ &= LX'^{(k+1)} - \widehat{\delta}(X'^{(k)}). \end{aligned} \tag{2.72}$$

Thus the approximation in (2.72) is accurate only when $\|J(X'^{(k)})\| \ll \|L\|$.

At each Gauss-Newton iteration k, $\widehat{\delta}(X'^{(k)})$ is known at previous step, and (2.72) is solved as a least-squares problem

$$min_{X'^{(k+1)}} \|LX'^{(k+1)} - \widehat{\delta}(X'^{(k)})\|_2^2. \tag{2.73}$$

The solution proposed for (2.70) applies (2.73) which is an approximation of (2.69) which uses the further simplification of $J = 0$ in (2.71).

To update $\widehat{\delta}(X'^{(k)})$ in (2.73) at each step we follow the two-phase procedure:

- (STEP 1) For each vertex X_i with $N(i)$ neighbors, solve for $\mu^i = (\mu_1^i, \mu_2^i, \ldots, \mu_{N(i)}^i)$:

$$\sum_{j \in N(i)} \mu_j^i ((X_j - X_i) \otimes (X_{j-1} - X_i)) = \delta_i, \tag{2.74}$$

  where δ_i are the Laplacian coords. before deformation, and $(X_j - X_i) \otimes (X_{j-1} - X_i)$ is the normal vector to the triangle X_i, X_j, X_{j-1} on X_i. The overdetermined linear system (2.74) can be represented in matrix-vector form as

$$A_i \mu^i = \delta_i \tag{2.75}$$

  where A has dimension $3 \times N(i)$ and solved by SVD method.
- (STEP 2) Plug the computed μ_j^i in

$$d_i(X') = \sum_{j \in N(i)} \mu_j^i ((X'_j - X'_i) \otimes (X'_{j-1} - X'_i)), \tag{2.76}$$

  with X' vertices of the deformed mesh M'. Since the μ^i are the same before and after deformation, we can show that $d_i(X) = R_i \delta_i$ for local rotations R_i. Finally the Laplacian coordinates are normalized as follows:

$$\widehat{\delta}(X_i) = \frac{\|\delta_i\|}{\|d_i\|} d_i. \tag{2.77}$$

Let us consider the more general nonlinear deformation model (2.67). The minimization of (2.67), where $E(X)$ is given by (2.61), is

$$\begin{aligned} (L^2 - 2LK_G + \lambda_1 I + \lambda_2 (L^T - J)L)X' &= (L^2 - 2LK_G)X + \lambda_1 C + \\ &+ \lambda_2 (L^T - J)\widehat{\delta}(X'), \end{aligned} \tag{2.78}$$

which is still in the nonlinear form $Ax = b(X)$. Using a linearization similar to (2.72), and a simplification $J = 0$, we can apply a Gauss-Newton procedure which solves at the step k, the least squares problem

$$min_{X'^{(k+1)}} \|AX'^{(k+1)} - b(X'^{(k)})\|_2^2, \tag{2.79}$$

where $A = L^2 - 2LK_G + \lambda_1 I + \lambda_2 L^T L$ and $b = (L^2 - 2LK_G)X + \lambda_1 C + \lambda_2 L^T \widehat{\delta}(X'^{(k)})$.

The algorithm for nonlinear deformation using inexact Gauss-Newton method is here summarized:

Algorithm: Nonlinear Deformation using Inexact Gauss-Newton Method
INPUT: the initial mesh with vertex set X_0,
OUTPUT: the deformed vertex set X
Set $X^{(0)} = X_0$, $\widehat{\delta}(X^{(0)}) = L(X_0)$, k=0
STEP 1: Compute for each vertex $X_i \in X_0$ the set μ^i by solving (2.74)
Repeat
  Solve the linear LS problem (2.79)
  STEP 2: Compute $\widehat{\delta}(X^{(k+1)})$ by solving (2.76) and (2.77)
  k=k+1
until $\|X^{(k)} - X^{(k-1)}\| < 1 \cdot 10^{-6}$

2.3.5 Deformation based on Differential surface representations

The main idea behind surface deformation approaches based on differential surface representations is to use a surface representation that puts the local differential properties in focus, and to preserve these differential properties under deformation, aspiring to obtain an intuitive, detail-preserving deformation result.

In Laplacian-based representation the surface is represented by the differential coordinates, obtained by applying the Laplacian operator to the mesh vertices, that is $\delta_i = \triangle_{\mathcal{M}}(X_i)$. In the continuous setting the surface mesh deformation is obtained by minimizing the energy

$$min_{X'} \frac{1}{2} \int_{\Omega} \|\triangle X' - \delta\|^2 dudv. \tag{2.80}$$

The Euler-Lagrange equation derived is

$$\triangle^2 X' = \triangle \delta. \tag{2.81}$$

Taking the input surface as the parameter domain, and considering the

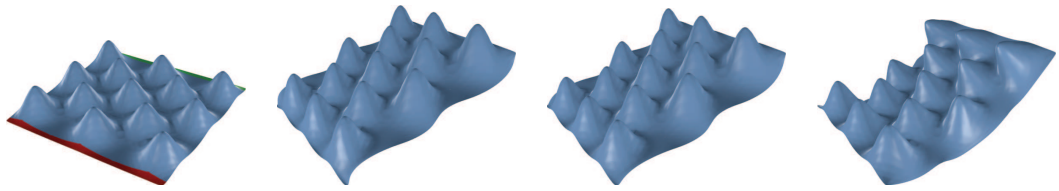


Figure 2.14: Deformation of the `bumpy-plane` mesh.

discretization of the Laplace-Beltrami operator, we get

$$L^2X' = L\delta, \tag{2.82}$$

which is constrained by the positional constraints $X_i = c_i$ for the fixed vertices in $\mathcal{F}$ and the handle vertices in $\mathcal{H}$. Note that (2.81) and (2.58) are equivalent, since $X' = X + d$. We can also obtain (2.82) by minimizing the discretization of the continuous energy (2.80):

$$min_{X'} \sum_i A_i |LX'_i - \delta_i|^2. \tag{2.83}$$

Eq. (2.83) forces each new vertex position to resemble its undeformed Laplacian as closely as possible, that is, in view of the fact that $\triangle_{\mathcal{M}} X_i = -H_i N_i$, where H_i is the mean curvature at X_i, it preserves the local curvatures of the undeformed mesh.

The minimization of (2.83) leads to the normal equations:

$$L^T D L X' = L^T D \delta, \tag{2.84}$$

with $D_{ii} = 1/A_i$, and using $L = D\overline{L}$, (2.84) can be rewritten as the bi-Laplacian equation (2.82).

2.3.6 Deformation results

Fig. 2.14, Fig. 2.15 and Fig. 2.16 show the result of some example deformations. In particular we considered the deformation models $E_{STRETCH}$ (2.57), E_{COMBO} (2.58) and E_{TOTAL} (2.61), the latter augmented with the Inexact Gauss-Newton Method presented in Section 2.3.4. The first column

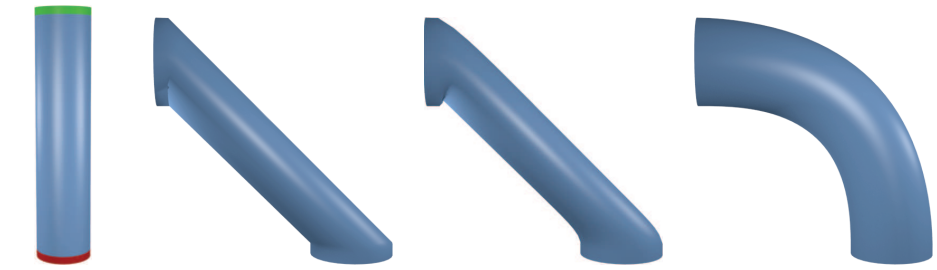


Figure 2.15: Deformation of the `cylinder` mesh.

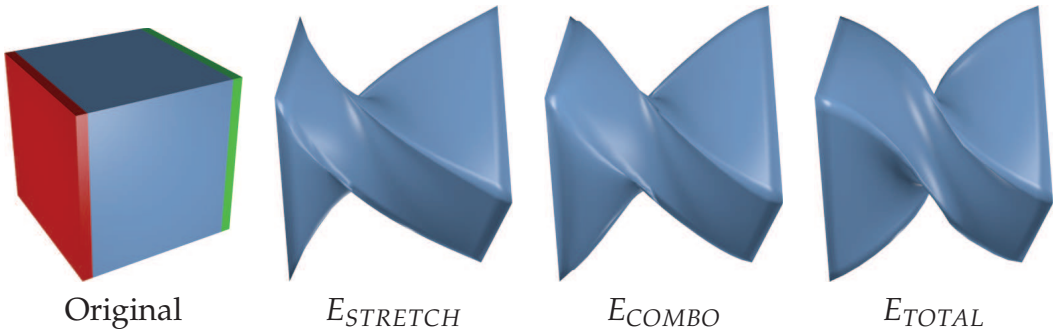


Figure 2.16: Deformation of the `cube` mesh.

of each figure shows the boundary constraints on the undeformed mesh: the red areas are the fixed constraints $\mathcal{F}$; the green areas are the handle (target) constraints $\mathcal{H}$; the blue area contains the remaining unconstrained vertices $X \setminus \mathcal{F} \cup \mathcal{H}$.

2.4 Simplification

A multiresolution mesh provides several mesh-based approximations of a 2-manifold representing the boundary of a solid object. The accuracy of the approximations is related to the mesh resolution, i.e., to the density (size and number) of its faces. Multiresolution meshes are a common basis for building representations of a large and complex object model at different levels of refinement. In a multiresolution modeling environment we need tools to coarsen a given fine mesh as well as tools for refining

a coarse mesh. Mesh refinement approaches range from multiresolution wavelet-based methods to subdivision schemes, where the quality of the generated mesh is dependent on the subdivision mask used. From fine irregular meshes, produced for example by 3D scanning devices, several approaches have been proposed to obtain a coarser/smoother mesh that meets user-defined quality criteria, i.e. mesh simplification, progressive meshes, and surface fairing. There are many applications for multiresolution meshes [24]. They are used to adjust the complexity of a geometric data set (i.e. oversampled 3D scan data). Differently complex models can be easily adapted to hardware with different capabilities. Furthermore, simplified models can be used in level-of-detail rendering, where the mesh resolution is proportional to the distance of the object to the camera [65].

We present a new approach to multilevel surface simplification based on the evolution of surfaces under p-Laplacian regularization. Such an evolution can be understood as a natural geometric filter process applied to an initial high resolution mesh, approximating a 2D-manifold, which leads to a coarse surface mesh of high approximation quality. This enables to reach different level of simplification, while preserving structural details. Simplification may be regarded as a cascadic iteration scheme, where one iteratively alternates between a spatial clusterization phase based on a weighted p-Laplace operator, and a decimation phase. Our proposal and its results is presented in [77].

The simplification approach thus combines the advantages of incremental decimation (arbitrary topology, local control and efficiency) with those of variational design (high quality surfaces).

The basic idea behind this strategy is to apply, at each simplification level, a spatial clustering diffusion flow to determine the potential candidates for deletion, followed by a decimation process to update the vertex mesh locations in order to decrease the overall resolution. We considered the minimization of energy which involves a regularization and a data fidelity in term of the initial mesh. The corresponding evolution problems would lead to p-Laplacian flow, which is a second order nonlinear Partial Differential Equation (PDE) problem. Moreover, to better preserve the surface features we introduce a weighted p-Laplacian strategy which involves curvature quantities. The numerical approximation of the varia-

tional p-Laplacian flow on the mesh is based on differential discrete operators following an implicit time discretization scheme, and one confines to a small number of diffusion steps within the corresponding iterative linear solver.

For the decimation step we followed the incremental quadric-based mesh decimation originally introduced by Garland in [41]. Instead of applying the quadric error metric both for sorting and for the computation of the new vertices for all collapsed edges, we propose to use the quadric error only to compute the optimal vertex position by combining it with the edge length penalty carried out by the spatial clusterization flow.

In the literature the different simplification approaches are basically classified into: vertex clustering algorithms, incremental decimation algorithms, and resampling algorithms [13]. The first class of algorithms is usually very efficient and robust, however, the quality of the resulting meshes is not always satisfactory and they can suffer from lack on the original topology. A popular vertex clustering method was proposed by J. Rossignac and P. Borrel [90], where the three dimensional space is divided into small cubes, the so-called cluster. Incremental algorithms in most cases lead to higher quality meshes, but suffer from large computational complexity overhead especially when a global error threshold is to be respected. Finally, resampling techniques select new samples more or less freely distributed over the original surface and construct a completely new mesh by connecting these samples [47]. The major advantage for resampling techniques is to satisfy special connectivity structure, i.e. subdivision connectivity (or semi-regular connectivity). In [6] a resampling method is proposed to compress triangulated surfaces, which is based on the PDE geometric diffusion equation where the finite element method is directly applied on the triangular mesh. A substantial improvement over classical decimation methods is the generation of progressive meshes during the decimation proposed in a paper by H. Hoppe [45] which is the first algorithm that employed the edge collapse operator. Most of the simplification algorithms use a distance metric to estimate the error which would result in the deletion of a vertex. A very popular choice is the quadric error metric, proposed by M. Garland [41], and a new quadric error metric described in [46]. The distance metric is very efficient in measuring geometric errors, but it has difficulties distinguish-

ing important shape features such as a high-curvature regions.metric. In [57] the authors suggest a discrete curvature norm to measure geometric error for such features.

2.4.1 The p-Laplacian flow

The variational p-Laplacian is defined by

$$\Delta_p f := \nabla \cdot (|\nabla f|^{p-2} \nabla f), \tag{2.85}$$

where $f : \Omega \to \mathbb{R}$ is a real function defined on the domain $\Omega \subset \mathbb{R}^n$, and usually $p \geq 1$. In the special case when $p = 2$, (2.85) reduces to the regular Laplacian operator $\Delta f = div(\nabla f)$, while for $p = 1$, we get $\Delta_1 f = \nabla \cdot (\frac{\nabla f}{|\nabla f|}) = -H$, where H is the mean curvature operator.

Needless to say, the p-Laplace equation with Dirichlet boundary conditions

$$\nabla \cdot (|\nabla f|^{p-2} \nabla f) = 0$$

is the minimizer in a domain Ω of the variational integral

$$J(f) = \frac{1}{p} \int_\Omega |\nabla f|^p \, d\Omega \tag{2.86}$$

among all functions in the Sobolev space $W^{1,p}(\Omega)$ satisfying the boundary conditions in the trace sense, where p is allowed to range over $1 < p < \infty$. We refer the reader to [82] for detailed note on the p-Laplace equation.

In image and surface processing contexts, a typical inverse problem considers the restoration of a function f^δ which represents an observation of an unperturbed function corrupted by noise. To recover the uncorrupted function the following form of regularization has proved to be successful:

$$\min_f \left\{ J(f) + \frac{\lambda}{2} \int_\Omega (f - f^\delta)^2 d\Omega \right\}, \tag{2.87}$$

where the first term is the regularization functional, the second integral is seen as a fidelity term, since f has required to be close enough to f^δ, and $\lambda \geq 0$ a regularization parameter. When $p \geq 1$ the energy in (2.87) is a convex functional, and there exists a global minimum for the minimizer

(2.87).

The Euler-Lagrange equation associated with (2.87), supplied with a gradient descent, which gives the solution of (2.87) as $t \to \infty$, is of the form

$$\frac{\partial f}{\partial t} = \triangle_p f + \lambda(f - f^\delta), \qquad f^0 = f^\delta, \tag{2.88}$$

where f^0 denotes the initial function. When $p = 1$, (2.88) gives the popular Total Variation (TV) model [21], a very popular choice in image processing for regularization, originally introduced for noise reduction [91], and also used for image deblurring [62] and super-resolution image reconstruction [81]. For $p = 2$, (2.88) gives the classical Tikhonov regularization. We are interested in the extension of these well known cases for p approaching to zero. We observe that for $p < 1$, $J(f)$ is nonconvex, and the global minimum of (2.87) is not insured. Nevertheless, this case is also considered in this paper since it leads to interesting p-Laplacian flows. The variational p-Laplacian evolution (2.88) is a continuous model, thus we have to discretize it in time and space in order to integrate it in the iterative process of clusterization and decimation as described in Section 2.4.2.

In the following we first address the problem of discretizing the p-Laplacian flow in time, then a good spatial discretization is proposed on a 2-dimensional manifold $\mathcal{M}$ without boundary which is approximated by a picewise linear triangular mesh M, and functions on the manifold are sampled at the vertices.

Semi-discretization in time

Considering a uniform discretization of the time interval $[0, T], T > 0$, with a temporal time step dt, then (2.88) can be semi-discretized using the forward Euler scheme which yields a first order scheme in time. An explicit time scheme applied to (2.88) gives

$$f^{t+1} = f^t + dt(\triangle_p f^t + \lambda(f^t - f^\delta)), \qquad f^0 = f^\delta. \tag{2.89}$$

For large problems sizes the explicit method is prohibitively slow since it may be nonrobust for relatively large time steps. The advantage of the

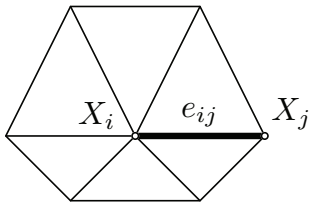


Figure 2.17: Vertex neighborhood stencil.

implicit time scheme given by

$$(I - dt\triangle_p)f^{t+1} = f^t + dt\lambda(f^t - f^\delta), \qquad f^0 = f^\delta, \tag{2.90}$$

is that the convergence rate does not depend on the size of the problem. We decided to follow the implicit time scheme (2.90) to avoid instability problems and to obtain better performances.

Fully discretization in space

Finite element and finite difference approximations of the p-Laplacian have been proposed in literature, see [82, 49] for some most recent works. Finite element discretization of the p-Laplacian results in highly nonlinear and degenerate algebraic systems [1, 64], therefore in this work we adopt a special finite difference scheme. In fact, since we are dealing with differential equation models over 2-manifolds $\mathcal{M}$ embedded in $\mathbb{R}^3$, the classical finite difference schemes will be replaced by discretized differential geometric operators over the mesh M which is a triangulated linear approximation to the manifold $\mathcal{M}$. The mesh M consists of a finite set of vertices $\mathbf{X}$, together with a subset $E \subseteq \mathbf{X} \times \mathbf{X}$ of edges. Functions on the manifold are sampled at the vertices X_i of a mesh: each point X_i has immediate neighbors X_j to which it is connected, see the stencil illustrated in Fig.2.17. Let us first address the problem of discretizing the p-Laplacian for a smooth function f for which we know the samples $f(X_i)$ at points of the set $\mathbf{X}$ in $\mathbb{R}^3$, with the mesh structure explained above. In the following we use a variant of the gradient and divergence definitions on graphs given in the context of machine learning [114, 115]. In our case, the weights are not normalized pointwise. Recently, in [16] and [36] the authors proposed a regularization framework on graphs which also uses similar operators.

At this aim we approximate the *directional derivative* or edge derivative of f at a vertex $X_i \in \mathbf{X}$ along an edge $e = (X_i, X_j) \in E$ with the difference operator $df(X_i, X_j)$:

$$\frac{\partial f}{\partial e}|_{X_i} \approx df(X_i, X_j) := \sqrt{w(X_i, X_j)}(f(X_j) - f(X_i)), \tag{2.91}$$

where $0 \leq w(x, y) < \infty$ is a positive measure defined between points x and y which is symmetric. The weigthed gradient $\nabla f(X_i)$ at a vertex X_i, is defined as the vector of all partial derivatives $df(X_i, X_j), \forall j \in N(i)$. Since for a given function $b(X) \geq 0$, the *divergence operator* may be written as

$$div(b(X_i)\nabla f(X_i)) = \frac{1}{2} \sum_{j \in N(X_i)} \frac{\partial}{\partial e_{ij}}(b(X_i)\frac{\partial f(X_i)}{\partial e_{ij}}), \tag{2.92}$$

replacing (2.91) and (2.92) in (2.85) we get a spatial discretization of the p-Laplacian operator

$$\begin{aligned} L_p^w f(x) &= \tfrac{1}{2} \textstyle\sum_{j \in N(X_i)} \sqrt{w(X_i, X_j)}(b(X_j)df(X_j, X_i) - b(X_i)df(X_i, X_j)) \\ &= \tfrac{1}{2} \textstyle\sum_{j \in N(X_i)} w(X_i, X_j)(b(X_j)(f(X_i) - f(X_j)) - b(X_i)(f(X_j) - f(X_i))) \\ &= \tfrac{1}{2} \textstyle\sum_{j \in N(X_i)} w(X_i, X_j)(b(X_i) + b(X_j))(f(X_i) - f(X_j)). \end{aligned} \tag{2.93}$$

Let $b(X) = |\nabla f(X)|^{p-2}$, if we define

$$\gamma(X_i, X_j) = w(X_i, X_j)(|\nabla f(X_j)|^{p-2} + |\nabla f(X_i)|^{p-2}) \tag{2.94}$$

then the weighted p-Laplacian (2.93) is approximated as

$$L_p^w f(x) = \frac{1}{2} \sum_{j \in N(X_i)} \gamma(X_i, X_j)(f(X_i) - f(X_j)). \tag{2.95}$$

Note that (2.95) reduces to the classical Laplace Beltrami discretization on mesh when $p = 2$ and the weights are the cotangential proposed in [69].

For the p-Laplacian discretization in case $p \leq 1$ we use a smoothed version $|\nabla f(X_j)|_{2,\epsilon}$ of $|\nabla f(X_j)|_2$ in (2.94). To this end, let us define $|v|_{2,\epsilon} := \sqrt{\sum_i |v_i|_\epsilon^2}$, with $|v_i|_\epsilon^2 := v_i^2 + \epsilon$ for any $v_i \in \mathbb{R}$ and $\epsilon > 0$, a small regularization parameter.

When f is a vector valued function $f : \mathbf{X} \subset \mathbb{R}^3 \to \mathbb{R}^3$, where $f(X) =$

$[f_1, f_2, f_3]^T$ and f_1, f_2, f_3 are the three vertex coordinate components (x, y, z) respectively, the p-Laplace operator is different for each component and in particular, γ represents a different weight for each component. Since vector-valued data needs to be driven by equivalent geometric attributes, we have to take the coupling between vector components into account, this is achieved by using the same vectorial variation defined by

$$|\nabla f(X)|_2 = \sqrt{\sum_{i=1}^{3} |\nabla f_i(X)|^2}. \tag{2.96}$$

Therefore, instead of having a component-wise value for γ in (2.95) we have the same value (2.96) in (2.95) for all components $f_i, i = 1, 2, 3$, thus obtaining a component coupling representation. The accuracy of the p-Laplacian discretization in (2.95) is $O(h)$, where h is the spatial resolution defined as the maximum length of the neighbors edges, $h_i = \max_{j \in N(i)} |e_{ij}|$. In practice, we obtain acceptable accuracy using a relatively narrow neighborhood. This motivates us to apply more time steps of the p-Laplacian flow for meshes with a relatively small h.

The effects of the p exponent for several values of the fidelity parameter λ in (2.88) are illustrated in Fig. 2.18 for $p = 2$, Fig. 2.19 for $p = 1$, and Fig. 2.20 for $p = 0.1$, and the color coded modulus of the mean curvature of the surface is depicted. On the top of Fig.2.18 the initial mesh is shown.

The full discretization of the functional (2.88) is obtained replacing (2.95) in (2.90),

$$(I - dtL_p^w) f^{t+1} = f^t + dt\lambda(f^t - f^\delta), \qquad f^0 = f^\delta. \tag{2.97}$$

The proposed spatial adaptive model

We consider a spatial adaptivity of λ in (2.88) in order to locally control the extent of diffusion over mesh regions according to their content. At this aim, we proposed to modify (2.97) to the following adaptive p-Laplacian regularization model

$$(I - dtL_p^w) f^{t+1} = f^t + dt\Lambda(f^t - f^\delta), \qquad f^0 = f^\delta. \tag{2.98}$$

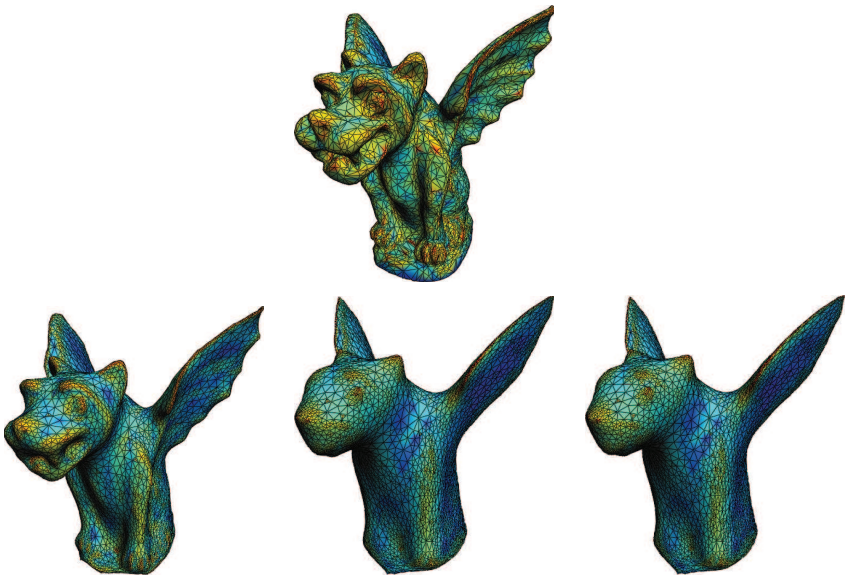


Figure 2.18: p-Laplacian flow with $p = 2$ and several λ values.

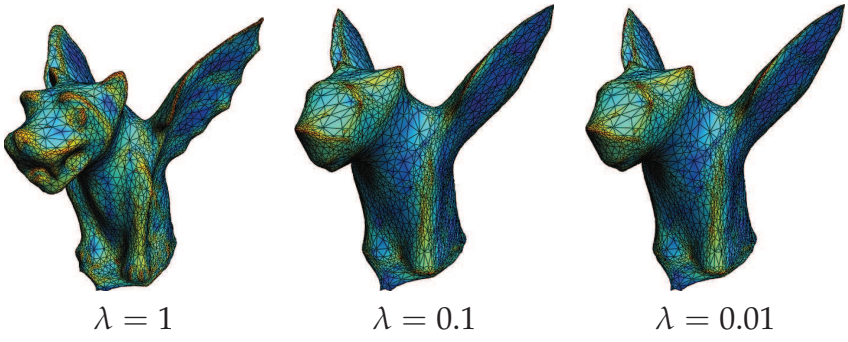


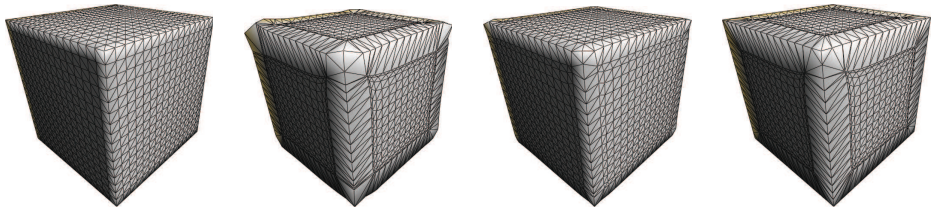
Figure 2.19: p-Laplacian flow with $p = 1$ and several λ values.

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix of order N, where N is the number of vertices, with λ_i representing the regularization parameter for the ith vertex. In order to enforce the fidelity to data which represent surface features, we define the $\lambda_i = \sqrt{H_i + \epsilon}$, where H_i represents the mean curvature associated to the vertex X_i.

Effects of spatial adaptivity are illustrated in Fig.2.21, where we applied the p-Laplacian flow (2.98) to the cube mesh shown in Fig.2.21(a). The

$\lambda = 1$ $\lambda = 0.1$ $\lambda = 0.01$

Figure 2.20: p-Laplacian flow with $p = 0.1$ and several λ values.



(a) original (b) $\lambda = 0$ (c) constant λ (d) adaptive λ

Figure 2.21: Spatial adaptivity effects.

benefit of the fidelity term is clear in Fig.2.21(b) where $\lambda = 0$, while applying a constant $\lambda = 0.1$ to every vertex of the cube mesh results in a mesh where the features are not well preserved, as illustrated in Fig.2.21(c). Finally, the adaptive Λ matrix allows for enforcing the fidelity at vertices representing significant local features, as shown in Fig.2.21(d).

The weights $w(X_i, X_j)$ in (2.94) incorporate a intercomponent information on the local shape of the mesh, we propose the following choices:

w1 $w(X_i, X_j) = 1$,
w2 $w(X_i, X_j) = 1/(\epsilon + |e_{ij}|)$,
w3 $w(X_i, X_j) = \frac{1}{\sum_{j \in N(i)} w(X_i, X_j)} e^{-\|H(X_i) - H(X_j)\|_2^2 / \sigma}$.

The parameter σ in **w3** controls how much the similarities of two local neighbors are penalized. Larger σ preserves sharper features. By using **w3** we get a good measurement of similarity, which penalizes in (2.98) the spatial clusterization flow of the vertices with different curvature features. More details on the weights defined in **w3** are provided in [78], where the authors apply a weighted Laplace-Beltrami operator to surface fairing.

2.4.2 The Numerical Algorithm

The simplification algorithm is simple. The multilevel framework allows a progressive simplification of the mesh. The number of levels can be chosen to satisfy special application-dependent requirements. The stopping criterion at each level is specified as a percent reduction of the original mesh (or equivalent) to reach a given maximum simplification value for the entire multilevel process. This percent reduction decreases for increasing levels, motivated by the fact that the mesh size h increases for increasing levels. Therefore, the first levels can better support a more aggressive removal of edges without significant loss of information.

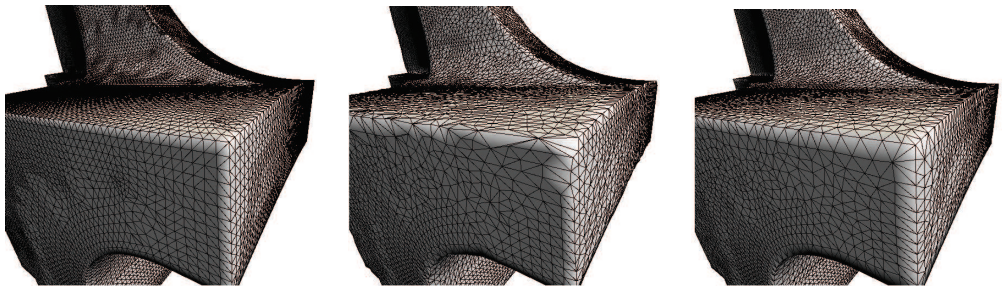
At each level, the algorithm operates by combining iteratively a spatial clustering phase with a subsequent edge decimation phase driven by detected quadratic error metrics and surface features. The three steps of the algorithm are: characterize the local vertex geometry and surface features, clusterize the vertices through a p-Laplacian based flow, and simplify vertices through incremental mesh decimation.

The following algorithm summarizes the computations required by our method. It's able to produce a sequence of simplified meshes $X^{(k)}$ from three input parameters: the original fine resolution mesh with vertex set X_0, the desired Percentage of the total Edges to Remove (PER) and the number of levels $nlev$, that is the number of intermediate meshes inbetween.

Algorithm: Multilevel Mesh Simplification
INPUT: X_0, nlev, PER
OUTPUT: the set of simplified mesh $X^{(k)}$, k=1,2,...
Set $X^{(0)} = X_0$,
for k=0:nlev - 1
  H_k=MeanCurvature($X^{(k)}$)
  $\widetilde{X}^{(k)}$=Clusterize($X^{(k)}$,H_k)
  $X^{(k+1)}$= Decimate($\widetilde{X}^{(k)}$,PER,k,H_k)
end

The decimate procedure at the simplification level k, determines the number of edges to be removed by the linearly decreasing function

$$PER_k = \frac{3(nlev-1)-2k}{2nlev(nlev-1)} * PER \tag{2.99}$$



(a) Original (b) Decimated (c) Clusterized + Decimated

Figure 2.22: `fandisk` mesh: (a) original mesh; (b) decimation only; (c) spatial clusterization and decimation.

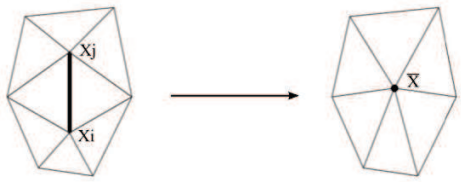


Figure 2.23: edge collapse operation.

that ensures $\sum_{k=0}^{nlev-1} PER_k = PER$.

The aim of the spatial clusterization phase is to help the selection of the edges candidates to be removed and to maintain an appearance of the resulting mesh more natural and less faceted than as they would appear if it was applied the only part of decimation. It exploits the fact that the p-Laplacian operator has different behaviors depending on the exponent p provided. In fact, if $p > 1$ then it behaves as a normal operator for smoothing (Δ_p, with $p = 2$, is in fact the classic Laplace-Beltrami flow already successfully used in surface fairing applications, [78]), but when $p < 1$ then the p-Laplacian flow tends to clusterize the vertices towards the areas of high curvature, and this is precisely the case we need and is used in this algorithm. The benefits of the spatial clusterization flow are illustrated in Fig.2.22 where the results of the decimation phase applied to the mesh in Fig.2.22(a) without spatial clusterization are shown in Fig.2.22(b), while the structure preserving effects of applying the p-Laplacian flow before decimation, are clearly observed in Fig.2.22(c).

For the Decimation step we followed the incremental quadric-based mesh decimation originally introduced by Garland in [41]. The base operation is the edge collapse. An edge collapse is an operation that reduces an edge into a single vertex, i.e. two vertices are merged into one, see Fig.2.23. When this is done all edges and faces connected to the removed vertices have to be reconnected to the new vertex. The problem is to know where to position the new vertex while minimizing the change in overall appearance. When working with quadric-based mesh decimation the error for a vertex X is defined as $X^T Q X$, where the 4×4 matrix Q measures the distance between the vertex and its connecting faces. The incremental decimation procedure thus follows the two steps:

- Put edges into a sorted queue,
- Orderly perform a series of edge collapses.

We propose to use the following sorting criteria driven by a penalty p_e associated to an edge e with vertices X_i and X_j

$$p_e = (1 + \frac{p_{Q_e}}{max_{e \in X} p_{Q_e}})^{\alpha} + (1 + \frac{|e|}{max_{e \in X}|e|})^{\beta}, \tag{2.100}$$

which depends on the quadric error $p_{Q_e} = \overline{X}^T (Q_{X_i} + Q_{X_j}) \overline{X}$ computed on the optimal vertex position $\overline{X}$ and on the edge length $|e|$. The exponents α and β can be used to tune the effect of the quadric error and the effect of the edge length on the ordering of the edge collapse operations.

Every time a decimation has been executed, the surface geometry in the vicinity changes. Therefore, the penalty measure have to be re-evaluated on the affected edges.

2.4.3 Experimental results

We illustrate the performance of the multilevel mesh simplification Algorithm applied to the simplification of several high resolution meshes with different structural characteristics, arbitrary topology, and regular/irregular vertex distributions.

Good algorithms for simplifying triangle meshes are available in common modeling packages like Maya, and freely-available software pack-

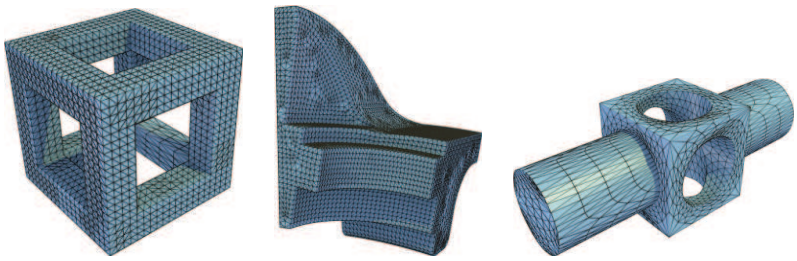


Figure 2.24: Example 1. The original meshes. From left to right: `cube-hole` (Number of edges 7680),`fandisk` (Number of edges 19419) and `mechpart` (Number of edges 5712).

ages such as Blender or MeshLab [23]. We compared the simplification results with the decimation method integrated as a tool in MeshLab which is mainly based on the Garland algorithm [41].

Typical criteria to judge the results of a simplification methods are based on geometric distance (e.g. Hausdorff-distance) or visual appearance (e.g. tesselation quality, feature preservation, ...). In order to evaluate and compare the efficacy of our simplification algorithm we both provide images for visual inspection of the simplified meshes and numerical results. In order to quantitatively assess the quality of the mesh generated by the simplification algorithm we propose the following measures based on the Hausdorff distance error estimate, which is defined to be the maximum minimum distance.

In particular, let $d_{\mathrm{H}}(x_0 \in X_0, X)$ be the distance between a point $x_0 \in X_0$ and the mesh X defined by

$$d_{\mathrm{H}}(x_0, X) = \inf_{y \in X} d(x_0, y). \tag{2.101}$$

We considered unidirectional Hausdorff distance instead of the classical bidirectional definition since we measure the distance between an original fine resolution mesh with a coarse simplification mesh. Therefore we avoid the inaccurate sampling of the coarse mesh to compute the distance to the fine mesh. The distance is carried out by MeshLab software package. We define the following related measures of approximation error:

Maximum Hausdorff distance

$$d_{\mathrm{H}}^{max}(X_0, X) = max_{x \in X_0} d_{\mathrm{H}}(x_0, X) \tag{2.102}$$

that is, a measure of the maximum error that the simplification algorithm performs in simplifying the original mesh X_0 into X.

Mean Hausdorff distance

$$d_{\mathrm{H}}^{mean}(X_0, X) = \frac{1}{|X_0|} \sum_{x \in X_0} d_{\mathrm{H}}(x_0, X) \tag{2.103}$$

that is, a measure of the average simplification error.

RMS Hausdorff distance

$$d_{\mathrm{H}}^{RMS}(X_0, X) = \sqrt{\frac{1}{|X_0|} \sum_{x \in X_0} d_{\mathrm{H}}^2(x_0, X)} \tag{2.104}$$

that is, a measure of the error introduced by the simplification algorithm.

The goal of our simplification algorithm is to minimize the error introduced by the simplification, i.e., to minimize the proposed distances. With the following three examples we demonstrate how our algorithm improves the tessellation quality while reducing mesh complexity and maintaining the arbitrary topology of the initial fine resolution mesh. Moreover, the numerical results in terms of Housedorff distances show a better performance with respect to the MeshLab simplification tool.

Example 1. In this example, we demonstrate the efficacy of how when dealing with meshes characterized by many sharp features. The high resolution meshes considered are illustrated in Fig 2.24. In Fig. 2.25, 2.26 and 2.27 the simplified meshes are shown for increasing percentages of final simplifications. From a qualitative comparison we can visually appreciate the improvement with respect to the MeshLab simplifications where the tesselation quality is poor and the global shape is badly preserved. In particular, for the meshes `cube-hole` and `fandisk` our algorithm noticeably improves the simplification results providing a reduced mesh that perfectly preserves sharp features, but also provides well shaped triangles. The quality of the triangulation represents a fundamental requirement for successive mesh processing tasks.

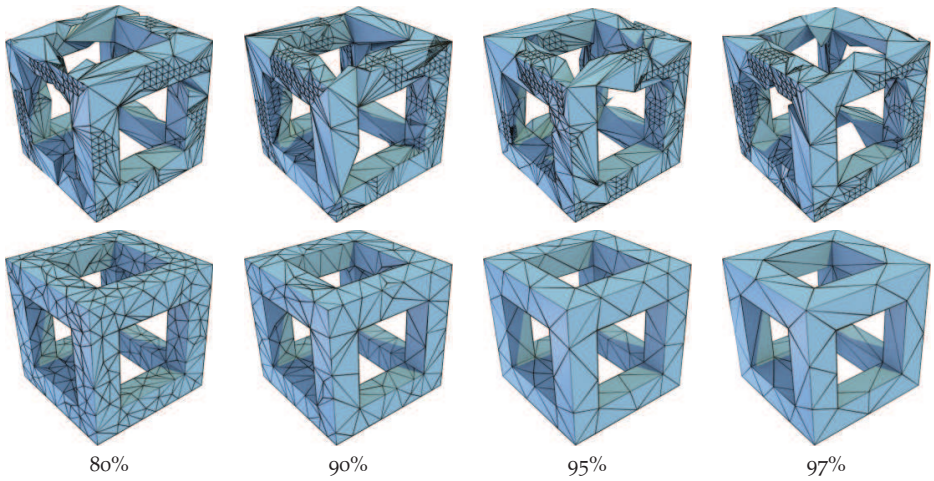


Figure 2.25: Example 1. Simplification of the `cube_hole_tri` mesh. Top row: MeshLab simplification. Bottom row: our algorithm.

Table 2.4 reports results obtained by the simplifications of meshes in Fig.2.24 which are visualized by using bar-plots in Fig. 2.28. Comparisons between the results of the two simplification methods in Table 2.4 demonstrate the improvements obtained by our method and confirm the visual inspection of Fig. 2.25, 2.26 and 2.27. For all the simplification experiments the parameters α and β in (2.100) have been set to 1 and we always performed three level of simplifications ($nlev = 3$), although, for comparison purposes, we only showed the final level.

Example 2. In this example we evaluate the size-quality tradeoff results on input meshes of large sizes by performing aggressive simplifications to 99% and 99.9% of the original mesh.

Fig. 2.29 (`horse`), Fig. 2.30 (`dinosaur`), Fig. 2.31 (`hand`), Fig. 2.32(`foot`) and Fig. 2.33(`igea`), illustrate original and decimated versions generated by the multilevel simplification algorithm applied to different models of different complexities. The multilevel mesh simplification algorithm is capable of reducing an initial mesh to an extremely low number of faces well preserving the global mesh shape. Moreover, even after decimating 99% and 99.9% of the edges, the simplified models still show a very good tesselation quality independently on the regularity of the input mesh. Consider, for example, the `dinosaur` model depicts in Fig. 2.30. In spite of a 99.0% reduction in size, the resulting simplified mesh keeps the overall

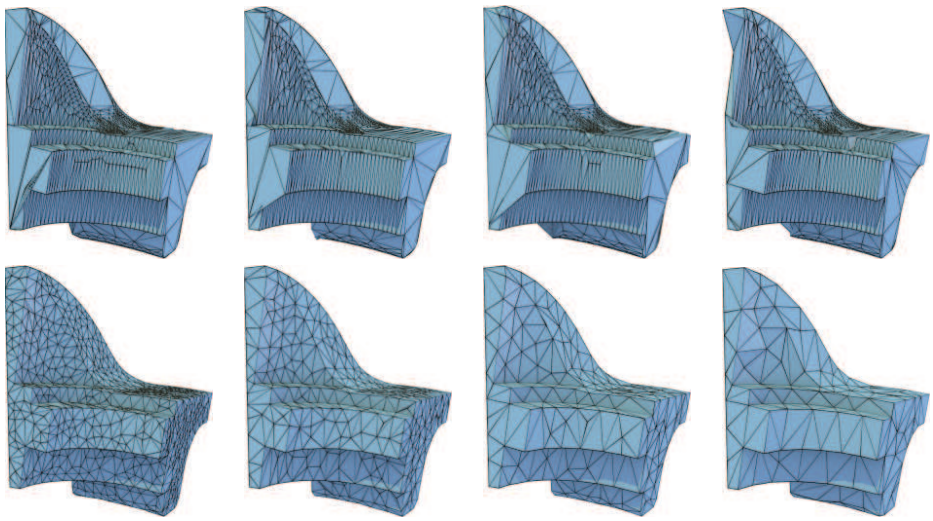


Figure 2.26: Example 1. Simplification of the `fandisk` mesh. Top row: MeshLab simplification. Bottom row: our algorithm.

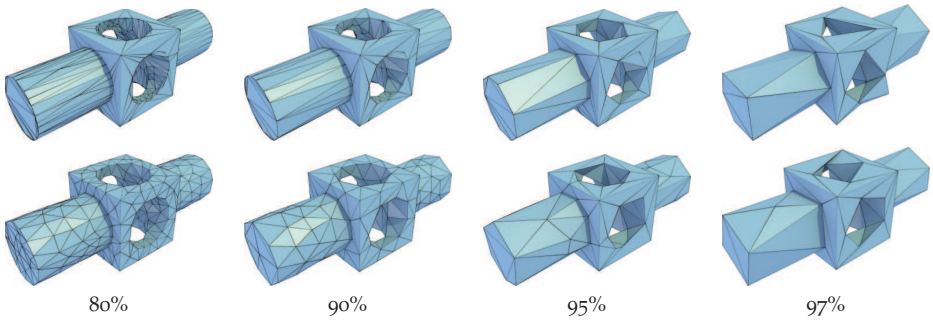


Figure 2.27: Example 1. Simplification of the `mechpart` mesh. Top row: MeshLab simplification. Bottom row: our algorithm.

appearance of the original model and preserve many visually important parts like the skeleton details on the back and the foot fingers.

A numerical assessment is given in Table 2.5, where a comparison with the MeshLab simplification tool is also shown. It is apparent that our algorithm performs better than MeshLab in terms of Hausdorff distance geometric errors.

Example 3. In this example we illustrate the multilevel capabilities of the proposed algorithm. We applied the Multilevel Simplification Al-

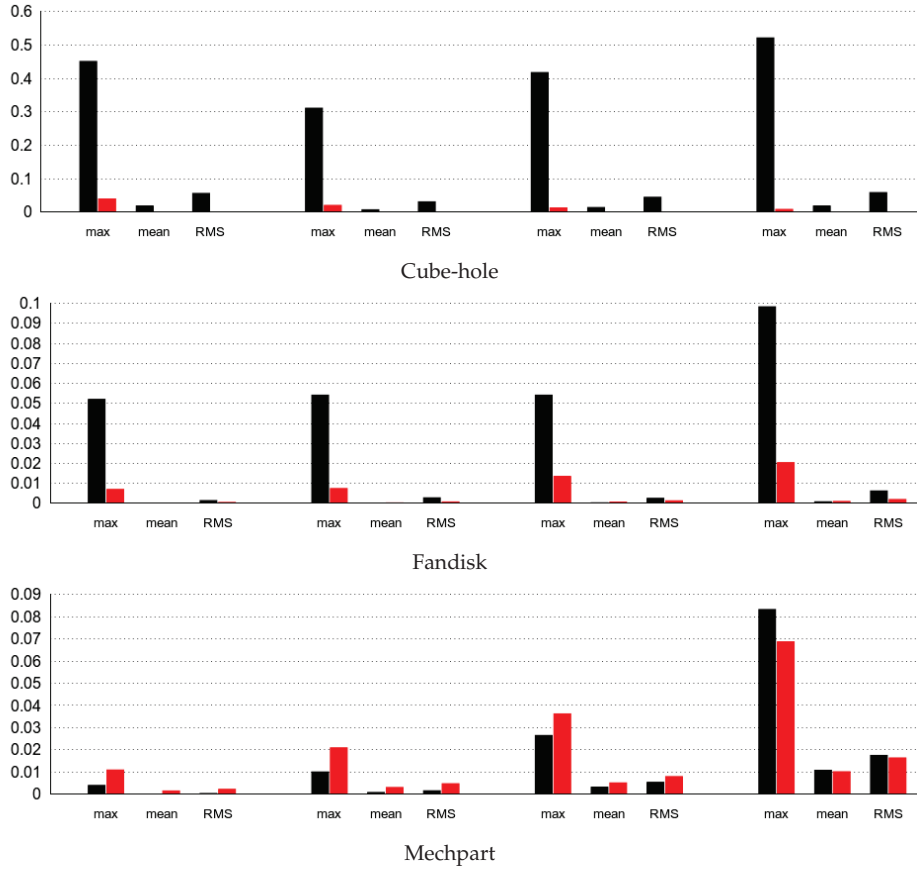


Figure 2.28: Comparison of max, mean and RMS Hausdorff distances for meshes in Fig. 2.24 and simplification percentages. Black bars are MeshLab results while red ones are obtained by the Multilevel Mesh Simplification Algorithm. Lower is better.

gorithm with $nlev = 7$ to reduce the original mesh **hand** shown in Fig. 2.31(left). The method produces a sequence of increasingly simplified meshes, where at each level k the percentage of edges to remove is defined by PER_k in (2.99), in order to obtain a final simplification of 99.0%. Instead of collapsing a constant number of edges at each level the decreasing function (2.99) allows for a decimation which is proportional to the size of the mesh. Fig. 2.34 shows the simplified models at each level k which maintain a very good tesselation quality, throughout all levels of simplification.

Mesh	Algo.	Simp.(%)	d_{max}	d_{mean}	d_{RMS}
cube-hole	MeshLab	80	0.451881	0.020130	0.057686
cube-hole	Our	80	0.041205	0.000515	0.001484
cube-hole	MeshLab	90	0.312231	0.008965	0.032611
cube-hole	Our	90	0.021760	0.000822	0.001798
cube-hole	MeshLab	95	0.418654	0.015544	0.046249
cube-hole	Our	95	0.014150	0.000815	0.001572
cube-hole	MeshLab	97	0.522198	0.019978	0.060159
cube-hole	Our	97	0.009750	0.000937	0.001454
fandisk	MeshLab	80	0.052219	0.000163	0.001662
fandisk	Our	80	0.007219	0.000331	0.000728
fandisk	MeshLab	90	0.054350	0.000450	0.003027
fandisk	Our	90	0.007670	0.000505	0.000976
fandisk	MeshLab	95	0.054343	0.000536	0.002795
fandisk	Our	95	0.013716	0.000855	0.001540
fandisk	MeshLab	97	0.098489	0.001068	0.006474
fandisk	Our	97	0.020589	0.001210	0.002178
mechpart	MeshLab	80	0.004266	0.000380	0.000633
mechpart	Our	80	0.011192	0.001768	0.002482
mechpart	MeshLab	90	0.010306	0.001085	0.001842
mechpart	Our	90	0.021135	0.003309	0.005000
mechpart	MeshLab	95	0.026714	0.003438	0.005651
mechpart	Our	95	0.036440	0.005390	0.008207
mechpart	MeshLab	97	0.083372	0.011076	0.017769
mechpart	Our	97	0.068840	0.010379	0.016617

Table 2.4: Example 1. Simplifications of the fine resolution meshes in Fig.2.24 by applying our algorithm and MeshLab simplification tool.

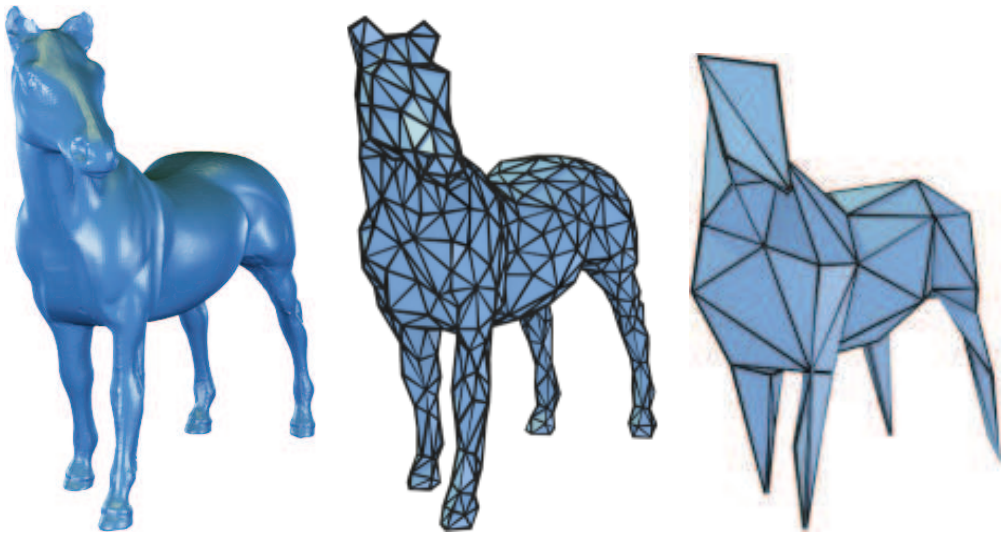


Figure 2.29: Example 2. Simplification of the `horse` mesh (Number of edges 145449, 1449, 144).

Figure 2.30: Example 2. Simplification of the `dinosaur` mesh (Number of edges 168576, 1680, 168).

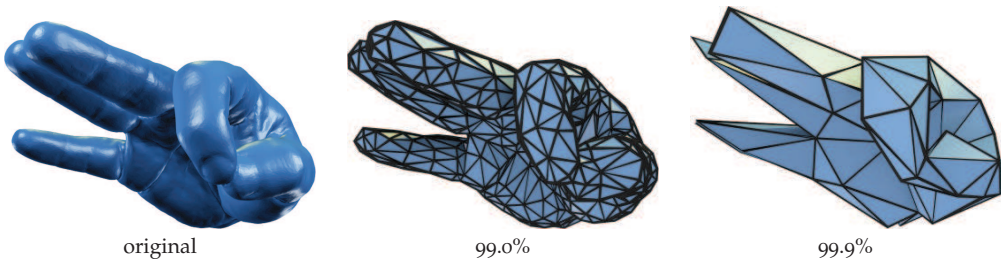


Figure 2.31: Example 2. Simplification of the `hand` mesh (Number of edges 152169, 1518, 150).

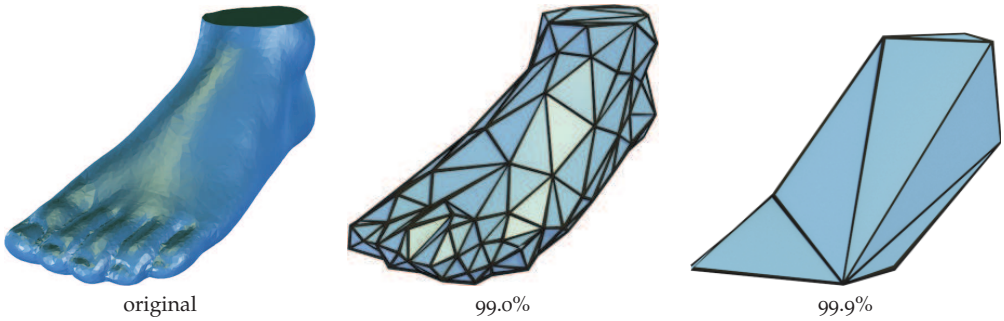
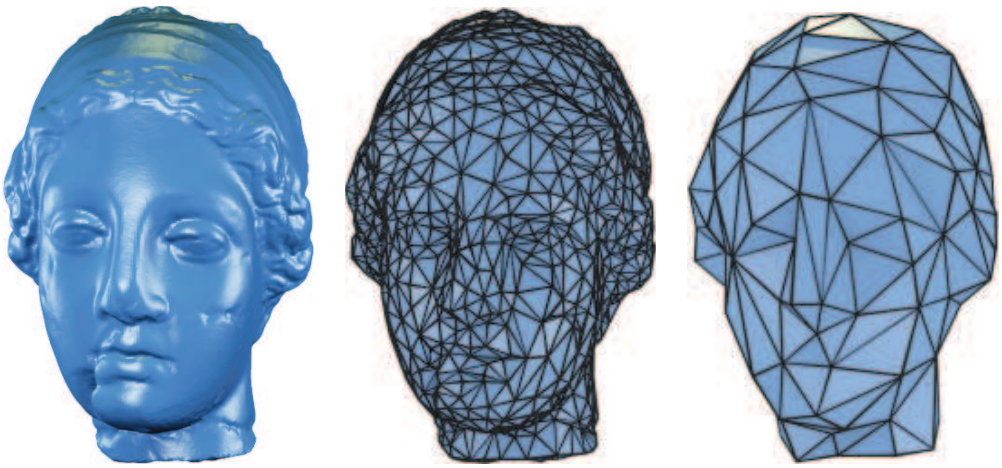


Figure 2.32: Example 2. Simplification of the `foot` mesh (Number of edges 30714, 303, 24).



original 99.0% 99.9%

Figure 2.33: Example 2. Simplification of the `igea` mesh (Number of edges 403029, 4023, 396).

Mesh	Algo.	Simp.(%)	d_{max}	d_{mean}	d_{RMS}
horse	MeshLab	99.0	0.028324	0.002269	0.003014
horse	Our	99.0	0.023522	0.002283	0.002976
horse	MeshLab	99.9	0.161733	0.018871	0.027483
horse	Our	99.9	0.161740	0.020073	0.027316
dinosaur	MeshLab	99.0	0.022501	0.003289	0.004312
dinosaur	Our	99.0	0.018831	0.003295	0.004232
dinosaur	MeshLab	99.9	0.125394	0.018922	0.024967
dinosaur	Our	99.9	0.134250	0.018896	0.025031
foot	MeshLab	99.0	0.023356	0.003518	0.004635
foot	Our	99.0	0.017761	0.003127	0.004170
foot	MeshLab	99.9	0.110770	0.024386	0.031521
foot	Our	99.9	0.090827	0.025222	0.030887
hand	MeshLab	99.0	0.015215	0.002073	0.002715
hand	Our	99.0	0.017967	0.001979	0.002536
hand	MeshLab	99.9	0.121775	0.017971	0.024672
hand	Our	99.9	0.085372	0.017085	0.022208
igea	MeshLab	99.0	0.012795	0.001324	0.001753
igea	Our	99.0	0.011045	0.001340	0.001759
igea	MeshLab	99.9	0.046810	0.008417	0.010807
igea	Our	99.9	0.042718	0.008091	0.010267

Table 2.5: Example 2. Simplifications of the fine resolution meshes in Fig.2.29, 2.30, 2.31, 2.32 and 2.33 by applying our algorithm and MeshLab simplification tool.

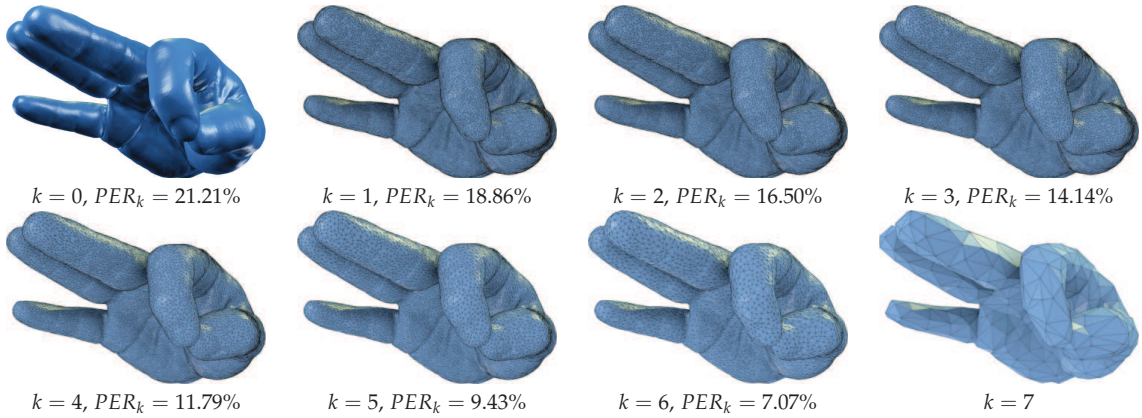


Figure 2.34: Example 3. Multilevel simplification of the hand mesh.

Part II

Virtual Modeling

Chapter 3

Reverse Engineering and Virtual Modeling

3.1 Introduction

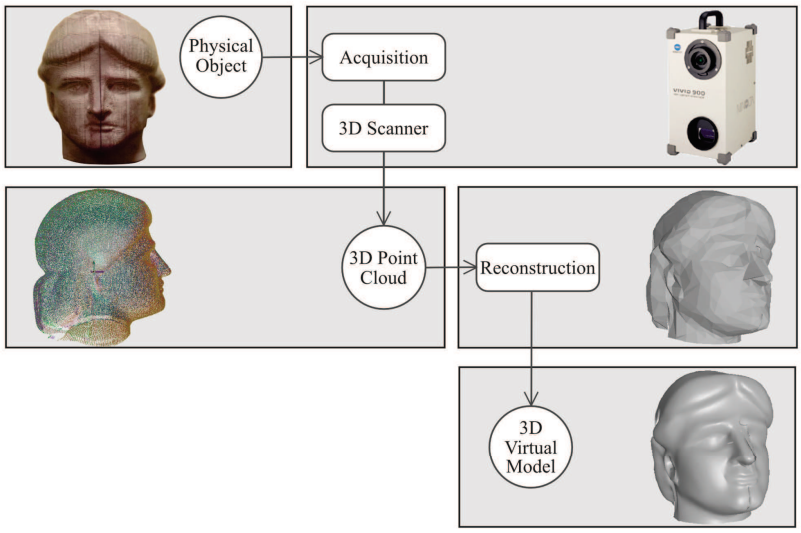
In the field of Computer-Aided Design (CAD), reverse engineering has become an effective method to create a 3D virtual model of a physical object for later use in software for computer aided design and manufacturing. Reverse engineering has many applications in different fields, such as medical imaging, entertainment, cultural heritage, web commerce, collaborative design and obviously engineering; all these applications can take advantage in different ways from the reconstructed 3D virtual model. Traditional reverse engineering involves two main steps (see Figure 3.1a), the measurement of the physical object and its reconstruction as a 3D virtual object. The physical object can be measured using 3D scanning technologies such as coordinate measuring machines or computed tomography scanners, which provide outputs in the form of an unstructured point cloud, i.e. a large set of vertices in a three dimensional coordinate system, which lacks topological information and therefore is generally not directly usable in most 3D applications. The point cloud is then usually converted to a mesh model, NURBS (Non Uniform Rational B-Spline) surface model, or CAD model through a process commonly referred to as 3D reconstruction so that it can be used for various purposes. This second step of reconstruction of the virtual 3D object from the dense point

cloud is an inverse problem and generally does not admit a unique solution. Most proposed approaches to reconstruction from unstructured data points build polygon meshes that interpolate or approximate the input points. The fundamental difficulties of reconstruction arise from the lack of topological information in the data but also from the noise and inaccuracies of the measuring process, the presence of obstructions and holes and consequently, additional assumptions and requirements on the input data are generally needed to make the problem tractable. As a result, most reconstructed models need to be post-processed for simplification and optimization introducing another step in the reverse engineering process.

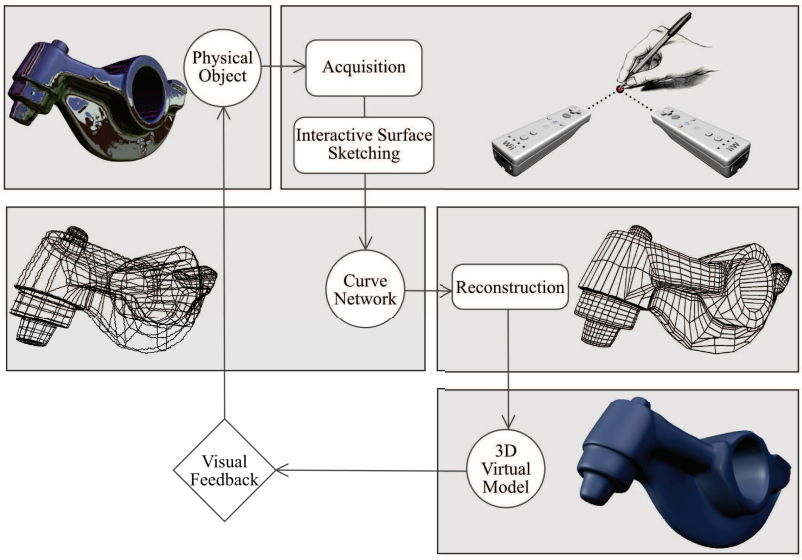
The steps of measuring and reconstruction could be achieved by using different techniques and devices, and all of them have strengths and weaknesses. Regarding the alternative 3D scanning methods, we can outline the following quality measures: accuracy and resolution, environmental sensitivity, repeatability, speed, and cost. A reconstruction process is usually required to be automatic, sensible to the object topology, time and space efficient, and robust (with respect to noisy data). All current reverse engineering solutions suffer from some common limitations. For example, the overwhelming number of points acquired and the lack of topological information in this data, combined with the presence of noise and inaccuracies, usually require complex and time-consuming solutions. Moreover, the strict separation of the two fundamental steps of measuring and reconstruction makes this process a non-iterative and non-interactive process.

In this thesis, we introduce a new method of reverse engineering for fast, simple and interactive acquisition and reconstruction of a virtual 3D model representing an existing physical object that exploits a pen-based active stereo acquisition system supported by a reconstruction and visualization layer based on subdivision surfaces.

The design and development of 3D spatial tracking devices together with digital sketch-based interfaces represents a powerful way to combine the natural and intuitive human expression with the power of computation. However, the potential of these sketch-based systems strongly depends on the effectiveness of the input devices, on the facilities provided by the user interface as well as on the underlying algorithms to create digital 3D



(a)



(b)

Figure 3.1: (a) Traditional reverse engineering pipeline, (b) Fast and Interactive Reverse Engineering pipeline

models from the input.

The goal of this thesis is to enable real-time interactive designing/editing of a 3D shape, by sketching a 3D network of curves that approximate the desired shape. To achieve this goal we developed an acquisition and reconstruction framework for arbitrary network of 3D curves and we designed and developed a low cost, low power consumption wireless pen-like device with the capability of drawing and selecting points and curves, which introduces a natural way to draw and edit the style-lines of a physical object.

We named our approach Fast Interactive Reverse Engineering System (FIRES).

FIRES integrates the 3D curve sketching and the surface construction step into an iterative and incremental process that allows the user to have a real-time visual feedback on the ongoing work.

The 3D curve sketching process (see Acquisition in Fig. 3.1b) is supported by an active stereo vision system made of two infrared cameras and (at least) one infrared light emitter mounted on a pen-like device. The pen 3D position is tracked by the stereo rig and the user can intuitively draw and refine the style lines of the object. The user sketches arbitrary 3D curves, and interactively, at each traced curve, the system adapts the shape so that the sketch becomes a feature line on the model. The process of interactively and incrementally drawing the irregular curve network is called Interactive Surface Sketching (ISS). When a designer defines a shape with 3D curves, it is often the case that these curves indicate the curvature features of the surface. Intuitive and aesthetically pleasing surfaces are obtained by the system using a curve not only as a series of positional constraints but also taking into account the sharpness information given by the user on the characteristic lines.

Although the resulting 3D model does not have the accuracy of classical Reverse Engineering systems our approach allows us to significantly shorten acquisition and integration time of the real model directly into the virtual environment up to an interactive feedback.

3.1.1 Related work

Computer modeling of 3D geometry using alternative three-dimensional user interfaces (3D UIs) and interaction techniques has received considerable attention in recent years. While a number of techniques involving 3D UIs, 3D devices, haptic devices, and VR systems have been proposed, the usability of 3D UIs in many real world applications is still surprisingly low. For example, the digitizer 3D Microscribe, interfaced to the Rhinoceros commercial CAD software, is limited by its wired short connection, low usability and poor shape reconstruction support. Research in 3D UIs has addressed both the design of novel 3D input or display devices, and the development of design and/or evaluation approaches specific to 3D UIs. An in-depth survey on new directions in 3D user interfaces is given in [17].

Much of the early work on 3D user interfaces focused on systems for inferring plausible 3D free-form shapes from visible-contour sketches, which involves the difficulty of interpreting 3D information from 2D input. Some works on generating 3D geometry by inflating 2D silhouettes has been proposed in [50, 55]. The emphasis in such systems is to quickly generate a reasonable 3D shape rather than a precise modeling of the object. In gesture-based techniques designers' strokes are used for editing exiting primitive objects into the desired shape, [112]. A number of template-based methods have also been proposed, where the desired 3D form is obtained by deforming an underlying 3D template, such as for example a six-faced topological template,[70, 54, 30], or a given network of curves, [43, 86]. Recently, a system for designing free form surfaces from a collection of projected 3D curves inserted through a 2D line drawing sketching system has been presented in [80]. This approach lacks from a direct control on the object shape since a functional optimization is used to construct the smoothed surface. A real 3D sketching system is instead presented in [106] using a projection-based virtual environment. The resulting surface is created stitching together pieces of spline patches with C^0 or G^1 continuity. The development of 3D input devices is of great interest not only for sketching and modeling, but also for computer animation and interactive control [85].

Authors does not know any other pen-based reconstruction approach,

but literature papers present several other applications based on pen-like hardware; they are focused on handwritten, gestures recognition and HCI. These approaches involved two main information sources: off-line (video-based) and in-line recognition (inertial sensors). An interesting work about 3 axes digital (3D) accelerometer-based recognition is presented in [22] where an Hidden Markov Model recognizes Arabic numerals. Dynamic Time Warping technique normalizes temporal windows allowing data templates comparison while PCA is used to under-sampling the data. The poor dataset acquired doesn't permit to quantify the classification performance. In [83] the system is improved with a 3D gyroscope and taking advantage from the use of an ensemble recognizer consisting of 3 FDA (Fisher Discriminant Analysis) it reaches the generalization rate of 95.04% with a dataset of 4,945 samples acquired on 16 people. Also [51] present a pen system comprising accelerometer and gyroscope, both 3D, to reproduce the sign trace. In [7] a dual 3D accelerometer configuration is used; furthermore a pressure transducer permits to compensate the accelerometer offset voltage.

On the market different products are also available. Most of the solutions are off-line handwriting recognition where a tiny video sensor is embedded inside the pen. Instead VPen from OTM Technologies includes a laser diode, detectors and optics to converts handwriting to ASCII text supporting Latin and Asian characters but it is also capable of 3D detection.

The reconstruction problem in our domain can be generalized as the fitting of a surface mesh which interpolates a given curve network. These methods can be roughly classified into two categories: methods which use smoothly stitching parametric patches, like Bézier, spline patches, or subdivision surfaces [93],[63], and, more recent approaches, which construct a smooth surface embedding by applying functional optimization.

In this thesis we followed both approaches. The first version of our reverse engineering system FIRES^{V1} is based on bilinearly blended Coons patches, while the second version FIRES^{V2} constructs a surface using a surface diffusion flow.

3.2 System Overview

In this section we present the FIRES architecture which integrates the steps of measuring and reconstruction into an iterative and incremental process that allows the user to have a real time visual feedback on the ongoing work.

Experimental equipment during a typical FIRES work session is shown in Figure 3.3b. The measuring step (see Acquisition block in Figure 3.1b) is achieved through an active stereo vision system made of two infrared cameras and (at least) one infrared light emitter usually mounted on a pen-like device. The pen 3D position is tracked by the stereo rig and the user can intuitively draw and refine the style lines of the object, i.e. the lines and curves that mainly characterize the object shape. We note that traditional design is mostly based on drawing characteristic curves for designing a surface. This set of 3D curves is called the Curve Network and the process of interactively and incrementally drawing the curve network is called Interactive Surface Sketching (ISS).

Active involvement of the user in the acquisition process has different advantages. For example, it allows for a fast interaction by adding, modifying or discarding measures right during the acquisition process, and the detection of features of the objects like creases, corners and symmetries. Moreover, using a specific set of modeling tools, described in Section 5, the user naturally provides topological information on the object to be reconstructed. As a consequence of this, the reconstruction step is simplified with respect to the classical RE reconstruction.

The ISS produces a curve network which is internally represented as a polyline mesh, described in Section 3.3, that is, a mesh with faces, vertices and edges augmented with polylines associated to each edge.

The surface reconstruction step (see Reconstruction in Figure 3.1b) produces meshes and subdivision surface representations that, due to their intrinsic recursive nature, perfectly fit in FIRES real time process providing a fast, multiresolution method for the representation and visualization of surfaces. To this aim, FIRES implements the three following steps (see Figure 3.2):

1. triquadrification: from polyline mesh to Base Mesh

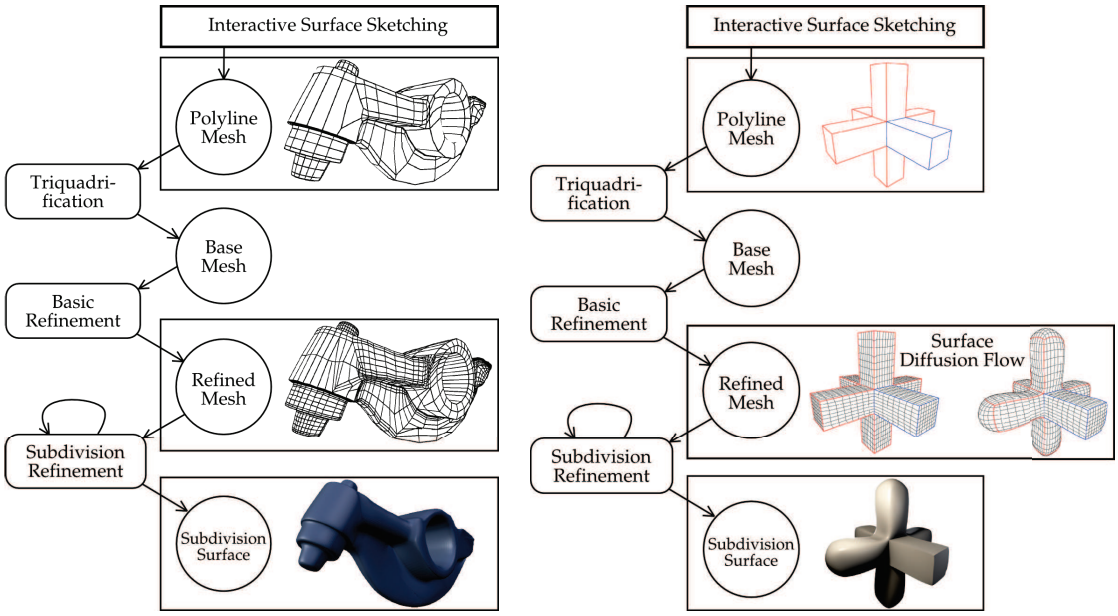
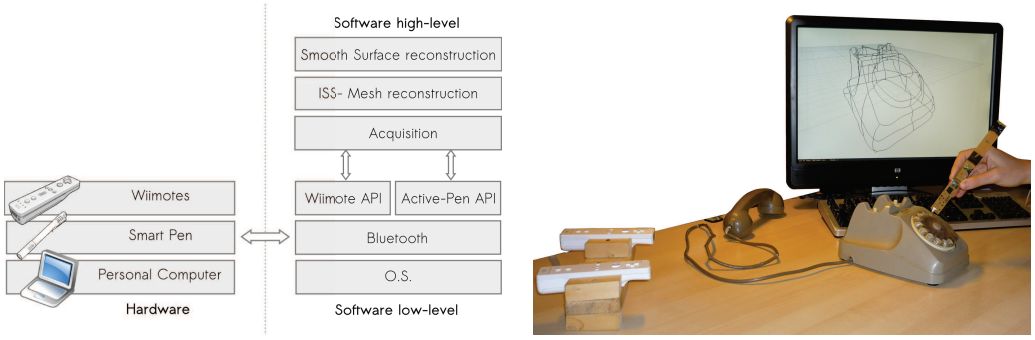


Figure 3.2: The multi-step reconstruction process in FIRES^{V1}(a) and FIRES^{V2}(b).

2. basic refinement: from Base Mesh to Refined Mesh
3. subdivision refinement: from Refined Mesh to Smooth Surface.

Since the curve network can be irregular, that is, the curves can intersect each other without any restriction on the number of curves intersecting at a given point, its associated polyline mesh can contain n-sided, non-planar, and non-convex faces. Then, the objective of the triquadrification step is to find a good splitting for each n-sided face in three and four sided polygons in order to construct a Base Mesh. The Base Mesh is however very coarse, especially at the beginning of the reconstruction process. We exploit bilinearly blended Coons patches in FIRES^{V1} and a surface diffusion flow in FIRES^{V2} to refine the Base Mesh into a resulting Refined Mesh which interpolates points on the polylines, thus to supply sufficient data to the last step of reconstruction through subdivision. The Refined Mesh is finally refined by a subdivision scheme that produces a smooth surface interpolating or approximating the given curve network.

In Figure 3.3a the hardware and software layers of FIRES are illustrated.



(a) (b)

Figure 3.3: (a) Hardware and software layers in FIRES, (b) Experimental equipment and visual device during a work session aimed to reconstruct an old telephone

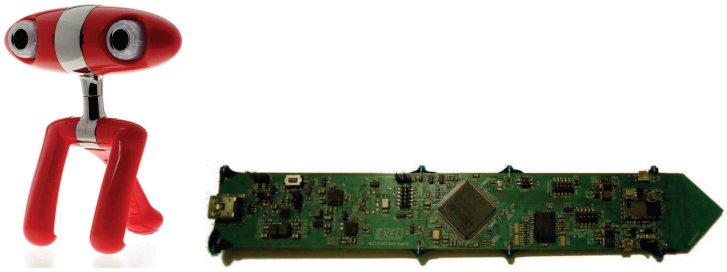


Figure 3.4: The minoru stereoscopic camera (a) and the new smart-pen (b) used in FIRES^{V2}.

The minimal required hardware devices in FIRES could be any active stereo vision system capable of real-time Infra Red (IR) 3D tracking of a wireless led emitter. The crucial aspect pursued in FIRES is a low-cost technology strategy to achieve an optimal compromise between accuracy and cost. In fact we have equipped FIRES^{V1} with two Nintendo Wii remote controllers (wiimotes) and an ad-hoc developed pen-like device which communicates with the software layers via a bluetooth interface. The data available on the wiimote cameras are obtained through the wiiuse[59] open-source C library built on top of the OS bluetooth stack. In FIRES^{V2}, we upgraded the stereo rig to use the Minoru cameras (shown in Fig. 3.4a) and an improved pen device, called the smart-pen (shown in Fig. 3.4b).

The hardware devices in FIRES, besides their use for acquisition and reconstruction purposes, are used as a system for navigation in the workspace. That is, the user can intuitively control the virtual camera of FIRES directly moving the smart-pen around the scene pointing the pen tip towards the region of interest. Automatically, the system takes the position and direction of the smart-pen as respectively the origin and the virtual camera direction and provides a visual feedback of the reconstruction. This additional feature of FIRES enhances the system usability.

3.3 Polyline Mesh Data Structure

The 3D acquisition and reconstruction layers are internally supported by an ad-hoc designed data structure called Polyline Mesh which expands a general mesh data structure for storing the acquired curve network. The design of this data structure has been tailored to provide an efficient and easy-to-use way to store the acquired curve network and to support the development of the reconstruction and visualization algorithms that need to be executed with strict time requirements to provide a real-time visual feedback on the ongoing process. During the process of reconstruction from curve network, summarized in Figure 3.2, different algorithms modify the geometric and topological information contained in the polyline mesh. Therefore the data structure should offer facilities for storing, accessing and modifying the information contained in it.

The polyline mesh is the geometrical representation of the model underlying the curve network created by the user during the process of ISS. While the curve network is only a visual representation of the object in terms of curves in the 3D space, the polyline mesh contains also topological information. The Base Mesh and the Refined Mesh are successive manipulations that alter only the geometry of the virtual model.

A polyline mesh $\mathcal{M}$ is a collection of vertices, polylines, edges and faces defined as $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{P})$ where:

$\mathcal{V}$ is a set of points $v \in \mathbb{R}^3$ called vertices

$\mathcal{P}$ is a set of polylines P. A polyline is a series of line segments connecting consecutive vertices p

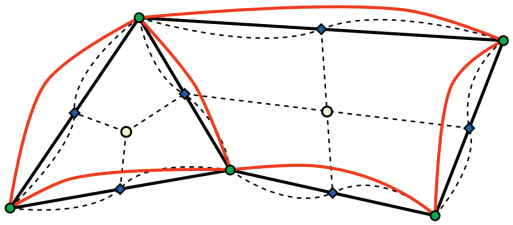


Figure 3.5: Access relations among polyline mesh components. Polylines are represented in curved solid, edges in straight solid and the vertices of the polyline mesh as solid dots. The dashed lines represent bidirectional references available in the data structure. The elements are represented by rhombus marks for edges and empty dots for faces. The references edge-polyline, not shown here, are unidirectional

$\mathcal{E}$ is a set of edges e. An edge is identified by a pair of vertices v and a polyline P

$\mathcal{F}$ is a set of faces f. A face is a closed sequence of edges
Between the elements of $\mathcal{E}, \mathcal{V}, \mathcal{F}, \mathcal{P}$ the following relations must be respected:

1. Every $e \in \mathcal{E}$ edge connects exactly two vertices and is shared at most between two faces.
2. Every $e \in \mathcal{E}$ has a single associated polyline $P \in \mathcal{P}$. The starting and ending vertices of the edge coincide with the first and the last vertex of P.
3. Every $v \in \mathcal{V}$ is shared by at least two edges.
4. Every $f \in \mathcal{F}$ is closed, i.e. the starting vertex of the first edge coincide with the ending vertex of the last edge.

An important issue in the design of a polygon mesh data structure is the choice of the appropriate access relations between faces, edges and vertices. The implementation of the polyline mesh provides the following access relations illustrated in Figure 3.5: every face has a reference to an ordered list of edges, every edge has exactly two reference to vertices, at most two reference to faces and exactly one reference to a polyline, every vertex has a reference to an unordered list of edges. If we describe these relations using the notation used by Rossignac[89] we have:

$$\left\{\mathcal{F},\mathcal{E},\mathcal{V},\mathcal{P} \;:\; \mathcal{F} \Rightarrow \mathcal{E} \overset{2}{\to} \mathcal{V}\ ,\ \mathcal{V} \to \mathcal{E} \overset{\leqslant 2}{\to} \mathcal{F}\ ,\ \mathcal{E} \overset{1}{\to} \mathcal{P}\right\}$$

Chapter 4

Curve Acquisition

The acquisition framework, illustrated in Fig. 4.1 and Fig. 4.2, consists in a set of techniques that, combining a custom designed smart-pen and a stereo optical tracking system, allow the user to draw 3D curves in space, to navigate the scene and to easily interact with the system.

The smart-pen is provided with four collinear IR led emitters, and is further equipped with a 3-axis accelerometer, a 3-axis gyroscope, a magnetometer, four mode buttons, a vibrating motor and a speaker, and it has been designed and prototyped for satisfying the main requirements in terms of 3D tracking and Human Computer Interaction. Its wireless capability, small dimensions and low-weight allow to move the pen naturally in the workspace. The inertial sensors gather information about the acceleration and orientation of the device and, together with the 4 IR led emitters, provide the hardware support for the tracking system. The 4-mode buttons equipped on the device allow to interact with the system without the need of using the keyboard and mouse. These characteristics allow to maintain the attention and effort of the user onto the object to be

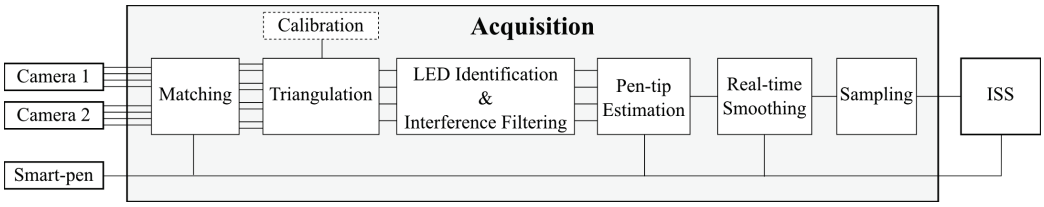


Figure 4.1: 3D curve acquisition pipeline.

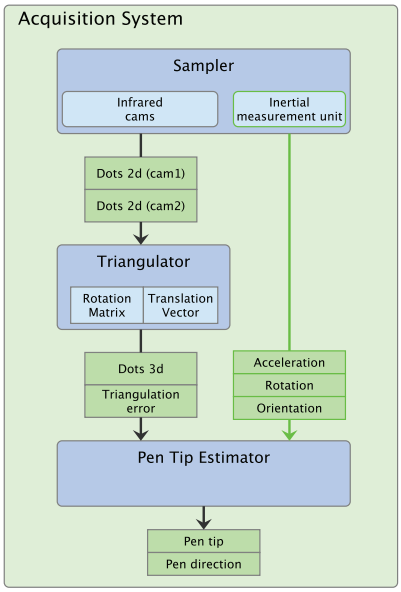


Figure 4.2: Software architecture of the acquisition system.

reconstructed, rather than the screen or the keyboard.

In FIRES^{V1}, the stereo optical tracking system exploits commercial infrared cameras available in the Nintendo Wii Remote, which are equipped with a 128×96 IR monochrome camera that includes a built-in processor capable of tracking up to 4 moving IR sources, and, with further processing, provides an image plane (u, v) with a virtual resolution of 1024×768 pixels. The wiimotes were chosen because of their wide availability, their accessible price, and their technical specifications that allow us to perform a real-time 100Hz tracking of four IR led emitters. Both the wiimotes and the smart-pen communicate with the software layers via a bluetooth interface.

In FIRES^{V2}, we switched to an improved stereo setup based on the Minoru stereo cameras, equipped with a 640×480 sensor. Processing of the 4 moving IR sources is implemented in software via OpenCV and the cvBlob library. Limitations in the Linux drivers allow us to stream only at 15Hz.

The raw data gathered both from the smart-pen and from the cameras is processed according to the acquisition pipeline described in Fig. 4.1.

The projected led image points of both cameras are first matched and then triangulated in order to reconstruct at most four 3D point locations representing the position in space of the leds mounted on the smart-pen. After identification the 3D points are used to estimate the position of the pen-tip. The sequence of 3D pen-tip positions traces a curve that is shown to the user after a real-time smoothing and sampling step.

Using the sensor inertial data, the pen orientation is estimated and used by the matching process to correctly order the image points on the camera plane. The triangulation step then outputs the coordinates of the 3D points computed from the ordered pairs of corresponding image points. Calibration must be performed before each acquisition session.

Due to the limited resolution of the IR cameras, the pen-tip position is in general perturbed by noise. Inertial data from the smart-pen sensors is combined with the vision system data in a Kalman filter to detect the correct pen orientation, used to compute the pen-tip position. The smart-pen sensor's accelerometric data could also be useful in the estimation of the pen-tip location when more than two leds are occluded and the pen-tip is hidden to the stereo vision system. In these cases, the pen-tip position can be obtained through integration of the pen acceleration in time starting from the last known position computed by the vision system.

4.1 3D Tracking

In the following section we describe the stereo vision triangulation technique used to compute the 3D coordinates corresponding to each pair of matched image points on the two cameras.

4.1.1 Stereo vision triangulation

The objective of the stereo vision triangulation is to find the coordinates of a 3D point $\boldsymbol{p}$ starting from two corresponding image points $\boldsymbol{m}_l, \boldsymbol{m}_r$ and the projection matrices M_l, M_r of the left and right cameras such that $\boldsymbol{p} = M_l^{-1}\boldsymbol{m}_l$, $\boldsymbol{p} = M_r^{-1}\boldsymbol{m}_r$. Different triangulation methods exists in literature for solving this issue, see [44]. In FIRES, different methods

have been implemented and are available to the user. The calibration method described in [113] has been used to calibrate the stereo rig, i.e. to determine the matrices M_l, M_r.

Given a pair of homogeneous normalized image coordinates $\boldsymbol{m}_l = [u_l \; v_l \; 1]^T$, $\boldsymbol{m}_r = [u_r \; v_r \; 1]^T$ corresponding to the unknown 3D point $\boldsymbol{p} = [x \; y \; z \; 1]^T$ and positioning the reference system in the left camera center we have:

$$\boldsymbol{p} = \boldsymbol{M}_l = \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{M}}_r = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = R(\tilde{\boldsymbol{M}}_l - \boldsymbol{C}_r) \tag{4.1}$$

Using the pinhole camera model and considering normalized image coordinates we can write

$$\tilde{\boldsymbol{M}}_l = z_l \boldsymbol{m}_l \; , \quad \tilde{\boldsymbol{M}}_r = z_r \boldsymbol{m}_r \tag{4.2}$$

and then

$$z_r \boldsymbol{m}_r = R(z_l \boldsymbol{m}_l - \boldsymbol{C}_r) \quad \Rightarrow \quad \begin{bmatrix} z_r u_r \\ z_r v_r \\ z_r \end{bmatrix} = \begin{bmatrix} z_l(\boldsymbol{r}^{1T} \cdot \boldsymbol{m}_l) - \boldsymbol{r}^{1T}\boldsymbol{C}_r \\ z_l(\boldsymbol{r}^{2T} \cdot \boldsymbol{m}_l) - \boldsymbol{r}^{2T}\boldsymbol{C}_r \\ z_l(\boldsymbol{r}^{3T} \cdot \boldsymbol{m}_l) - \boldsymbol{r}^{3T}\boldsymbol{C}_r \end{bmatrix} \tag{4.3}$$

Equation (4.3) could be rewritten in a linear system $A\boldsymbol{z} = \boldsymbol{b}$ with unknowns z_l, z_r:

$$\begin{bmatrix} u_r - (\boldsymbol{r}^{1T} \cdot \boldsymbol{m}_l) \\ v_r - (\boldsymbol{r}^{2T} \cdot \boldsymbol{m}_l) \\ 1 - (\boldsymbol{r}^{3T} \cdot \boldsymbol{m}_l) \end{bmatrix} \begin{bmatrix} z_r \\ z_l \end{bmatrix} = \begin{bmatrix} -\boldsymbol{r}^{1T}\boldsymbol{C}_r \\ -\boldsymbol{r}^{2T}\boldsymbol{C}_r \\ -\boldsymbol{r}^{3T}\boldsymbol{C}_r \end{bmatrix} \tag{4.4}$$

that can be solved in the least squares sense by solving the normal equations:

$$A^T A\boldsymbol{z} = A^T \boldsymbol{b} \tag{4.5}$$

The current version of the FIRES system offer other two different triangulation techniques, the *linear* and *midpoint* triangulations.

In the *linear* triangulation the equations of projection $\boldsymbol{m}_l = P_l \boldsymbol{p}$ and $\boldsymbol{m}_r = P_r \boldsymbol{p}$ are combined into a linear equation in the form $A\boldsymbol{p} = \mathbf{0}$, where

$$A = \begin{bmatrix} u_l \boldsymbol{p}_l^{3T} - \boldsymbol{p}_l^{1T} \\ v_l \boldsymbol{p}_l^{3T} - \boldsymbol{p}_l^{2T} \\ u_r \boldsymbol{p}_r^{3T} - \boldsymbol{p}_r^{1T} \\ v_r \boldsymbol{p}_r^{3T} - \boldsymbol{p}_r^{2T} \end{bmatrix} \tag{4.6}$$

where $\boldsymbol{p}_l^i$, $\boldsymbol{p}_r^i$ are the rows of P_l, P_r. The homogeneous system $A\boldsymbol{p} = 0$ can be solved for example by using the SVD decomposition of A.

While the linear triangulation works better respect to the previously used solution, it doesn't have a geometrical interpretation. A common and intuitive solution for solving the triangulation problem in a geometrical way is the *midpoint* triangulation.

Let us define the rays $\boldsymbol{r}_l, \boldsymbol{r}_r$ respectively from the two camera centers $\boldsymbol{C}_l, \boldsymbol{C}_r$ passing through the two image points $\boldsymbol{m}_l, \boldsymbol{m}_r$ as $\boldsymbol{r}_l = \boldsymbol{C}_l + t\boldsymbol{l}$, $\boldsymbol{r}_r = \boldsymbol{C}_r + t\boldsymbol{r}$ with $t \in [0, \infty)$. Considering noise and inaccuracies in the vision system, the two rays $\boldsymbol{r}_l, \boldsymbol{r}_r$,in general, do not intersect with each other.

The midpoint triangulation computes the point $\boldsymbol{p}$ that lies at the middle of the shortest line segment orthogonal to both rays. Knowing that the shortest line segment has direction $\boldsymbol{d} = \boldsymbol{l} \wedge \boldsymbol{r}$ we can define the two planes:

$$\pi_l(\boldsymbol{n}_l, \boldsymbol{C}_l), \boldsymbol{n}_l = \boldsymbol{l} \wedge \boldsymbol{d}, \tag{4.7}$$

$$\pi_r(\boldsymbol{n}_r, \boldsymbol{C}_r), \boldsymbol{n}_r = \boldsymbol{r} \wedge \boldsymbol{d} \tag{4.8}$$

then the points $\boldsymbol{p}_1, \boldsymbol{p}_2$ are respectively defined by:

$$\boldsymbol{p}_1 = (\pi_l \cap \boldsymbol{r}_r) = \boldsymbol{C}_r + \left(\frac{(\boldsymbol{C}_l - \boldsymbol{C}_r) \cdot \boldsymbol{n}_l}{\boldsymbol{r} \cdot \boldsymbol{n}_l} \right) \boldsymbol{r}, \tag{4.9}$$

$$\boldsymbol{p}_2 = (\pi_r \cap \boldsymbol{r}_l) = \boldsymbol{C}_l + \left(\frac{(\boldsymbol{C}_r - \boldsymbol{C}_l) \cdot \boldsymbol{n}_r}{\boldsymbol{l} \cdot \boldsymbol{n}_r} \right) \boldsymbol{l}. \tag{4.10}$$

The midpoint $\boldsymbol{p}$ is then obtained as the average $\boldsymbol{p} = \frac{1}{2}(\boldsymbol{p}_1 + \boldsymbol{p}_2)$.

The midpoint triangulation process is repeated to compute the 3D points $\bar{\boldsymbol{p}}_i$, $i \leq 4$ from the ordered pairs of corresponding image points provided by the cameras.

4.1.2 Tracking the pen-tip

The objective of the 3D tracking of the smart-pen tip is to associate to the 3D points $\bar{\boldsymbol{p}}_i$, $i \leq 4$, obtained by triangulating the ordered output of the tracking system, a label corresponding to a specific led emitter in order to locate the pen tip in the 3D space. Unfortunately, in presence of occlusions or interferences, it is difficult to distinguish between the light from the emitters on the smart-pen and interferences produced by the sunlight or the reflection of the emitters on the acquired object or on nearby surfaces. Moreover, it is also not trivial to distinguish among the emitters themselves.

At this aim we developed an interferences filtering and an emitter identification procedure which are executed after computing the 3D points $\bar{\boldsymbol{p}}_i$, rather then relying on classical filtering and identification methods performed on the 2D image plane. We can always identify which leds are visible when at most two leds are occluded, by comparing the real distances of the emitters mounted on the smart-pen with the computed relative distances among $\bar{\boldsymbol{p}}_i$. This relies on the fact that the distances between each pair of leds is different from each other (see Figure 4.3). Moreover, the pairs of 3D points for which the computed distance does not match with any of the real distances among the emitters, are considered as interferences and thus discarded.

Concerning with the smart-pen tip estimation we will proceed as follows. The redundancy of the number of leds on the smart-pen has been designed in order to guarantee that even in case of occlusion of the pen tip we are always able to estimate its position. This estimate can be evaluated by knowing at least the position of two leds which allow to identify the direction of the pen. The fourth led has been included to handle the occlusion of at most two leds.

In particular, given the distances d_1, d_2, d_3 among the four leds and the distance between the pen tip and the tip led (see Figure 4.3), we call l_1, l_2, l_3, l_4 the cumulative distances among the leds starting at the tip led, in other words the coordinates of the leds in a mono-dimensional reference system with center in l_1. In this reference system we have:

$$l_1 = 0 \ , \ l_2 = d_1 \ , \ l_3 = d_1 + d_2 \ , \ l_4 = d_1 + d_2 + d_3 \tag{4.11}$$

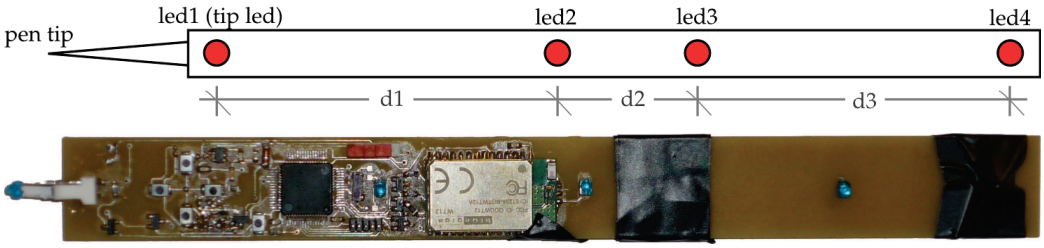


Figure 4.3: (top) A schematic representation of the smart-pen, (bottom) image of the smart-pen in case the pen tip coincides with the tip led

At each frame the number of led for which the 3D position in space is known depends on occlusions, reflections, matching errors or triangulation errors, and can vary frame by frame. Let's call δ_i the discrete value:

$$\delta_i = \begin{cases} 1 & \text{if i-th led is visible} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

For each quadruplet with at least two $\delta_i = 1$, which represent a particular configuration of at least two visible and identified leds, we preliminary compute and store the centroid $C \in \mathbb{R}$ of the led positions l_i, that is $C = \sum_{i=1}^{4} l_i \delta_i / \sum_{i=1}^{4} \delta_i$, and its distance T from the pen tip.

At each frame, we estimate the position $\boldsymbol{p}$ of the pen tip as

$$\boldsymbol{p} = \boldsymbol{C}' + \boldsymbol{v}_{pen}(C + T), \tag{4.13}$$

where $\boldsymbol{C}' \in \mathbb{R}^3$ is the centroid of the computed 3D points $\bar{\boldsymbol{p}}_i$, given by:

$$\boldsymbol{C}' = \frac{\sum_{i=1}^{4} \bar{\boldsymbol{p}}_i \delta_i}{\sum_{i=1}^{4} \delta_i},$$

and $\boldsymbol{v}_{pen}$ is the direction of the smart-pen. When more than two leds are visible, $\boldsymbol{v}_{pen}$ is given by the direction of the least squares line.

At any moment in time, due to occlusions, on each camera we could see 0 to 4 dots. In order to identify to which led a pair of visible dots belongs (if any), we triangulate the pairs and check if the distance between the computed 3D led positions (called *distSample* in Fig. 4.4) matches any of the 6 possible physical distances (called $readlDist_i$ in Fig. 4.4). Such procedure allow to identify the visible leds, compute their 3D positions,

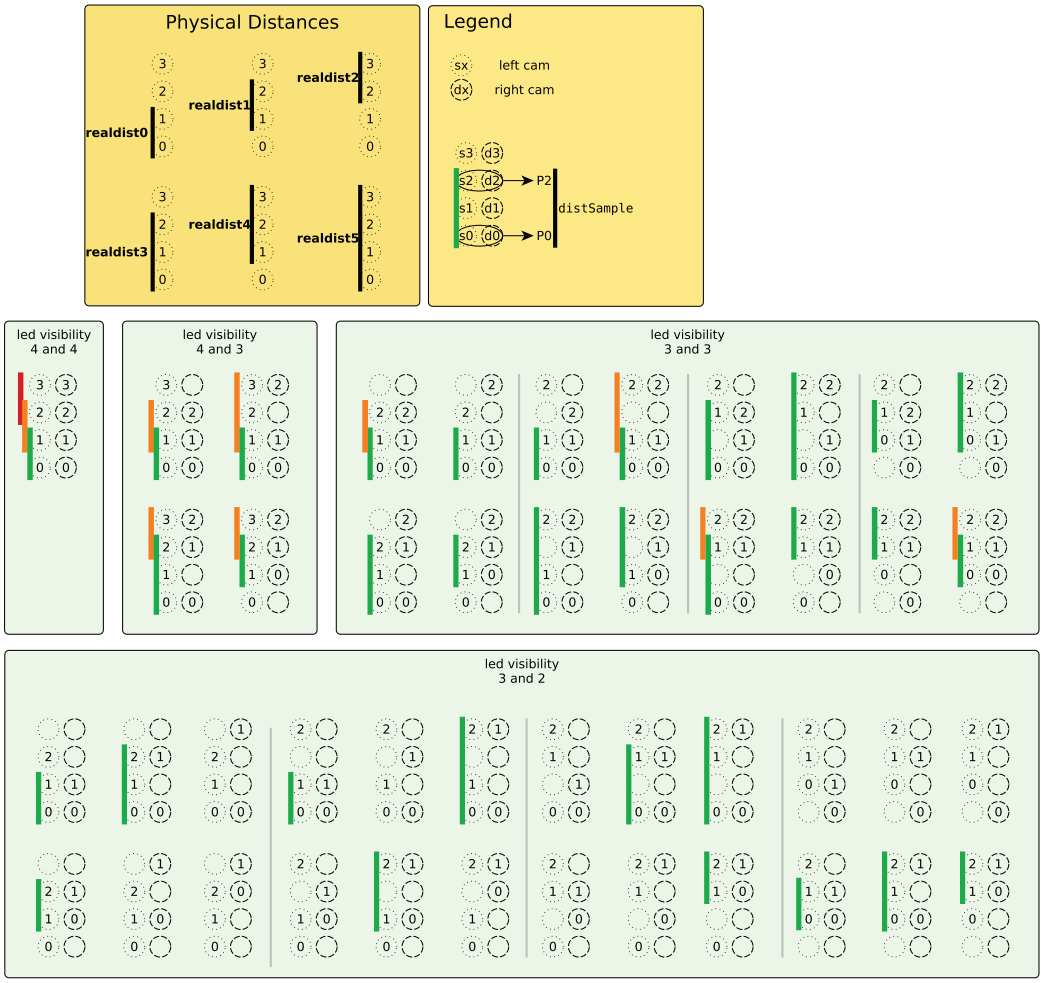


Figure 4.4: Led identification based on the visibility in the cameras.

and also allows to filter any interference due to external light sources or reflexes. All possible configurations of at least 2 visible dots per camera are listed in Fig. 4.4. For each category, we have to perform a variable number of triangulations to establish which led are actually visible.

4.1.3 Raw data filtering

The sequence of the tracked pen tip p positions gives rise to the curve drawn by the user. Unfortunately noise and inaccuracies in the acquisition procedure lead to a noisy curve which needs to be suitably filtered.

Noise and inaccuracies in the data received from the IR cameras are due to different factors but primarily related to the limited resolution of the camera devices and to the desynchronization between the two cameras i.e. pairs of image points on the two cameras are never captured in the same time instant. Both these factors strongly depend on the distance of the smart-pen from the camera and on its velocity. We addressed these problems through a real time smoothing and a post-processing step, the latter includes sampling, spline approximation and resampling.

Real Time Smoothing

The real-time smoothing is primarily executed to give the user a denoised visual feedback about a smoother curve. In the FIRES system the real time smoothing is performed by default using a convolution of the data with a Gaussian kernel. In the experimental results, shown in Section 7, we considered a discretization of the Gaussian kernel in 300 points and we repeated the filtering 3 times. The acquisition is performed at a frequency 100Hz. The temporal window in which the data is filtered could also vary and be chosen based on the expected distance of the work area from the cameras.

The acquired smoothed data is further post-processed by a sampling procedure in which a distance threshold is used to select the set of raw points which relative distance is higher than the given threshold.

The post-processed points will be the vertices of the new polyline included in the curve network. The effect of the post-processing steps can be appreciated in the example of Figure 7.2 by comparing the second with the third row where the curve networks before and after the post-smoothing are shown respectively. The weighted and constrained least squares spline approximation in the post-smoothing step guarantees the best fit to the manual sketch input given by the user while offering great flexibilities leaving to the system enough degrees of freedom to suitably filter the curve network.

4.1.4 Sensor Fusion

Sensor fusion means combining data generated by multiple sensors with the purpose of improving a measured quantity with respect to what we can achieve using the sensors by themselves.

In this section we will introduce the Kalman filter[53][104] and define multiple state-space models which allow us to combine inertial data coming from the pen sensors with vision data obtained from the stereo camera in order to improve the pen-tip position estimation.

The discrete Kalman filter

The Kalman filter addresses the general problem of trying to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{4.14}$$

with a measurement $z \in \mathbb{R}^m$ that is

$$z_k = Hx_k + v_k \tag{4.15}$$

The random variables w_k and v_k represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w)\ N(0, Q) \tag{4.16}$$

$$p(v)\ N(0, R) \tag{4.17}$$

The $n \times n$ matrix A in the difference equation (4.14) relates the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. The $n \times l$ matrix B relates the optional control input $u \in R$ to the state x. The $m \times n$ matrix H in the measurement equation (4.15) relates the state to the measurement z_k.

Sensor fusion through Kalman filter

To exploit the Kalman filter for sensor fusion we need to define a state-space model that combines information from the stereo-vision system and inertial data from the smart-pen sensors.

Since both the vision system and the inertial data provide us the pen direction, we can combine them through the equations

$$\begin{cases} \alpha(t) = \alpha(t-1) + \omega(t-1)dt + Q_\alpha \\ \omega(t) = \omega(t-1) + Q_\omega \end{cases} \tag{4.18}$$

$$\begin{cases} \alpha_v(t) = \alpha(t) + R_v \\ \alpha_a(t) = \alpha(t) + R_a \end{cases} \tag{4.19}$$

where $\alpha \in \mathbb{R}^2$ is the pen direction θ, γ (pitch e roll) while ω is the angular veolocity. The measurement equation (4.19) links the vision system measurement α_v with the inertial system measurement α_a through the state variable α.

We can now define the state transition matrix A and the measurement matrix H

$$\begin{bmatrix} \alpha_\theta(t) \\ \alpha_\gamma(t) \\ \omega_\theta(t) \\ \omega_\gamma(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} \alpha_\theta(t-1) \\ \alpha_\gamma(t-1) \\ \omega_\theta(t-1) \\ \omega_\gamma(t-1) \end{bmatrix} + Q \tag{4.20}$$

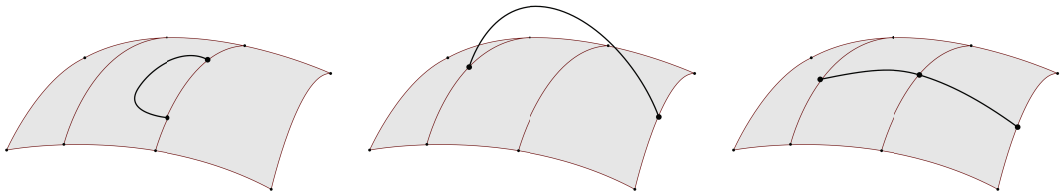
$$\underbrace{\begin{bmatrix} \alpha^v_\theta(t) \\ \alpha^v_\gamma(t) \\ \alpha^a_\theta(t) \\ \alpha^a_\gamma(t) \end{bmatrix}}_{m} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H} \begin{bmatrix} \alpha^v_\theta(t-1) \\ \alpha^v_\gamma(t-1) \\ \alpha^a_\theta(t-1) \\ \alpha^a_\gamma(t-1) \end{bmatrix} + R. \tag{4.21}$$

Chapter 5

Interactive Surface Sketching

The ISS represents the process that actively supports the user in drawing 3D curves. From the system point of view, the key feature of this process is to infere the topology and the connectivity of the mesh underlying the curve network from the curves drawn by the user. In this chapter we will first consider the ISS from the user perspective and then we will reconsider the same issues from the system internal logic point of view.

Aiming at the development of a powerful sketching design system usable through a simple and intuitive interface, we provide a limited number of tools which offer multiple functionalities transparently managed by FIRES. For example, while the creation of the boundary of the object or the closure of the mesh or the border gluing are very different operations in terms of modification of the mesh connectivity, from a user perspective they all appear as an operation of curve insertion and in fact all of them can be started by clicking the same button. Moreover, every tool has been designed to require the minimum user intervention to perform the operations on the polyline mesh. The low number of tools and their simplicity facilitate the user in the refining process and speed up the acquisition.



(a) (b) (c)

Figure 5.1: Curve insertion tool scenarios

5.1 ISS tools from the User perspective

FIRES put at user disposal different tools for ISS which allow the user to add new curves, create holes, connect more than one surface and create a surface by curve skinning. Starting from the basic curve insertion we describe the ISS tools in the remaining subsections.

5.1.1 Curve insertion tool

The basic tool used for interactively constructing the 3D virtual model of the object is the "curve insertion" tool that allows to intuitively draw new 3D curves by means of the smart-pen device.

Initially, the user traces a closed polyline which defines the boundary of the object to be reconstructed. Then the user updates the model by adding new curves. Each curve has to start and end on already given points of the curve network respecting the following rules:

1. the starting and ending points should not belong to the same polyline (see Figure 5.1a for a violation of this rule);
2. in case the drawn curve crosses another curve, then the two curves must intersect (see Figure 5.1b and Figure 5.1c for respectively a wrong and correct curve insertion).

To be precise, with intersection we mean an intersection between two curves under a defined distance threshold.

An example of reconstruction of an object using only the curve insertion tool is shown in Figure 5.2. In Figure 5.2(column a) three subsequent

curve insertion steps are shown while in Figure 5.2(column c) is shown the real time feedback that the system visualizes to the user at each step.

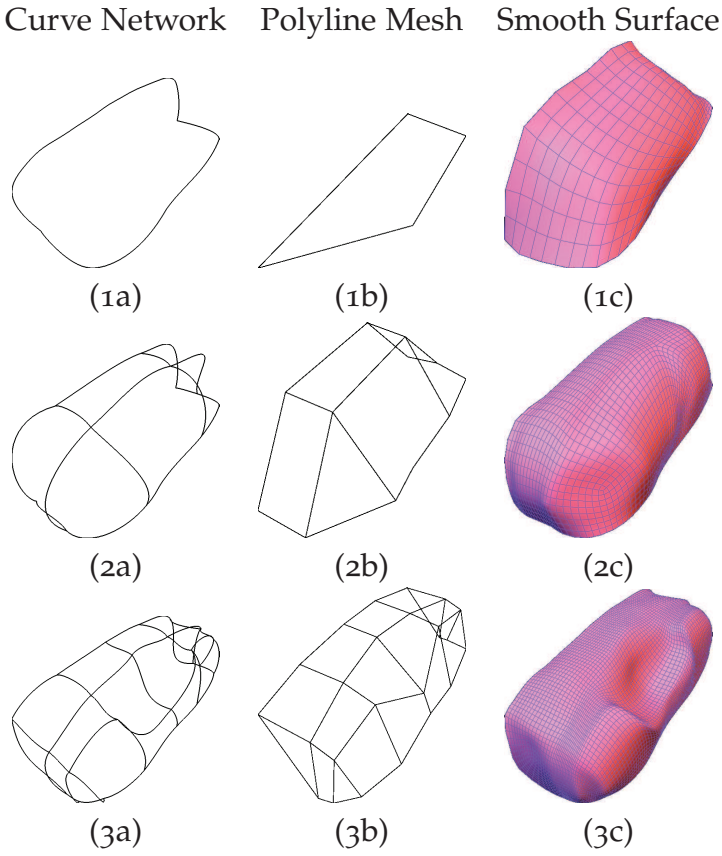


Figure 5.2: Example of successive updates of the curve network by the curve insertion tool (column a), the effect of the face splitting algorithm on the corresponding polyline mesh(column b) and the real-time visual feedback shown to the user (column c). The example refers to the physical object shown in Figure 7.2a

Moreover, FIRES offers also a curve delete tool and a mirroring tool for curves which replicates a curve with respect to a symmetry axis. The drawn curve is projected onto the least squares plane of the curve itself with the symmetry axis identified by the extremities of the curve thus determining a closed curve.

The major limitation of the curve insertion tool is that it allows only for building open orientable manifolds with genus zero and one border, i.e.

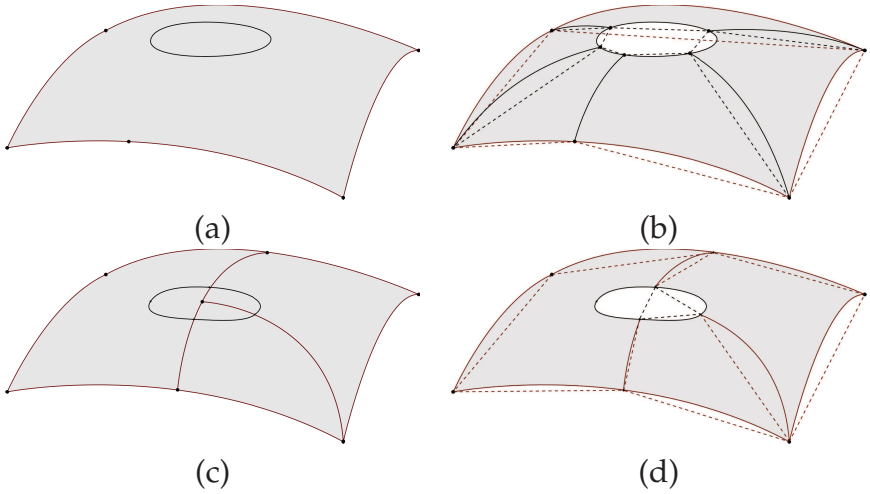


Figure 5.3: Hole creation tool: (a) (b) in case of zero intersections, (c) (d) with more than one intersections

the same topology of a single non-trimmed spline patch. For the reconstruction of objects with arbitrarily complex topology, we have to provide the user with other tools for executing different operations like creating holes, gluing borders, closing meshes etc.

For the modeling of arbitrary topology surfaces we need at least the following basic operations:

1. hole creation: which corresponds to a surface trimming by a single closed curve;
2. border gluing: which is used both for connecting two surface borders by a tubular surface (i.e. open, connected, oriented 2-manifolds with two boundaries) and for adding a handle to a single surface.

5.1.2 Hole creation tool

After the selection of the hole creation tool by keyboard or smart-pen button, the user proceeds with the drawing of a convex closed curve which represents the boundary of the hole. The drawn curve can intercept zero or more existing curves. Automatically, if necessary, the system connects the hole to the existing curve network by inserting n new curves. In case of zero intersections (see Figure 5.3 first row) n represents the number of

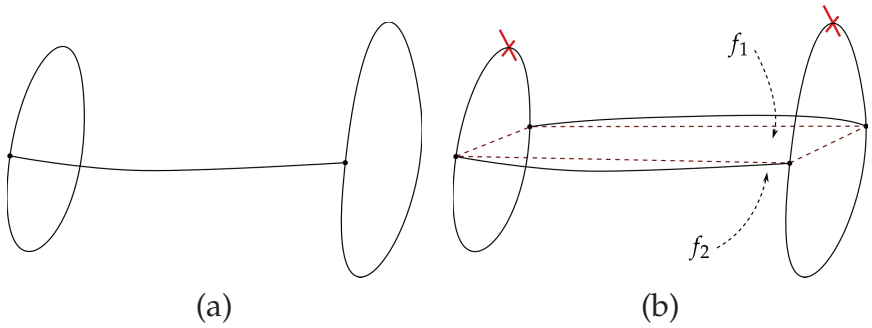


Figure 5.4: Border gluing tool

curves that enclose the hole ($n = 5$ in the example of Figure 5.3b).

In case of intersection with at least one existing curve, then $n = 0$, that is the system does not generate any new curve but simply connects the hole to the curve network using the parts of curves intersecting the boundary hole. We refer the reader to Figure 5.3, second row, for an example of hole creation in case of intersection with two already given curves.

5.1.3 Border gluing tool

The border gluing is one of the fundamental operations in the reconstruction process and allows for joining two boundary curves belonging to either one surface or two not-connected surfaces. A boundary curve can be defined as a boundary of the open manifold represented by the curve network.

From a user perspective the border gluing tool is very similar to a simple curve insertion. When the user draws a new curve that starts and ends on two different boundary curves (see Figure 5.4a), the system asks the user to insert the missing information to construct a tubular surface in-between. At this aim the user has to insert a second curve, which permits to separate in two halves the tubular surface, and select one curve on one border and the associated one on the other border which identify the same half. In Figure 5.4b the two selected half curves are marked by a cross symbol.

The border gluing can be used together with the hole creation tool to increase the genus of the surface by creating a handle. An example of

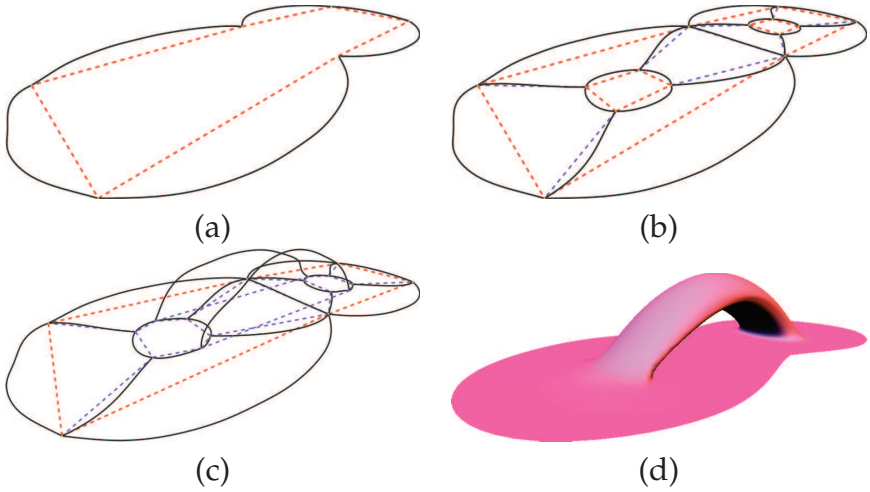


Figure 5.5: Example of handle creation using the hole creation tool (b) and border gluing tool (c)

handle creation is illustrated in Figure 5.5. Starting from the boundary curve in Figure 5.5a, the user traces two holes as shown in Figure 5.5b, then, using the border gluing tool, the user connects the holes, see Figure 5.5c. The system thus produces the result shown in Figure 5.5d.

5.1.4 Skinning tool

The skinning tool allows for the creation of tubular surfaces, using an approach similar to the procedure of skinning used for creating NURBS surfaces. The skinning tool involves the user in a two steps process in which the user first inserts $n \geq 2$ closed curves (called profiles), then draws 2 spine curves which intersect the same sequence of profile curves. In Figure 5.6 we illustrate the skinning process using two spine curves and three profile curves.

5.2 ISS tools internal logic

In this section we consider the ISS process from a system development point of view. Therefore all the tools described in Section 5.1 which affect

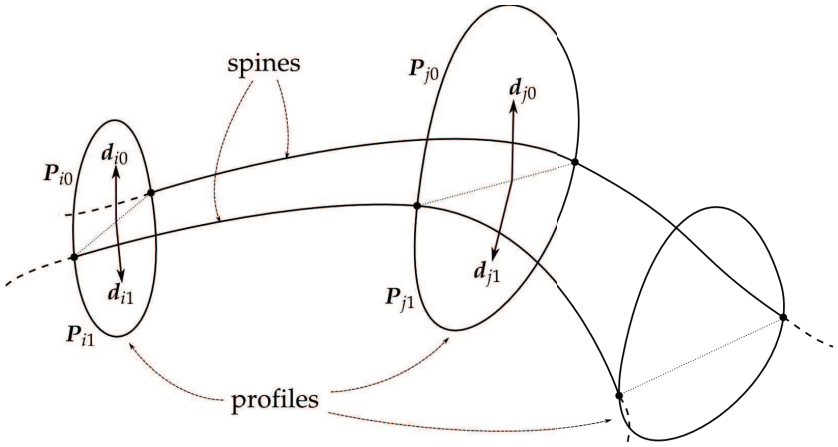


Figure 5.6: A schematic example of the skinning process

the curve network are revisited as operations on the associate polyline mesh.

5.2.1 Curve insertion internals

Using the curve insertion tool, the curve drawn by the user will be inserted into the polyline mesh data structure through a face splitting process. The face splitting requires that the starting and ending edges associated to the starting and ending polylines of the new drawn curve, lie on the same face f. The system splits the face f into two new faces associating the polyline inserted by the user to the new shared edge. Then the starting and ending edges, and the associated polylines, are split into two new edges and two new polylines. To be precise, the new polyline drawn by the user is approximated by a weighted and constrained least squares spline and then resampled before updating the polyline mesh.

In the simple case of zero intermediate intersections with other existing curves, the curve insertion tool leads to a single face splitting.

When the curve insertion involves a drawn curve which intersects more than one face of the polyline mesh, a sequence of face splitting steps is instead required and automatically performed by the system.

A simple example shown in Figure 5.7 illustrates the curve insertion tool in the latter case. Starting from the curve network (and its associated

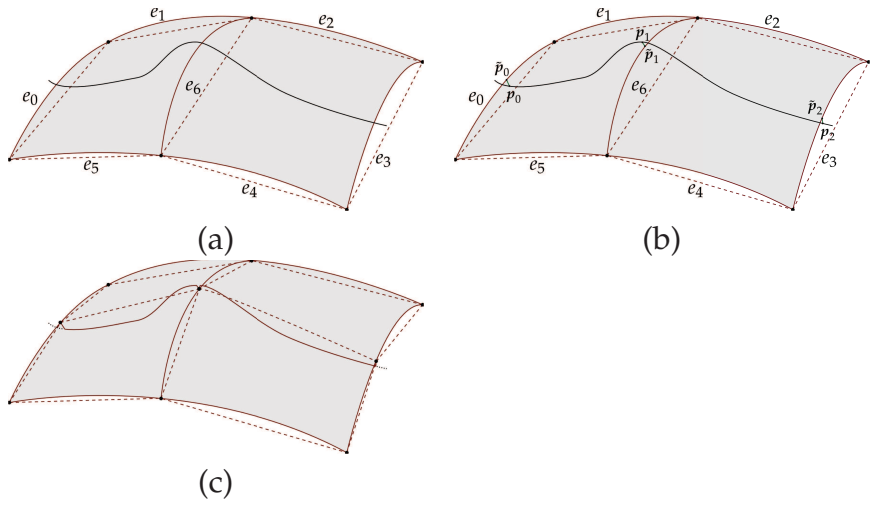


Figure 5.7: Curve insertion tool

polyline mesh) (Figure 5.7a) the user draws the curve represented in Figure 5.7a from e_0 to e_3. The system needs to perform two face splitting steps and thus needs to determine the intersections of the drawn curve with the existing polylines. The ordered sequence of these intersections defines the input data for each face splitting. In our example the curve crosses three different edges: e_0, e_6 and e_3 (see Figure 5.7a). In order to determine each intersection point, the system finds the pair (new point, old point) using a shortest distance criterion, where new point represents a point on the new inserted curve, while old point is a point on an existing polyline. In Figure 5.7b these pairs are $(p_0, \tilde{p}_0), (p_1, \tilde{p}_1), (p_2, \tilde{p}_2)$. These pairs can now be used to invoke the face splitting step passing to it subsequent pairs of points, the edges to split and their shared face in order to produce the polyline mesh shown in Figure 5.7c. In the Figure 5.2(column b) the effect of the face splitting algorithm on the polyline mesh is shown for the reconstruction of the object illustrated in Figure 7.2a.

When the user needs to insert a curve starting and/or ending on preexistent vertices FIRES offers a "vertex snapping mode" in which the system forces the intersection to coincide with the nearest vertex when the starting/ending/intermediate pair (new point, old point) is under a threshold distance from a vertex on the edge.

We observe that more sophisticated techniques for vertex picking can be

implemented; the reader is referred to [28] for details.

The curve insertion tool also handles the insertion of closed curves but only when this curve crosses at least one existing curve.

5.2.2 Hole creation internals

As described in Section 5.1.2, the hole creation procedure is performed differently according to the number of intersections between the hole boundary curve drawn by the user and the existing curve network.

In case of zero intersections the hole lies on a single n-sided face and, in order to automatically generate the needed polylines (see Figure 5.3b), the system determines the Coons patch $\boldsymbol{x}(u,v)$ associated to the n-sided face, as will be described in Section 6.1.2, and then computes the parametric coordinates $(\bar{u},\bar{v})$ associated to the hole centroid $\bar{\boldsymbol{C}} \in \mathbb{R}^3$. This is achieved by using a hybrid Newton method[29] to solve the nonlinear system:

$$\begin{cases} (\boldsymbol{x}(u,v) - \bar{\boldsymbol{C}}) \cdot \boldsymbol{x}_u(u,v) = 0 \\ (\boldsymbol{x}(u,v) - \bar{\boldsymbol{C}}) \cdot \boldsymbol{x}_v(u,v) = 0 \end{cases},$$

where $\boldsymbol{x}_u(u,v), \boldsymbol{x}_v(u,v)$ denotes respectively the partial derivatives of the Coons patch $\boldsymbol{x}(u,v)$ with respect to the parametric coordinates u,v. The distance between $\bar{\boldsymbol{C}}$ and $\boldsymbol{x}(u,v)$ is minimized when the nonlinear system is satisfied[103]. In case $\bar{\boldsymbol{C}}$ does not belong to the patch the procedure returns the parametric coordinates of the projection point of $\bar{\boldsymbol{C}}$ on $\boldsymbol{x}(u,v)$. The initial guess is estimated by evaluating the Coons patch $\boldsymbol{x}(u,v)$ on a regular grid and then performing a search for the point on the patch nearest to $\bar{\boldsymbol{C}}$.

In order to generate the n new polylines, the Coons patch is evaluated along the lines that connect $(\bar{u},\bar{v})$ to the parametric coordinates associated to each vertex of the n-sided face. Then, the system computes the intersections between the hole boundary curve drawn by the user and the generated polylines and creates n new edges from the vertices of the face to these intersections. Finally, the system updates the mesh connectivity deleting the old face and creating n new faces.

In case the hole boundary curve intersects existing polylines (see Fig-

ure 5.3c), then the system modifies the polyline mesh by splitting faces as in the case of curve insertion with multiple intersections, and by deleting the faces and edges internal to the hole curve (see in Figure 5.3d the updated polyline mesh).

5.2.3 Border gluing internals

As described in Section 5.1.3, the border gluing tool requires that the user draws two curves, and identifies the pair of associated pieces of curves to unambiguously determine the correct half of the tubular surface to be construct.

These are just the minimal data requested to the user to reconstruct the shape of a handle (or a connection between two borders), that is, the minimal data necessary to the system to build the correct underlying topology and connectivity without ambiguities. Afterward, more details can be added using the curve insertion tool and the other available tools. The system is able to reconstruct the tubular surface in-between by inserting two faces f_1, f_2 into the polyline mesh, as depicted in Figure 5.4b, that have exactly the same vertices but their edges have different associated polylines. Then, using this new polyline mesh the Basic refinement in the reconstruction step automatically builds the desired tubular surface.

5.2.4 Skinning tool internals

In FIRES the skinning surface can be interpreted as a sequence of tubular surfaces between consecutive profile curves. Therefore, as described in Section 5.1.3, we are facing with the same problem of identifying the two halves which form the correct tubular surface between consecutive profiles. Unlike from the border gluing tool, in this case the system does not require the user intervention, but it makes the assumption that successive profile curves lay on planes with dihedral angle $< 90°$. Exploiting this assumption we can construct the correct mesh connectivity using the following mechanism (see Figure 5.6). For each section between two consecutive profile curves $\boldsymbol{P}_i, \boldsymbol{P}_j$, the system computes the directions

$\boldsymbol{d}_{i0}, \boldsymbol{d}_{i1}, \boldsymbol{d}_{j0}, \boldsymbol{d}_{j1}$ of the half profiles $\boldsymbol{P}_{i0}, \boldsymbol{P}_{i1}, \boldsymbol{P}_{j0}, \boldsymbol{P}_{j1}$ as:

$$\boldsymbol{d}' := \frac{(\boldsymbol{p}_{start} + \boldsymbol{p}_{end})}{2} - \boldsymbol{p}_{middle} \quad , \quad \boldsymbol{d} = \frac{\boldsymbol{d}'}{\|\boldsymbol{d}'\|}$$

where $\boldsymbol{p}_{start}, \boldsymbol{p}_{end}$ and $\boldsymbol{p}_{middle}$ belong to each of the half profiles and $\boldsymbol{p}_{middle}$ is estimated by the cord-length strategy. The best candidate pair of associated half curves is the one with the maximum scalar product value among: $\boldsymbol{d}_{i0} \cdot \boldsymbol{d}_{j0}, \boldsymbol{d}_{i0} \cdot \boldsymbol{d}_{j1}, \boldsymbol{d}_{i1} \cdot \boldsymbol{d}_{j0}, \boldsymbol{d}_{i1} \cdot \boldsymbol{d}_{j1}$.

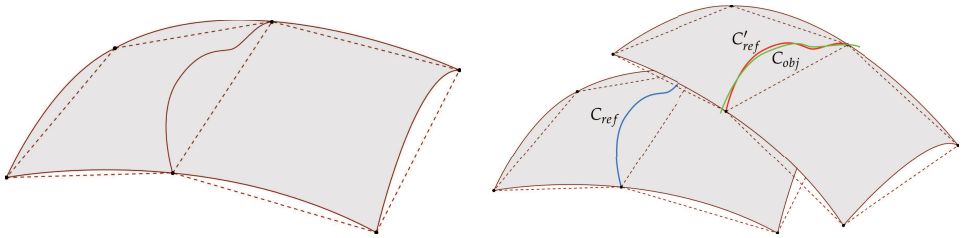
Results of skinning reconstruction can be appreciated in Figure 7.3 and Figure 7.4.

5.3 Alignment procedure

The alignment or registration procedure refers to the integration of multiple scans of the same object into a unique reference frame. The alignment procedure can be used both when multiple scans are acquired to reconstruct the whole object surface or when the user loads a previous working session. In these cases of reciprocal repositioning of the object and the stereo rig, the alignment is used to realign the reference frames in a common coordinate system in order to continue the work session. In literature different approaches exist, such as mechanical tracking, optical tracking, interactive alignment, and automatic alignment; we refer the reader to [10] for more details.

The alignment in FIRES is available through an interactive, semi-automatic procedure that allows for a real-time alignment which seamlessly fits into the ISS process and does not require any additional mechanical or optical device.

To perform the alignment the system requires that, when the object and/or the cameras have been relocated, the user selects a curve corresponding to an already inserted curve in a previous scan session. Suppose the user has drawn the simple curve network, shown in Figure 5.8a, and now, for avoiding occlusion or for other reasons, the user wants to move and rotate the object in order to complete the acquisition. To perform the alignment, the procedure requires that the user selects a curve C_{ref} on the display and



(a) (b)

Figure 5.8: An example of alignment procedure: (a) the original curve network; (b) the selected reference curve C_{ref}, the objective curve C_{obj} drawn on the moved object, the aligned curve network with the transformed curve C'_{ref}

then draws the corresponding curve C_{obj} on the moved object, as depicted in Figure 5.8c. The objective of the alignment procedure is to determine the rigid transformation M_{align} that will move and rotate the reference curve C_{ref} in a way to overlap the objective curve C_{obj}. The transformation M_{align} has to minimize the distances between corresponding points on the curve.

The solution used is based on the Principal Component Analysis and the Iterative Closest Point (ICP) algorithm[68] of the two curves interpreted as two sets of 3D points and the alignment matrix M_{align} is computed as follows:

1. consider the 4×4 homogeneous matrices $\mathcal{F}_{obj} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 & \bar{\boldsymbol{c}}_{obj} \end{bmatrix}$, $\mathcal{F}_{ref} = \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \boldsymbol{p}_3 & \bar{\boldsymbol{c}}_{ref} \end{bmatrix}$ where $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ and $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ are the eigenvectors of the covariance matrices associated to the sets of points of C_{obj} and C_{ref} respectively, and $\bar{\boldsymbol{c}}_{obj}, \bar{\boldsymbol{c}}_{ref}$ are the centroids of C_{obj}, C_{ref};
2. compute the transformation that takes the frame $\mathcal{F}_{ref}$ into $\mathcal{F}_{obj}$: $M_{align} = \mathcal{F}_{obj}\mathcal{F}_{ref}^{-1}$;
3. transform by the M_{align} matrix all the points of the curve network into the new aligned positions.

Actually, since the ordering of the points is unknown, we have eight different alignment transformation matrices that can be computed depending on the direction of the axis of the frames. To solve this issues the

algorithm computes all eight transformations M^i_{align} , $i \in 1,2,\ldots,8$ and choose the one that minimize the sum of squared distances of the curve C_{obj} from $C'_{ref} = M^i_{align} C_{ref}$.

The best alignment determined by this procedure could be the input guess of the ICP procedure. The current software implementation of the FIRES system consider this input guess as the final solution.

Chapter 6

Surface Reconstruction

As shown in Figure 3.2, the reconstruction process FIRES takes as input the polyline mesh produced by the ISS and builds on it a smooth surface using three steps. In the following sections we explain in details the reconstruction process used in the in FIRES^{V1} and then present the novelties introduced in FIRES^{V2}.

6.1 Triquadrification: from polyline mesh to Base Mesh

The ISS procedure produces a polyline mesh with some n-sided non-convex non-planar faces, with $n \geq 5$. The objective of the triquadrification step is to produce a mesh containing only 3 and 4-sided faces, choosing among all possible splittings of each n-sided face. Every n-sided face is thus split-up into n' faces and thus $n' - 1$ new polylines have to be generated. Independently on the chosen splitting criterion, these polylines will affect considerably the resulting surface because the Basic refinement step in the reconstruction pipeline interpolates some points on such polylines. At this aim the triquadrification procedure first associates to each n-sided face a single quad-face Q', then determines a good face splitting for the n-sided face, as described in Section 6.1.1, and finally generates the necessary new polylines associated to the new edges exploiting the parametrization of Q', as described in Section 6.1.2.

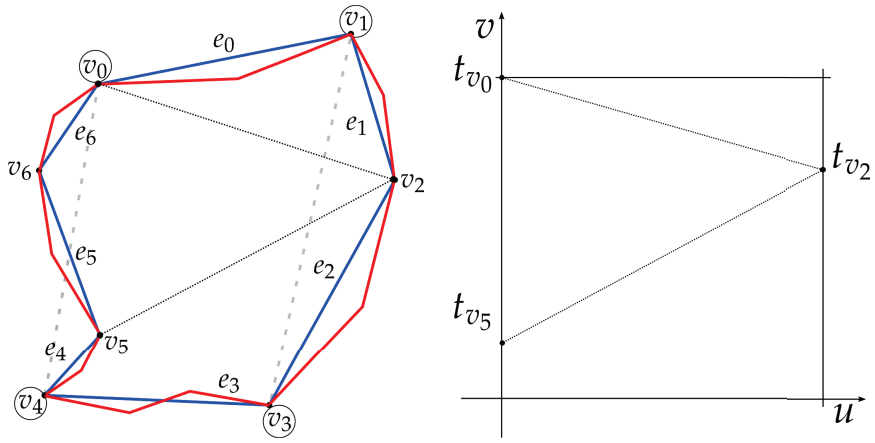


Figure 6.1: Triquadrification: (a) A 7-sided face. The quad-face Q' is identified by the circled vertices; (b) parametric domain of Q' and the parametric values associated with the vertices of the new polylines

The strategy to associate to each n-sided face a quad-face Q' is illustrated by the example in Figure 6.1 which maps the 7-sided face (see Figure 6.1a) into a quad-face Q' of vertices v_0, v_1, v_3, v_4. The system considers as vertices of Q' those in-between the pairs of edges whose internal angle is the smallest. Therefore, we have 4 resulting edges and associated polylines constructed by joining consecutive polylines curves, these are P_0, P_1 and P_2, P_3, P_4 and P_5 and P_6.

6.1.1 Finding a good face splitting

For each n-sided face we have to choose among all the possible splitting solution which is the best with respect to a chosen criterion. We could base our choice on different parameters, for example the planarity of the new polygons, their areas, the angles, the number of triangles versus quads, valence of vertices, etc. The criterion considered in FIRES is based on a measure of planarity computed as the angle between the normal vector of the least squares plane of the n-sided face and the normal vector of least squares plane of the tri-quad face under evaluation.

All the possible splittings generate a decision tree (see Figure 6.2 for an

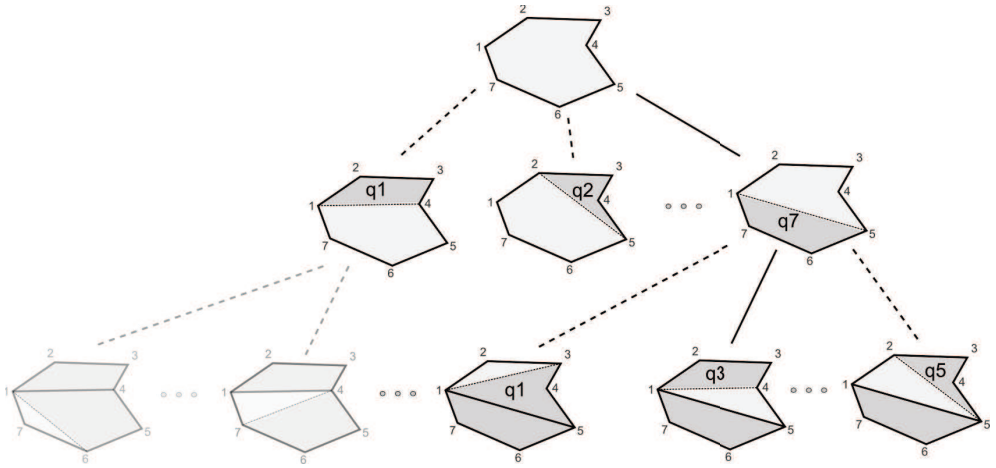


Figure 6.2: Starting from a 7-sided face as root, at the first level we have 7 different ways of splitting the face. From each one, the remaining 5-sided face could be split in 5 different ways with a total of 35 different face splittings

.

example) in which we search for a solution using a greedy search strategy which allows for an easy and fast implementation, very good performance and a sub-optimal solution. In particular, our search strategy works in a recursive fashion as follows.

Given an n-sided face, for each of the possible different 4-sided faces, each made of a sequence of 4 subsequent vertices which does not belong to the same edge of Q', the system computes the planarity measure and choose the best solution (see q_7 in the first level in Figure 6.2). In this example q_7 is chosen and the greedy search restarts with the remaining $(n-2)$-sided face (in the example 5-sided face). Recursion ends when the termination condition $n < 5$ is reached.

6.1.2 Generating new polylines

The method used for generating the new polylines exploits the Coons patches [38]. Coons patches are used to find a surface that interpolates four given boundary curves. In particular, the bilinearly blended Coons

patch is given by:

$$\begin{aligned} \boldsymbol{x}(u,v) &= \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(0,v) \\ \boldsymbol{x}(1,v) \end{bmatrix} \\ &+ \begin{bmatrix} \boldsymbol{x}(u,0) & \boldsymbol{x}(u,1) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} \\ &- \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(0,0) & \boldsymbol{x}(0,1) \\ \boldsymbol{x}(1,0) & \boldsymbol{x}(1,1) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} \end{aligned}$$

where $\boldsymbol{x}(0,v), \boldsymbol{x}(u,1), \boldsymbol{x}(1,v), \boldsymbol{x}(u,0)$ are four curves which form a closed loop.

Each polyline in the polyline mesh can be represented as a piecewise linear parametric curve defined in $[0,1]$ and the value of $P(\bar{t})$, where $\bar{t} \in [t_j, t_{j+1}] \subset [0,1]$ is computed by:

$$P(\bar{t}) = \boldsymbol{p}_j + (\bar{t} - t_j)(\boldsymbol{p}_{j+1} - \boldsymbol{p}_j).$$

In the example shown in Figure 6.1a a generic non-planar non-convex face with 7 edges $e_0, \ldots, e_6$ and 7 vertices $v_0, \ldots, v_6$ is shown together with its associated quad-face Q'. The triquadrification algorithm needs to generate the two new polylines from vertex v_5 to vertex v_2 and from v_0 to v_2. The new polylines will be obtained by evaluating a finite number of points on the constructed Coons patch $\boldsymbol{x}(u,v)$ associated with Q'. In the example of Figure 6.1b, we evaluate the patch $\boldsymbol{x}(u,v)$ along $t_{v_5}t_{v_2}$ and $t_{v_0}t_{v_2}$.

6.2 Basic refinement: from Base Mesh to Refined Mesh

The Base Mesh, output of the triquadrification process, is iteratively refined to produce the Refined Mesh exploiting Coons patches through the process that we called basic refinement. The objective is to obtain a polygonal mesh which best fits the curve network given by the user. Experimentally we estimated that 2 steps of basic refinement are sufficient to obtain a good Refined Mesh. In case the polylines associated to the edges

of the Base Mesh contain a very high number of points, then the system can perform more basic refinement steps.

Each basic refinement step consists in subdividing the face into 4 new 4-sided faces interpolating the central point of the bilinearly blended Coons patch and connecting this vertex to the middle points of each polyline thus creating 4 new edges. The 4 new polylines associated to the 4 new edges are then generated by evaluating the Coons patch in a number of points dependent on the number S of refinement iterations. In particular, at each refinement step s, $s = 1, 2, \ldots, S$, the patch is evaluated along the directions u, v in

$$\boldsymbol{x}(\frac{i}{n_s+1}, \frac{1}{2}), \boldsymbol{x}(\frac{1}{2}, \frac{j}{n_s+1}), i = 0, \ldots, n_s+1, j = 0, \ldots, n_s+1$$

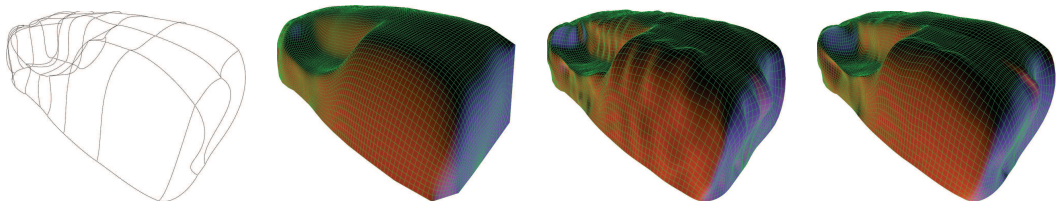
where n_s is recursively defined as

$$n_s = 2n_{s+1} + 1,\ n_S = 1.$$

The Coons patch can be adapted to the case of triangular faces by simply associating a degenerate edge and relative polyline to one of the vertices of the triangle. In this case we only need to create 3 new polylines from the central point of the patch to each edge of the triangle.

6.3 Subdivision refinement: from Refined Mesh to Smooth Surface

After the basic refinement step we obtained a Refined Mesh which is a good piecewise linear reconstruction of the shape defined by the curve network. The Refined Mesh consists of 4-sided faces with eventually extraordinary vertices, i.e. with more than four incident edges. Our next objective is to produce a smooth surface exploiting a subdivision surface scheme which naturally provide a unique representation of the arbitrary topology reconstructed object. For our reconstruction purposes we experimented with both interpolatory and approximating subdivision schemes. In particular, we used the well-known approximating Catmull-Clark subdivision scheme[19] and an adapted version of the NULISS in-



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

Figure 6.3: (a) The original curve network; (b) Catmull-Clark subdivision reconstruction; (c) NULISS reconstruction with polyline interpolation and the original face point rule; (d) NULISSmod reconstruction

terpolatory subdivision scheme[8]. While the former takes into account only the vertices of the Refined Mesh, the latter also integrates its associated polylines drawn by the user. This provides two different levels of control on the accuracy of the design process. Control curves used as handles for deformation right after their definition are approximated by the subdivision surface, while characteristic curves drawn to define specific features of the object to be reconstructed are interpolated. Moreover, the subdivision rules have been enriched by special rules for reconstructing sharp edges or corners.

In Figure 6.3 we show the results of the reconstructions obtained starting from the curve network in Figure 6.3a and applying respectively the Catmull-Clark subdivision scheme (Figure 6.3b), the NULISS subdivision scheme with polyline interpolation and the original face point rule (Figure 6.3c) and the modified NULISS subdivision scheme (Figure 6.3d).

6.3.1 NULISS Subdivision for polyline meshes

NULISS is the acronym for Non-Uniform Local Interpolatory Subdivision Surfaces. The NULISS subdivision process acts on quadrilateral meshes and, at each iteration, splits each existing face into four new faces, by inserting one vertex in correspondence of each vertex, at the middle of each edge and at the middle of each face of the current mesh. Thus, the newly inserted points are called either vertex, edge or face points respectively and for each of them we will have the corresponding insertion rule. After each refinement, the computed vertices are connected in the obvious way

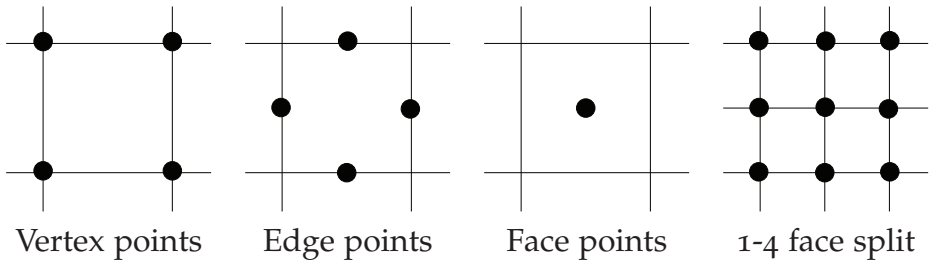


Figure 6.4: Points insertion and splitting of a single face through the NULISS scheme

to form four new faces out of each existing face, as illustrated in Fig. 6.4.

The NULISS algorithm includes refinement rules for both regular and extraordinary vertices, thus it works on meshes of arbitrary topology. Oppositely to other interpolating subdivision schemes, the NULISS refinement algorithm is non-uniform. This means that the refinement rules may vary from one region of the mesh to another and they are computed according to the local configuration of vertices so as to minimize interpolation artifacts.

While the NULISS subdivision scheme produces a surface that interpolates the vertices of the Refined Mesh, all information contained in the polylines associated to the edges are discarded. The FIRES allows for both using the NULISS subdivision scheme as originally proposed in [8], and its modified version for better exploiting all the information at our disposal. In fact we adapted the NULISS subdivision scheme, and in particular the rules for computing edge points and face points, in order to handle polyline points interpolation. We will refer to this modified NULISS scheme as NULISSmod.

The solution adopted for computing the edge point ep_i associated to the edge e_i in NULISSmod will first compute the edge point ep'_i, as usual, and then replace it with the nearest point on the polyline p_i associated to e_i.

The rule described for computing the face point fp_i associated to the face f_i is based on the position of the neighborhood vertices. If we don't take into account the modified edge points, the resulting surface does not fit well the shape defined by the underlying curve network, as can be seen in Figure 6.3c. To solve this issue we exploited Coons patches by

defining the face point fp_i through evaluation of the Coons patch in $\boldsymbol{x}(\bar{u}, \bar{v})$ where: $\bar{u} = \frac{1}{2}(t_{ep_0} + t_{ep_2}), \bar{v} = \frac{1}{2}(t_{ep_1} + t_{ep_3})$ and t_{ep_i} are the parameter values corresponding to each edge point.

Apart from the first subdivision refinement, the modified edge point rule is applied only to the polylines of the original Refined Mesh, not to the edges created in a subdivision step.

6.4 Surface Reconstruction improvements in FIRES^{V2}

FIRES^{V2} introduces a novel simple surface construction procedure based on functional optimization, which, for a given 3D curve network, automatically constructs a smooth surface preserving sharp features defined by the user.

The FIRES^{V2} surface construction step (see Reconstruction in Fig. 3.1b) is a multi-step process illustrated in Fig. 3.2b for a synthetic curve network with sharp (blue colored) curves and non-sharp (red colored) curves. The system first generates a Base Mesh from the polyline mesh, then a surface mesh constructor transforms the Base Mesh into a Refined Mesh taking into account also the user sharpness constraints, which can eventually be represented as Subdivision surface Mesh. Any of the different model representation forms (curve network/coarse mesh/surface mesh) can be integrated in a CAD system for further processing.

The basic refinement step first tessellates each n-sided face in four-sided polygons by following the same splitting rules as of a generalized Catmull-Clark subdivision, i.e., for each face a new vertex is placed at the face centroid and is connected with the midpoint of each edge. The tessellation can be repeated iteratively as necessary. The Base Mesh is however very inaccurate because it does not take into account the global shape of the model outlined by the curve network. Therefore, in the basic refinement step, we apply a functional optimization to transform the Base Mesh into a resulting refined quadrilateral mesh which interpolates points on the polylines and well represents the shape defined by the curve network. Eventually, the resulting mesh can be refined by a few steps of a subdivision scheme that produces a smooth surface interpolating or approximating the given curve network.

The approach we follow for surface construction is based on the surface diffusion flow

$$\triangle_{\mathcal{M}} H = 0. \tag{6.1}$$

Equation (6.1) can be derived as a simplification of the Euler-Lagrange equation (1.38) which results from minimizing the total curvature functional (1.31) which leads to a minimal energy surface.

The idea in fact is to produce the simplest shape which interpolates a given curve net. Moreover, since constant mean curvature surfaces satisfy equation (6.1), important basic shapes as spheres, cylinders and minimal surfaces with $H = 0$ can be reconstructed.

The resulting surface has to satisfy both geometric constraints, given by a set $\overline{X_0}$ of points on the 3D curve network, and sharpness constraints associated at each given curve, while preserving the topology defined by the polyline-mesh. The process of sharpness tagging is performed by simply selecting the curves on the object and a sharpness value.

In this section we present the construction process for a mesh M. Let X_0 be an initial surface which interpolates the set $\overline{X_0}$ of points on the 3D given curves and preserves the topology defined on the base-mesh, then we solve a global optimization problem by applying directly to the coordinate maps X the following fourth order flow

$$\frac{\partial X}{\partial t} = \triangle_{\mathcal{M}} \overrightarrow{H} + \lambda(X - \overline{X_0}), \quad X(0) = X_0, \tag{6.2}$$

where λ is a positive parameter which controls the effect of the data fidelity term that places positional constraints on all vertices of the 3D curves.

The construction method is based on a preliminary step (named basic refinement in Fig. 3.2b where we construct a sufficiently Refined Mesh X_0, which includes $\overline{X_0}$, by tessellating each polygon in the Base Mesh following the same splitting rules of the generalized Catmull-Clark subdivision and iterating so that each n-sided face is subdivided into $16n$ quads. The newly introduced vertices move according to (6.2) towards a minimal energy surface which satisfies the given geometry and feature constraints.

We propose a splitting strategy for solving the fourth order flow (6.2), which involves two coupled second order PDEs. The first PDE diffuses

the mean curvature normals on $\mathcal{M}$ preserving the sharp features defined on the curve network, and the second PDE refits the parameterization X according to the computed mean curvature distribution. In the first step we solve:

$$\frac{\partial \overrightarrow{H}}{\partial t} = \triangle^{w}_{\mathcal{M}} \overrightarrow{H}(X), \quad H(0) = H_0. \tag{6.3}$$

where $\triangle^{w}_{\mathcal{M}}$ is the weighted Laplace-Beltrami, whose weights are related to the sharpness of the curve network.

In the second step the obtained mean curvature vector field $\overrightarrow{H}$ is used to solve:

$$\triangle_{\mathcal{M}} X = -H\overrightarrow{N}(X) + \lambda(X - X_0), \tag{6.4}$$

where the data fidelity term forces the fitting of the $\overline{X_0}$ points on the 3D curve network.

To create sharp features along the curve network, a weighted Laplace-Beltrami operator is introduced in (6.3). The weights depend on a similarity measure between surface patches derived by the sharpness constraints along the curve network. The mean curvature diffusion is then penalized where creases and corners should be reproduced.

To this end, the interactive sketching process allows the user to specify a sharpness measure for each 3D curve of the curve network. Typically, it could be 0 or 1, where 1 indicates a sharp feature. The mesh edges constructed along sharp curves are labeled as sharp edges. Thus curves with vanishing sharpness values lead only to geometric constraints on the vertex positions, while curves labeled as sharp, lead to both vertex and curvature constraints. Let $S(X_i)$ be the sharpness of the vertex X_i defined as the sum of the number of adjacent sharp edges. The vertex X_i will represent a corner if $S(X_i) > 2$, a crease if $S(X_i) = 2$, or a dart if $S(X_i) = 1$.

Our proposal of similarity weight functions, based on sharpness values, defines the weights as follows

$$\begin{aligned} W_{ij} = &\quad \tfrac{1}{Z(i)} e^{-D(X_i,X_j)/\sigma^2}, \\ D(X_i, X_j) = &\quad |S(X_i) - S(X_j)|, \quad j \in N(i), \end{aligned} \tag{6.5}$$

where $N(i)$ is the set of 1-ring neighbor vertices of vertex X_i, and $Z(i)$ is

the normalizing constant $Z(i) = \sum_j e^{-D(X_i,X_j)/\sigma^2}$. The parameter σ controls how much the similarities of two patches are penalized. Larger σ gives results with sharper features. By using (6.5) the contribution of the vertices with different curvature features in $\triangle^w_{\mathcal{M}}$ is strongly penalized. Moreover, this is a rotationally invariant measure.

6.4.1 Discretization

The surface reconstruction algorithm iterates on the two steps implementing the PDEs (6.3) and (6.4), named in the sequel STEP 1 and STEP 2, converging to a mesh approximating the given 3D curve network and preserving the sharpness features defined by the user. From our experimental work we tuned up the maximum number of iterations to be between 5 and 10.

We discretize $\triangle^w_{\mathcal{M}}$ on M the *weighted Laplace-Beltrami operator* used in (6.3), analogously to (1.16) with weights w_{ij} replaced by

$$w_{ij}W_{ij}, \tag{6.6}$$

where w_{ij} is defined as in (1.18), while W_{ij} depends on a similarity measure between i*th* and j*th* vertex and are given by (6.5).

Since the mesh consists of quadrilateral faces, relation (1.28) is used to compute curvature normals $\overrightarrow{H}$, where the computation at each vertex involves a local triangulation suitably built around the vertex itself.

Considering a uniform discretization of the time interval $[0, T], T > 0$, with a temporal time step τ, then (6.3) can be discretized on time using the forward finite difference scheme which yields a first order scheme in time. Explicit time-stepping schemes are easily computable for every time-step τ, but the stability condition determines an upper bound for the time-step τ that guarantees stability of the evolution. The discrete elliptic operator of a fourth order problem is known to be characterized by a condition number $O(h^4)$, where h indicates the grid size. To ensure stability of an explicit discretization we would be led to a severe restriction of the type $\tau \leq Ch^4$ for the time step size τ [94]. Explicit methods therefore are computationally expensive.

Our two-step method reduces to the solution of two second order PDEs, and we used implicit time-stepping scheme in (6.3) which allows much larger time steps. Numerical experiments show that time steps of the order of the spatial grid size are still feasible with respect to the stability of the approach.

The PDE model (6.2) is then fully discretized and solved by iterating the alternate solutions of STEP 1 and STEP 2, as follows

STEP 1

$$(I - \tau L_w)\overrightarrow{H}^{n+1} = \overrightarrow{H}^n, \quad n = 0, 1, .., n_{MAX}$$

STEP 2

$$min_X \|LX - \overrightarrow{H}\|_2^2 + \lambda \|X - \overline{X_0}\|_2^2.$$

The alternating iterations represent a key aspect of the algorithm, as the initial rough approximation H_0 is improved at each step, providing a better accuracy to the entire process.

Decomposing L_w as $L_w = DL_w^s$, where L_w^s is the symmetric matrix derived from (1.16) with weights (6.6), and considering that W_{ij} in (6.5) are symmetric, then STEP 1 can be rewritten as the following symmetric definite positive system

$$(D^{-1} - dtL_w^s)\overrightarrow{H}^{n+1} = D^{-1}\overrightarrow{H}^n.$$

The resulting mean curvature normal vector field is then plugged into the constrained least square problem in STEP 2. The minimization problem in STEP 2 is rewritten as the following linear least-squares problem,

$$min_u \|Au - b\|_2, \tag{6.7}$$

where

$$A = \begin{pmatrix} L \\ C \end{pmatrix}, \quad u = \begin{pmatrix} X \\ P \end{pmatrix} \quad b = \begin{pmatrix} \overrightarrow{H} \\ X_0 \end{pmatrix} \tag{6.8}$$

$A \in \mathbb{R}^{(|X|+|\overline{X_0}|)\times|X|}$, $C \in \mathbb{R}^{(|\overline{X_0}|\times|\overline{X_0}|)}$ is the positional constraint matrix, rearranged as an Id matrix, b is the right-hand side, and u is the unknown vector, with the matrix P containing the $\overline{X_0} \subset X$ constraint vertices.

The construction algorithm implemented by alternating STEP 1 and STEP 2 is a global process which involves the solution of sparse but large linear systems whose dimension increases at increasing resolutions of the tes-

sellation. In order to further improve the system performance we applied STEP 1 and STEP 2 to a medium resolution base-mesh X_0. A finer resolution mesh is eventually obtained, after the Surface Diffusion Flow, by applying a few subdivision steps of a Catmull-Clark scheme, which also preserves sharp features by ad hoc subdivision refinement rules.

6.4.2 Generating new polylines in FIRES^{V2}

In Section 6.1.2 we presented a method for generating new polylines through sampling of Coons patches. Such need arises when new edges have to be created during the triquadrification phase, when splitting n-sided faces into triangles and quads, and during interactive surface sketching, when using the hole creation tool.

With the improved surface construction method presented in Section 6.4 we can simply generate the new polylines as linear interpolations of the endpoints of the new edge. The surface construction will seamlessly take care of evolving the shape of the polylines as points on the minimal surface, according to the properties of the proposed surface diffusion flow. Thanks to this improvement the implementation is more straightforward and the polyline generation process is more predictable.

Chapter 7

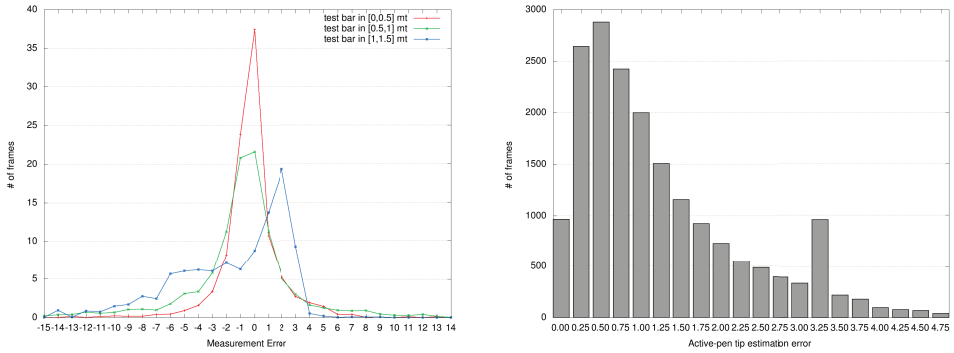
Experimental Results

7.1 Acquisition Results

Considering the real resolution of the wiimote camera image plane (u, v), 128×96 pixels, and the horizontal field of view, which is about $45°$, we can estimate approximately the maximum accuracy achievable during the 3D tracking. In fact, since each pixel "cover" an angle of $\theta_{pix} = \frac{45}{128}$ we can compute for each distance z from the camera the value $d_{pix} = z \tan(\theta_{pix})$ that roughly expresses the sensitivity of the camera, i.e. how many millimeters we can move in the x directions until a change in the u direction is detected. The value d_{pix} varies linearly in z and we have that, for example, at a distance of 0.5, 1, 1.5 and 2 meters from the camera the sensitivity is respectively about 3, 6, 9 and 12 millimeters.

The experiments showed that in practice we can achieve slightly better results in the 3D tracking. This is due thanks to both the virtual resolution (1024×768 pixels) that mostly helps when the leds cover more than one single pixel in the real image sensor, and thanks to the use of two wiimotes (so that the motion not detected by one wiimote could be detected by the other).

In the first experiment we measured with the FIRES acquisition system a test bar of width 81.0 millimeters with 2 IR leds mounted on the extremities. The length of the bar is then estimated as the 3D distance between the result of two triangulations of the two pairs of image points produced by the 2 IR leds on the 2 wiimote cameras. This estimate is acquired by



(a) (b)

Figure 7.1: (a) Measured error, in millimeters, at different distances from the cameras. Between 0 and 0.5 mt for a sequence of 2998 frames (red), between 0.5 and 1 mt for 7223 frames (green), between 1 and 1.5 mt for 4397 frames (blue). (b) Smart-pen tip estimation error in millimeters for about 18000 total frames

moving the bar in front of the cameras in the range $[0, 1.5]$ meters for about 15000 consecutive frames in order to compute the error as the difference between the width measured by the tracking system and the real width of the bar.

We consider the results grouped according to distances from the stereo rig. In particular, we considered when the test bar is less than 0.5 meter (see Figure 7.1a red line), between 0.5 and 1 meter (see Figure 7.1a green line), and between 1 meter and 1.5 meters (see Figure 7.1a blue line) from the cameras.

This experiment has demonstrated that, for guaranteeing an accuracy in the range $[-2; +2]$ millimeters for 75% of time/frames, the working area must not be behind 1 meter from the cameras.

In the second experiment we measured the accuracy of the method to estimate the smart-pen tip 3D position described in Section 4.1.2. The experiment aims at evaluating the error made when estimating the pen tip position in the case of occlusion of the tip led. The leds on the smart-pen used in this experiment are positioned at a distance of $d_1 = 70mm$, $d_2 = 40mm$, $d_3 = 50mm$. The experiment has been performed as follows. Moving the smart-pen in a range of $[0, 1.5]$ meters the system discards all frames where both cameras do not detect all four IR led emitters on the

smart-pen. Using an smart-pen where the pen tip coincide with the tip led, we can consider the tip led position $\bar{p}_1$ as the ground truth value for each frame. Forcing the system to detect only the three upper leds it performs the pen tip estimation using the formula (4.13). The histogram of Figure 7.1b shows the number of frames in which the error, computed as the distance between the estimated pen tip and the ground truth, assumes a given interval. This experiment demonstrates that the proposed method is able to estimate the pen tip 3D position with an error below 2mm for 82% of the analyzed frames. Moreover, bigger errors are automatically discarded by the FIRES led identification process.

Although we are not able to quantify the benefits in the use of the redundancies of the number of the leds, the application of FIRES to solve real reverse engineering problems has demonstrated a significant improvement in terms of system usability with respect to the case of a pen equipped with a single led.

7.2 FIRES^{V1} Reconstruction Results

The quality of the reverse engineering results strongly depend on the accuracy of our FIRES acquisition system. As said in Section 7.1 the FIRES acquisition system is capable of measuring distances in 3D with an error of around 2 millimeters if the object is located at most at 1 meter from the cameras. Unfortunately the limited resolution of the wiimote cameras used is not the only source of inaccuracy. In fact we have to face also with the limited human accuracy. Since the user controls the smart-pen we can not expect sub-millimetric precision.

In order to validate FIRES results related only to the reconstruction pipeline, without the eventual inaccuracies of the previous acquisition step, we generated the input as follows. Starting from a coarse mesh, a subdivision refinement is applied which generated a fine mesh useful to extract vertices for the synthetic polylines, then the coarse mesh with the associated synthetic polyline has been considered as the input polyline mesh for the reconstruction. These experiments has demonstrated very good quality reconstructions. An example is shown in Figure 3.2a for the reconstruction of a rock-arm dataset, the output of the single steps of FIRES

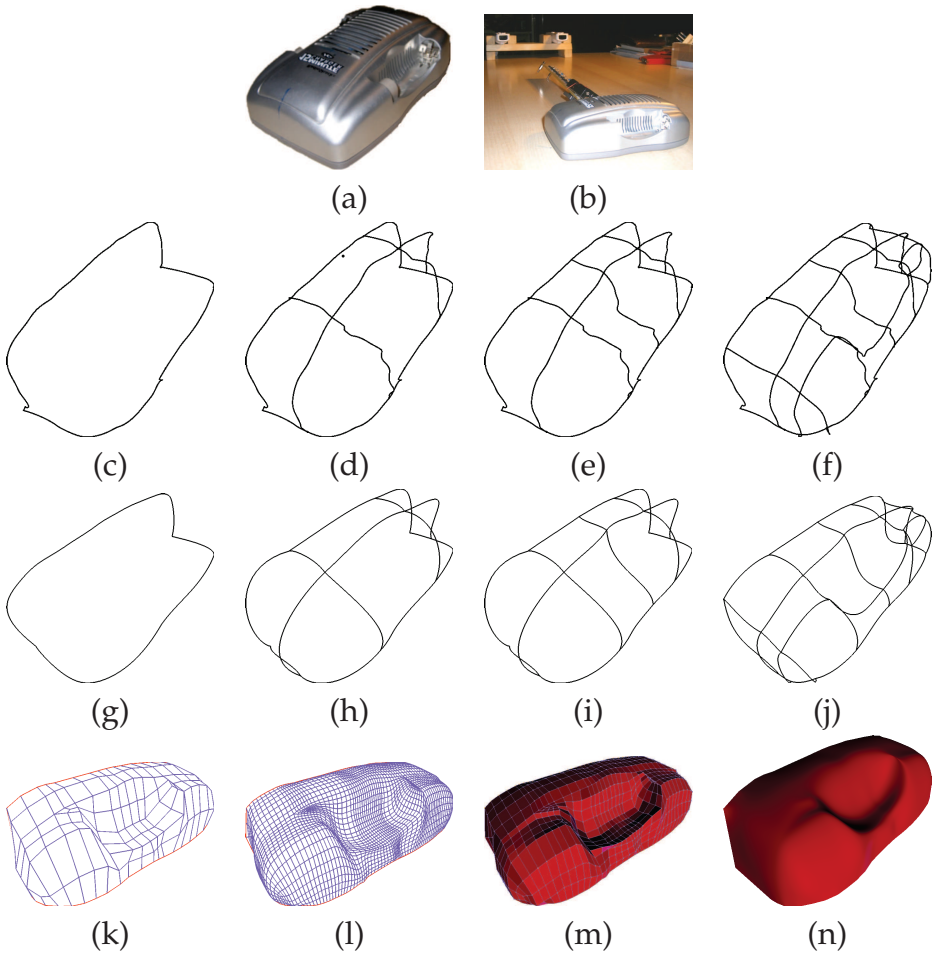
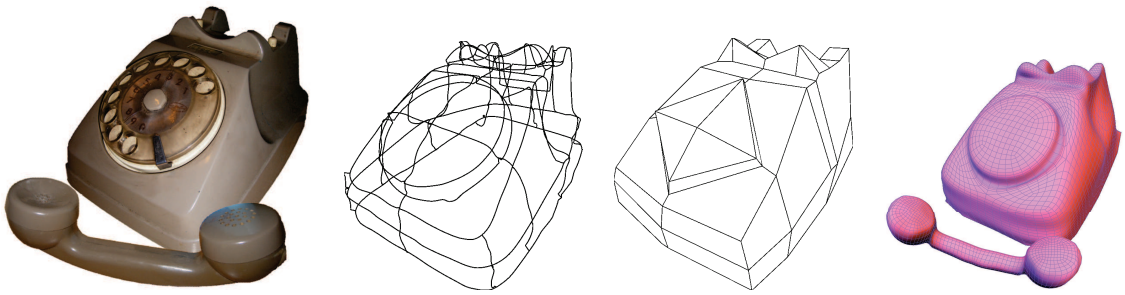


Figure 7.2: (a) (b) The physical object, the stereo setup and the smart-pen; (c) (d) (e) (f) The curve network after subsequent steps of ISS. (g) (h) (i) (j) The curve network after spline approximation. (k) The Refined Mesh after 1 step of basic refinement. (l) The first step of Catmull-Clark subdivision. (m) (n) The Refined Mesh and the smooth Catmull-Clark subdivision surface

reconstruction process are shown.

The first reconstruction experiment starts from the object (a battery charger) shown in Figure 7.2a. The object has been acquired in a single scanning session, without the need of repositioning the object or the cameras. Moreover, we exploited the symmetry of the object to only acquire a half of the object both for accelerating the sketching process and for ensur-



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

Figure 7.3: Reverse engineering process of an old telephone

ing that all the leds of the smart-pen will be visible most of the time to achieve the maximum accuracy. In Figure 7.2b the stereo setup used for the acquisition is illustrated together with the wiimotes in parallel stereo configuration, the prototype smart-pen and the object to be reconstructed. Some steps of the ISS process for the construction of the curve network of the battery charger are illustrated from Figure 7.2c to Figure 7.2f. From Figure 7.2g to Figure 7.2j the same curve networks are shown after the spline approximation. In Figure 7.2k the polyline meshes after one step of basic refinement is shown; while in Figure 7.2l the resulting Catmull-Clark subdivision refinement is given. The final virtual model in the form of a Refined Mesh is illustrated in Figure 7.2m, while the final smooth reconstruction is shown in Figure 7.2n.

In the second experiment we reconstructed the virtual 3D model of the old-style telephone and its telephone receiver illustrated in Figure 7.3a. The acquisition of the basement required a single alignment procedure while the telephone receiver has been reconstructed using the skinning tool and the mirroring tool. The curve network and the resulting polyline mesh of the basement are illustrated in Figure 7.3b and Figure 7.3c respectively. Using a hybrid subdivision procedure with 1 step of NULISSmod and 2 steps of Catmull-Clark scheme the resulting smooth surface is reported in Figure 7.3d.

In the third example FIRES is applied for the reconstruction of the vinegar bottle shown in Figure 7.4a. The skinning tool of the ISS is used to produce the curve network of Figure 7.4b which results into the corresponding polyline mesh of Figure 7.4c. Finally the subdivision result obtained by 3 Catmull-Clark subdivision steps is given in Figure 7.4d. In

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

Figure 7.4: Reverse engineering of a vinegar bottle

order to show the capability of FIRES to add virtual parts to an existing object, in Figure 7.5 we designed a handle to add to the 3D model of Figure 7.4d. At this aim, we first used the hole creation tool (curve network in Figure 7.5a and the associated polyline mesh in Figure 7.5b) and then we applied the border gluing to produce the handle (curve network in Figure 7.5c and the associated polyline mesh in Figure 7.5d). The final result after 3 steps of Catmull-Clark subdivision is shown in Figure 7.5e.

The tools for ISS in the acquisition phase are easily triggered through the buttons on the wireless smart-pen improving the system usability by requiring only one hand to perform both the 3D tracking and the interface with FIRES. Since the RE process in FIRES is interactive, the acquisition and reconstruction are real-time and the visual feedback is instantaneous the total time for a complete work session is primarily proportional to the complexity of the object to be reconstructed and to the accuracy required by the user.

7.3 FIRES^{V2} Reconstruction Results

In this section we provide some experiments to show the characteristics of our RE system integrated with the new surface construction system introduced in Section 6.4.

The outcome of the STEP 1 in the Surface Diffusion Flow depends on four parameters that in our examples revealed to be rather constants,

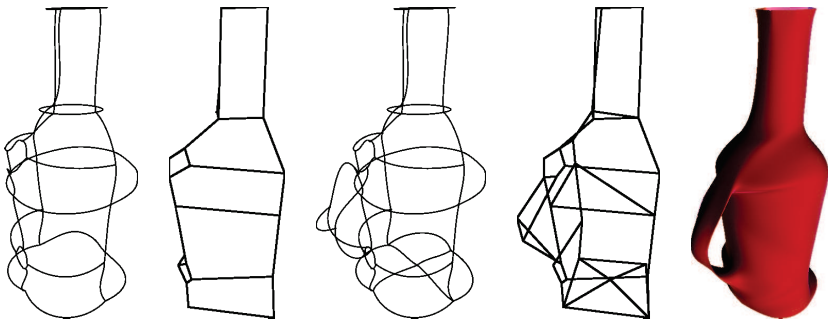


Figure 7.5: Adding a virtual handle to the geometric model illustrated in Figure 7.4

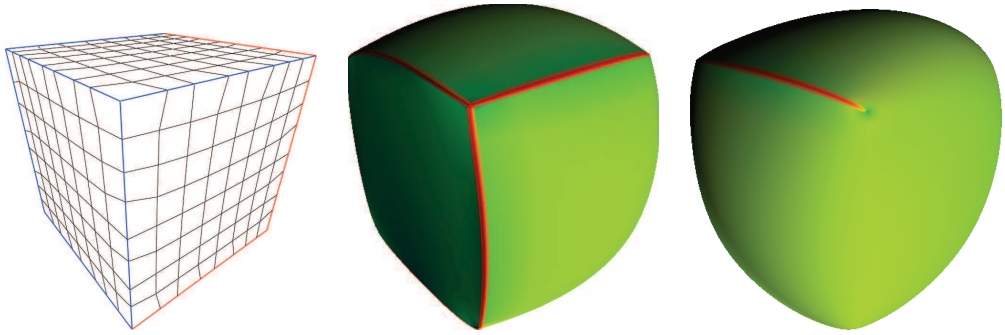


Figure 7.6: The reconstruction of sharp features on a cube-shaped synthetic curve network.

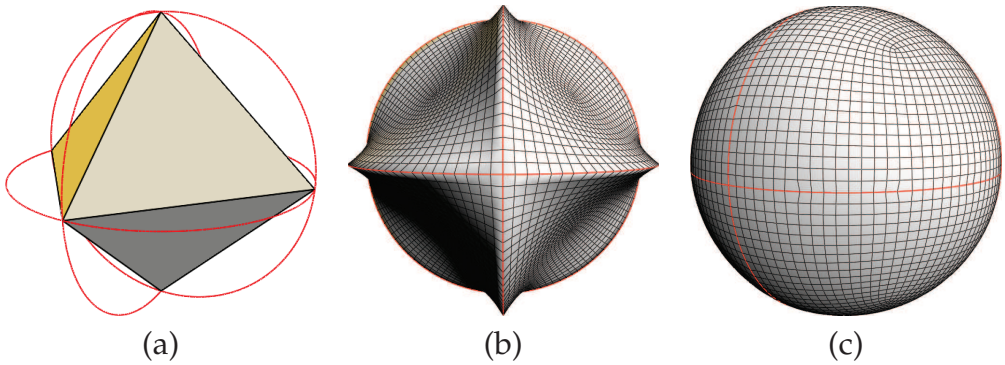


Figure 7.7: The reconstruction of a sphere starting from three orthogonal circumferences.

independently on the mesh. In particular, for optimal results the data fidelity parameter λ in (6.2) is set to be 1. The similarity parameter σ in (6.5) has been tuned to the value 0.2. The time-step τ is in the range [0.1,1.0] and $n_{MAX} = 5$.

Example 1: In the first example the reconstruction capabilities of our RE system are evaluated on three synthetic curve networks: the cross-shaped model in Fig. 3.2b, the sphere in Fig. 7.7 and the cube in Fig. 7.6.

The cross-shaped model in Fig. 3.2b presents different surface features (smooth curves, corners, edges and darts). The proposed reconstruction approach successfully creates an intuitive piece-wise smooth surface (Fig. 3.2b, bottom) starting from the simple 3D input curve network (Fig. 3.2b, top). The two intermediate steps show the Refined Mesh, output of the Basic Refinement process and the Refined Mesh after the Surface Diffusion Flow.

Similarly to the cross-shaped example, the cube in Fig. 7.6(a) presents different surface features. Due to the penalization introduced by the weights defined in (6.5) the face bounded by four sharp edges (marked in blue in Fig. 7.6(a)), is reconstructed into a planar surface (Fig. 7.6(b)). In Fig. 7.6(c) the reconstructed piece-wise smooth surface is shown from a different point of view where crease and dart features are well visible. The surface colors represent the surface curvatures, where green color indicates low curvature and red color high curvature values.

The example in Fig. 7.7 shows the ability of our method to precisely reconstruct a spherical surface starting from the three orthogonal circumferences and the underlying polyline mesh illustrated in Fig. 7.7(a). In this particular example we applied the Basic Refinement step to obtain a very Refined Mesh, shown in Fig. 7.7(b). The surface diffusion process evolves the Refined Mesh into the spherical mesh of Fig. 7.7(c), perfectly fitting the initial curve network.

Example 2: In the second experiment we show the result of the complete reverse engineering (RE) pipeline for two real objects shown in Fig. 7.8(a) and 7.9(a): an old-style telephone and a phone charger for a wireless telephone.

The reconstructed 3D virtual model (Fig. 7.8(e)) is the result of the multi-step reconstruction process consisting of the Basic Refinement step (Fig.

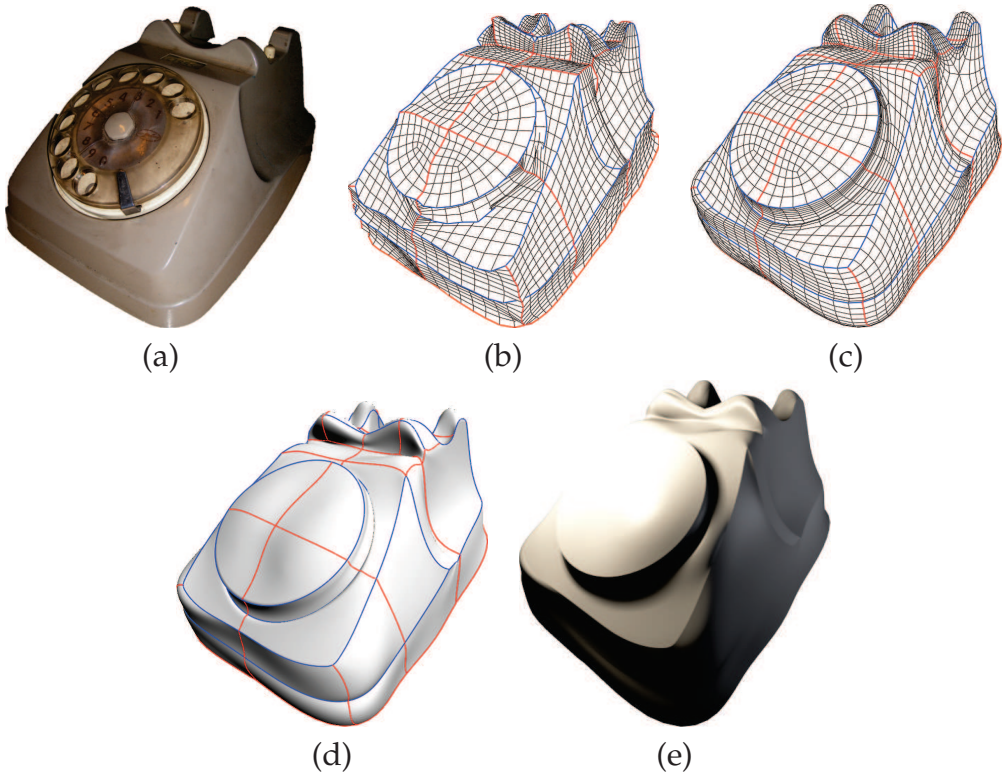


Figure 7.8: The reconstruction of an old-style telephone.

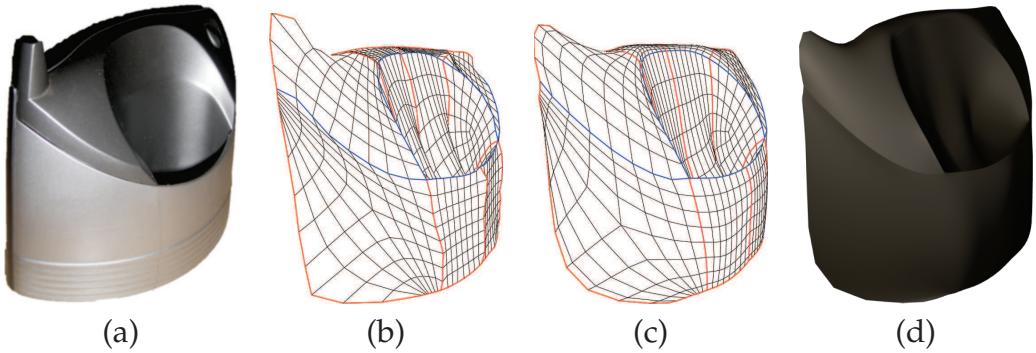
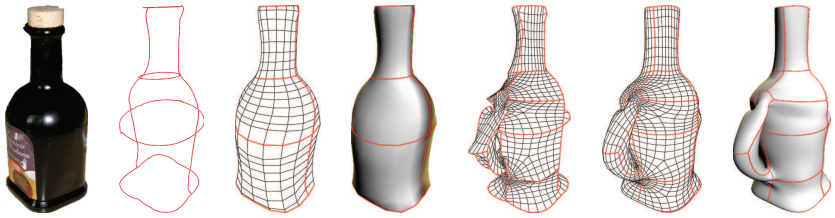


Figure 7.9: The reconstruction of an object with cavities.



(a) (b) (c) (d) (e) (f) (g)

Figure 7.10: The reconstruction of a bottle ((a)-(d)), and the result of adding a virtual handle to the bottle ((e)-(g)).

7.8(b)), the Surface Diffusion step (Fig. 7.8(c)), and the Subdivision Refinement (Fig. 7.8(d)). The final result of Fig. 7.8(e) clearly shows how a complex model with a variety of different surface features can be successfully reconstructed starting from a very simple and rough curve network (overlayed in Fig. 7.8(d)). The blue curves were marked as sharp during the RE session using our smart-pen sketching device.

The second reconstruction of the physical object shown in Fig. 7.9(a), followed the multi-step reconstruction process on a simpler physical object with a pronounced cavity, a feature that often creates problems with optical 3D scanner devices. Thanks to safe-occlusion characteristics of the smart-pen device, the user is able to sketch curves inside any cavities. The reconstruction system is then able to correctly build the smooth surface illustrated in Fig. 7.9(d). The intermediate reconstruction steps of Basic Refinement and Surface Diffusion Flow are shown in Fig. 7.9(b) and Fig. 7.9(c), respectively.

Example 3: In this example we describe an edit session performed using our RE system to interactively sketch new curves on the free space in order to add virtual parts to an object. In particular, in Fig. 7.10 the users draws the minimal amount of curves (Fig. 7.10(b)) necessary to reconstruct the basic shape of the bottle object in Fig. 7.10(a). The system provides a real-time visualization of the reconstructed smooth surface (Fig. 7.10(d)), created from the subdivision of the Refined Mesh of Fig. 7.10(c), output of the Surface Diffusion Flow.

The user then decides to add a handle to the virtual bottle. Using the "hole creation tool" and "border gluing tool" from the smart-pen sketching

interface (described in [9]) the user traces the 3D outlines representing the handle. The system reconstructs the smooth surface of Fig. 7.10(g) by first applying the Basic Refinement step (Fig. 7.10(e)) and then performing the Surface Diffusion Flow that produces the mesh in Fig. 7.10(f).

Conclusions and future works

In this dissertation, we focused on Geometric Surface Processing and Virtual Modeling topics.

In Part 1, we studied mathematical models based on Partial Differential Equations (PDE) applied to surface processing problems such as smoothing, remeshing, simplification and deformation. In Part 2, we introduced a Fast Interactive Reverse Engineering System enabling real-time acquisition and manipulation of complex geometrical shapes through wireless and interactive input devices. While Part 2 required a very technical and algorithmic focus, the contribution of the first part has instead required a deep study, understanding and research of new mathematical and numerical solutions. Moreover, from the documented studies in Part 1, we consolidated knowledge and tools that allowed us to tackle some of the problems we faced in Part 2.

In particular, we presented a novel smoothing method which is based on a two-step algorithm that solves a nonlocal surface diffusion flow PDE. The method is able to remove spurious oscillations while preserving and even restoring sharp features.

We also presented a new adaptive remeshing scheme based on the idea of improving mesh quality by a series of local modifications of the mesh geometry and connectivity. Our contribution to the family of parametrization-free remeshing techniques is a curvature-based mesh regularization technique which allows the control of both triangle quality and vertex sampling over the mesh, as a function of the surface mesh curvature.

Then we presented a new approach to simplification based on the evolution of surfaces under p-Laplacian motion. Such an evolution provides a natural geometric clustering process where the spatial effect of the p-

Laplacian allows for identifying suitable regions that need to be simplified. The concrete scheme is a multiresolution framework composed, at each simplification level, of a spatial clustering diffusion flow to determine the potential candidates for deletion, followed by an incremental decimation process to update the vertex mesh locations in order to decrease the overall resolution.

All the methods are based on solving evolutive PDE diffusion models and required the use of discrete differential geometry operator suitably weighted to preserve surface curvatures. The discretization lead to large linear system of equations solved by iterative numerical methods.

For each novel proposals we demonstrated that we achieve better results with respect to other well-known state of the art methods. As a drawback, in general, our differential models introduce a performance penalty due to the solution of large linear systems obtained by the discretization of the differential models. The prototypical numerical implementation realized in Meshviz can be definitely improved through advanced mathematical optimization techniques and GPU implementations.

Finally, the differential models for mesh deformation based on variational approach, gave preliminary satisfactory results, and certainly deserve future investigations. Moreover, most proposed methods could be generalized to tri-quad meshes to satisfy a broader range of applications.

In Part 2 we presented a novel system for reconstructing free-form surfaces from sketched irregular curve networks that consists in a basic first step which builds a low resolution base-mesh associated with the curve network, and a second step in which the base-mesh is refined to produce a smooth surface which preserves curvature features defined by the user on the curve network. While the system is not perfectly accurate as, for example, other, usually more expensive, laser scanning systems, it can be effectively used in 3D curve/silhouette sketching tools in CAD system environments. The developed project represents a low-cost solution to the challenging Reverse Engineering problem and we demonstrated that FIRES achieved an optimal compromise between accuracy and cost.

Several examples demonstrate the ability of the proposed method to produce complex 3D geometric models. A sensible improvement in the accuracy of the system can be obtained by using higher-cost hardware tech-

nologies such as higher resolution cameras. This would definitely improve the quality and stability of the signal, nevertheless, the system still suffers from intrinsic limitations due to human control of the pen device and limitations of active stereovision-based RE systems.

Appendix A: Meshviz

Meshviz User Interface

`Meshviz` is a GUI application developed in order to support experimentation of the geometric surface processing theories and models presented in Part 1.

The `meshviz` inteface is shown in Fig. 11 where we can see the main 4 areas:

1. The graphics window
2. The Display and Model menus in the left part of the window
3. The Selection Info tab and Log at the bottom.
4. The Main Menu bar at the top

The graphics window is an OpenGL view which shows the currently loaded mesh according to the view settings configured in the left Display tab. Navigation in the OpenGL view is performed via the three mouse buttons: left click and drag for rotation, right click and drag for zooming and middle-click and drag for panning.

Via the display tab, we can set the OpenGL camera to predefined views (free, rotating, top and side view). In the top right, the checkboxes allow to toggle the visibility of the mesh faces, edges, vertices, normals and other features.

The dropdown menu at the top-left allows switching the OpenGL rendering to predefined settings:

- Wireframe
- Normals

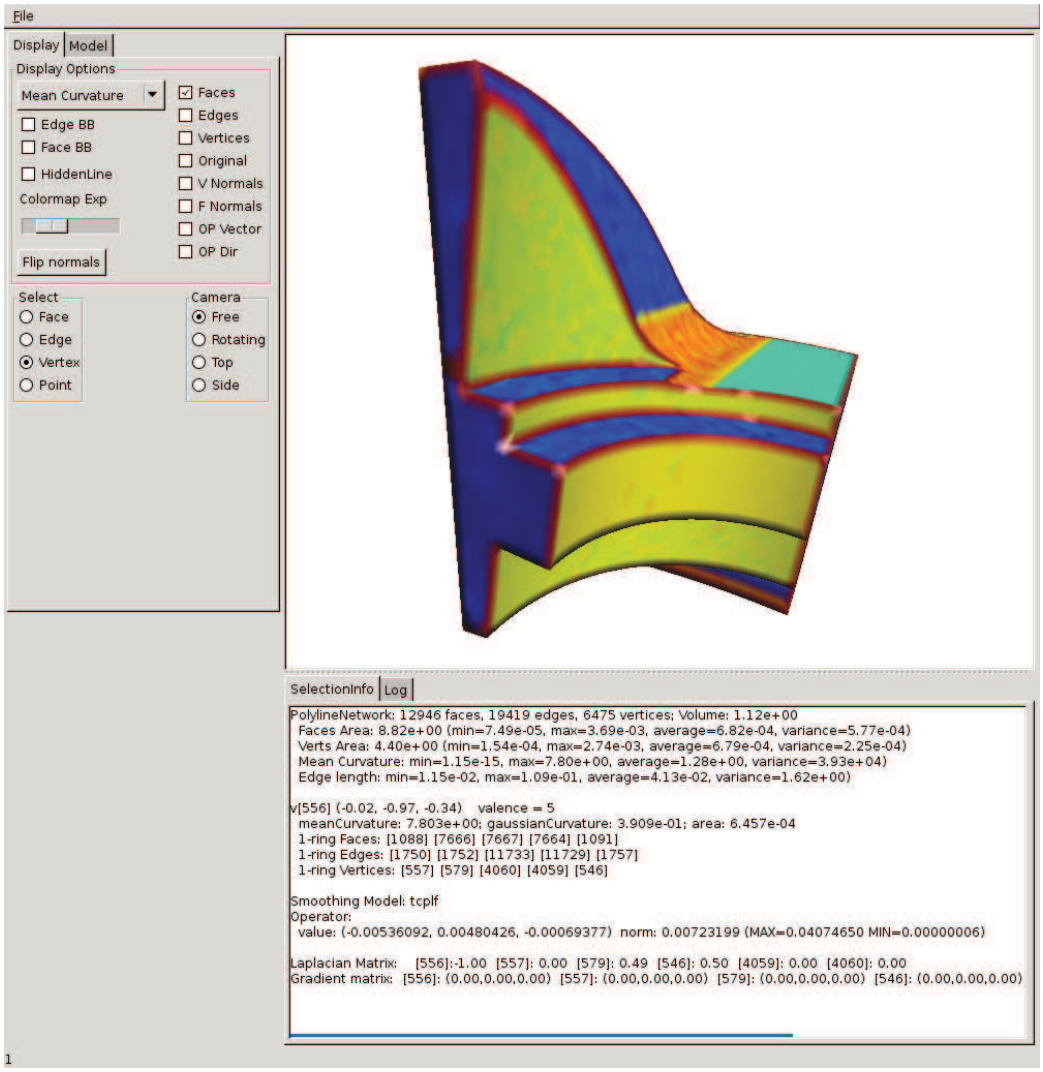


Figure 11: The `meshviz` interface

- Smooth
- Flat
- Operator
- Mean Curvature
- Gaussian Curvature
- Area

At the bottom, the SelectionInfo tab shows the mesh general properties, like Mean Curvature max, min, average and variance. After selecting a

vertex, it will also display specific vertex information like:

- mean curvature, gaussian curvature, area
- ID's of neighobrhood faces, edges and vertices
- the row in the laplacian matrix and in the gradient matrix corresponding to the selected vertex
- the value of the operator associated to the current model computed at the vertex position

The most important part of the UI is the Model tab.There, in the `App` dropdown menu we can select the primary application between Smoothing, Remeshing, Deformation and Reconstruction.

Once a specific application and a model has been selected, dynamically `meshviz` enables and disable specific control. Leaving the mouse for a couple seconds on a control shows a tooltip (when available) describing the effect of the control.

Development Envirnoment

Software development and experimentation for the implementation of the algorithm presented in the thesis and their results have been performed on different architectures ranging from low-end devices, such as a netbook with an Intel atom N270 processor (1.6GHz), 1GB RAM and an integrated graphic card, to more recent capable hardware such as a desktop pc with an Intel Core i7-2600 3.4 GHz processor, with 8 GB/RAM and an Nvidia GeForce GTX 560 Ti graphics card.

The visualization is based on the OpenGL graphics library and the applications are written using the C language, and occationally using C++. Numerical computations are performed using the "meschach" matrix library. Various small POSIX shell or python scripts have been developed to support experimental data collection and visualization.

Simplifcation in `Meshviz`

The current version of `meshviz` does not offer interactive experimentation with the simplification algorithm proposed in section 2.4. Instead a command-line application called `simplify` has been developed. Thanks to `simplify` and shell scripting, experimentation and data collection has been completely automatized.

The `simplify` program has the following syntax.

```
usage: ./simplify --in FILE --percentage NUM [--iterations NUM] [--smoothing-iterations NUM]
               [--ple FLOAT] [--lambda FLOAT] [--exp-length FLOAT] [--exp-garland FLOAT]
               [--exp-curvature FLOAT] [--exp-valence FLOAT] [--exp-curvdiff FLOAT]
               [--no-adaptive] [--out FILE] [--no-smoothing] [--no-decimation] [--no-sort]
               [--keep-mc] [--no-optimal] [--dist] [--help]
```

The parameters are described by running `./simplify --help`:

```
parameters:
 --in mesh.obj
    Specify the input mesh in obj format.

 --percentage N
    The percentage of edges to remove.

 --iterations N
    The number of "outer" iterations of the simplification algorithms. (Default: 2)

 --smoothing-iterations N
    The number of "inner" p-laplacian smoothing iterations. (Default: 3)

 --ple FLOAT
    The value of the p-laplacian exponent. (Default: 0.100)

 --lambda FLOAT
    Defines the fidelity parameter lambda in the solution of the smoothing model. (Default: 1.000)

 --exp-length FLOAT
    Defines how much the edge length influences the edge shape factor. (Default: 0.000)

 --exp-garland FLOAT
    Defines how much the garland quadric error influences the edge shape factor. (Default: 0.000)

 --exp-curvature FLOAT
    Defines how much the edge curvature influences the edge shape factor. (Default: 0.000)

 --exp-valence FLOAT
    Defines how much the edge valence length influences the edge shape factor. (Default: 0.000)

 --exp-curvdiff FLOAT
    Defines how much the edge local curvature difference influences the edge shape factor.
    (Default: 0.000)
```

```
--no-adaptive
   Disable the curvature adaptivity of the fidelity parameter lambda in the solution of
   the smoothing model.

--no-sort
   Disable sorting of edges in decimation. Enable threshold-based decimation.

--no-smoothing
   Do not perform the smoothing step in each simplification iteration.

--no-decimation
   Do not perform the decimation step in each simplification iteration.

--keep-mc
   Keep original curvatures after collapse without recomputing them after each outer iteration.
   (Default: disabled)

--no-optimal
   Keep original curvatures after collapse without recomputing them after each outer iteration.
   (Default: optimal)

--dist
   Will compute and print the hausdorff distance at the end of the simplification.

--out FILE
   Secify the output mesh file.
```

For example, the following command:

```
example: ./simplify --in dinosaur.obj --percentage 80 --iterations 2 --out simplified.obj
```

simplifies the `dinosaur.obj` mesh by 80% in two levels, producing an output mesh called `simplified.obj`.

Appendix B: Smart-pen

The smart-pen has the following hardware characteristics:

- MCU: ST Microelectronics STM32F103VE
- Accelerometer: ST LIS3LV02DL
- Gyroscope: ST L3G4200D
- Magnetometer: Freescale MAG3110
- Pressure sensor: NOT present in current modules.
- Bluetooth: BlueGiga WT12
- Internal NAND flash memory (8 GBit)
- 3 color LED (red/green/yellow).

The device can be turned on by pressing the button. Once turned on, the green led will turn on, then blink a couple of times during initialization and then stay on while in the IDLE state. In the idle state there is a 2 minute time-out, after the expiration of which the device will turn off. It is possible to turn off the device from every state by a long press of the button (about 5 seconds). In both cases the led will turn off, signaling that the device turned off. Once connected to the host, the device can receive commands.

After receiving via bluetooth the command to start streaming (BT_START) the green led blinks slowly to signal the operations. Operations can be terminated by a stop command (BT_STOP).

The device provides a Serial Port Protocol (SPP) service, and the host, after pairing with the code 1234, can communicate with the device through the corresponding serial port. The serial port settings are $230400, 8, 1, n$ but being a virtual serial port emulated via Bluetooth, the Bluetooth stack handles all the connection-related issues.

Table 1: Smart-pen packet format.

Byte 0:	`0x20`
Byte 1:	`0x0A`
Byte 2:	Packet Counter
Byte 3-8:	Accelerometer a_x, a_y, a_z
Byte 9-14:	Gyroscope $\omega_x, \omega_y, \omega_z$
Byte 9-14:	Magnetometer $\gamma_x, \gamma_y, \gamma_z$
Byte 21:	CheckSum

The device can receive the following commands, which are ASCII characters sent via the serial connection.

Command (hex)	Name	Description
= (`0x3D`)	`BT_START`	Starts sampling and streaming data
: (`0x3A`)	`BT_STOP`	Stops sampling and streaming data

When streaming data from the devicethe sensor data are organized in packets of 22 bytes as in Table 1. The first two bytes are fixed to identify the packet. The Packet Counter is an 8 bit unsigned integer, incremented by the device every packet transmitted. All the sensor data are represented by 16-bit signed integers. At the end the checksum is computed by and ex-or of all the previous bytes.

Bluetooth communication

The Bluetooth Protocol Stack (see Figure 7.3) is a set of open protocols for exchanging data over short distances from fixed and mobile devices. The responsibilities of the layers in the bluetooth stack can be summarized as follows:

The **radio** layer is responsible for transmitting and receiving packets of information on the wireless physical channel. The radio layer is controlled by the **baseband** layer, which controls and sends data packets over the radio link providing transmission channels for Synchronous Connection-Oriented (SCO) links and Asynchronous Connectionless (ACL) links. The

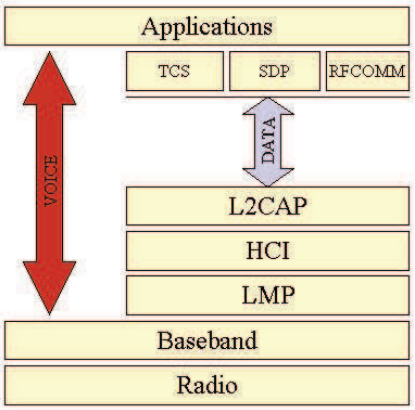


Figure 12: The bluetooth protocol stack

ACL links[1] are point-to-point symmetric connections commonly used for data transfers.

The Link Manager Protocol (**LMP**) uses the links set up by the baseband to establish connections and manage networks of bluetooth devices (called piconets). Responsibilities of the LMP also include authentication, security services, and monitoring of service quality.

The Host Controller Interface (**HCI**) is the dividing line between software and hardware. While the L2CAP and layers above it are usually implemented in software, the LMP and lower layers are in hardware. The HCI is the driver interface that connects these two components. The HCI may not be required.

The Logical Link Control and Adaptation Protocol (**L2CAP**) receives application data and adapts it to the Bluetooth format. Quality of Service (QoS) parameters are exchanged at this layer. The L2CAP may be accessed directly by the application, or through certain support protocols provided to ease the burden on application programmers.

An example of a support protocol is the **RFCOMM** protocol, a simple transport protocols made on top of the L2CAP protocol that provides emulated RS-232 serial ports via a reliable data stream similar to TCP.

RFCOMM is the protocol used by the bluetooth controller mounted on the

[1]Besides their misleading name, Asynchronous Connectionless links are not connectionless

smart-pen and we use it to connect to the device and exchange data using the C programming language and the blueZ stack on a linux platform.

Bibliography

[1] M. Ainsworth and D. Kay. The approximation theory for the p-version finite element method and application to non-linear elliptic PDEs. *Numerische Mathematik*, 82(3):351–388, 1999.

[2] P. Alliez, E. C. de Verdiere, O. Devillers, and M. Isenburg. Isotropic Surface Remeshing. *Shape Modeling International (SMI)*, pages 49–58, 2003.

[3] P. Alliez, E. C. de Verdiere, O. Devillers, and M. Isenburg. Centroidal Voronoi Diagrams for isotropic surface remeshing. *Graphical Models - Special issue on SMI 2003*, 67(3):204–231, 2005.

[4] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. *ACM Transactions on Graphics (TOG)*, 21(3):347–354, 2002.

[5] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. *Shape analysis and structuring*, pages 53–82, 2008.

[6] E. Bae and J. Weickert. Partial Differential Equations for Interpolation and Compression of Surfaces. *Mathematical Methods for Curves and Surfaces (MMCS)*, pages 1–14, 2010.

[7] R. Baron and R. Plamondon. Acceleration measurement with an instrumented pen for signature verification and handwriting analysis. *IEEE Transactions on Instrumentation and Measurement*, 38(6):1132–1138, 1989.

[8] C. V. Beccari, G. Casciola, and L. Romani. Interpolatory surface subdivision based on geometry-driven parameterizations. *Minisymposium on New Trends in Numerical Approximation and Applications, SIMAI*, 2008.

[9] C. V. Beccari, E. Farella, A. Liverani, S. Morigi, and M. Rucci. A fast interactive reverse-engineering system. *Computer-Aided Design*, 42(10):860–873, 2010.

[10] F. Bernardini and H. Rushmeier. The 3D Model Acquisition Pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

[11] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. In R. Martin, H. Bez, and M. Sabin, editors, *IMA international conference on Mathematics*, volume 3604 of *Lecture Notes in Computer Science*, pages 62–83, Berlin, Heidelberg, Sept. 2005. Springer Berlin Heidelberg.

[12] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. *Eurographics ACM SIGGRAPH symposium on Geometry processing (SGP)*, pages 185–192, 2004.

[13] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. *Polygon Mesh Processing*. AK Peters, 2010.

[14] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff, and C. Rössl. Geometric modeling based on polygonal meshes. In *ACM SIGGRAPH 2007 course*, New York, New York, USA, Aug. 2007. ACM Press.

[15] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics*, 14(1):213–30.

[16] S. Bougleux, A. Elmoataz, and M. Melkemi. Discrete regularization on weighted graphs for image and mesh filtering. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, pages 128–139, 2007.

[17] D. A. Bowman, J. Chen, C. A. Wingrave, J. Lucas, A. Ray, N. F. Polys, Q. Li, Y. Haciahmetoglu, J.-S. Kim, S. Kim, R. Boehringer, and T. Ni. New directions in 3D user interfaces. *The International Journal of Virtual Reality*, 5(2):3–14, 2006.

[18] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

[19] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[20] A. Chambolle. An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97, 2004.

[21] T. F. Chan and K. Chen. An optimization-based multilevel algorithm for total variation image denoising. *Multiscale Modeling and Simulation*, 5(2):615–645, 2006.

[22] S.-d. Choi, A. Lee, and S.-y. Lee. On-Line Handwritten Character Recognition with 3D Accelerometer. In *2006 IEEE International Conference on Information Acquisition*, pages 845–850. IEEE, 2006.

[23] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab an Open-Source Mesh Processing Tool. In *Sixth Eurographics Italian Chapter*, pages 129–136, 2008.

[24] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1997.

[25] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu. A finite element method for surface restoration with smooth boundary conditions. *Computer Aided Geometric Design*, 21(5):427–445, 2004.

[26] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. *Proceedings of the conference on Visualization*, pages 397–405, 2000.

[27] U. Clarenz, G. Dziuk, and M. Rumpf. On Generalized Mean Curvature Flow in Surface Processing. *Geometric Analysis and Nonlinear Partial Differential Equations*, pages 217–248, 2003.

[28] M. Dellisanti, M. Fiorentino, G. Monno, and A. E. Uva. Enhanced 3D object snap for CAD modelling on large stereo displays. *Computer Applications in Technology*, 33(1):54–62, 2008.

[29] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial Mathematics, 1996.

[30] C. M. D'Eramo, L. B. Kara, and K. Shimada. Pen-based styling design of 3D geometry using concept sketches and template models. In *ACM symposium on Solid and physical modeling*, pages 149–160, 2006.

[31] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH*, pages 317–324, New York, New York, USA, July 1999. ACM Press.

[32] J. E. Deschaud and F. Goulette. Point cloud non local denoising using local surface descriptor similarity. *IAPRS*, 38(3A):109–114, 2010.

[33] M. P. do Carmo. *Riemannian Geometry*. Birkhauser Basel, 1992.

[34] B. Dong, J. Ye, S. Osher, and I. Dinov. Level Set Based Nonlocal Surface Restoration. *Multiscale Modeling and Simulation*, 7(2):589–598, 2008.

[35] G. Dziuk. An algorithm for evolutionary surfaces. *Numerische Mathematik*, 58:603–611, 1991.

[36] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE Transactions on Image Processing*, 17(7):1047–1060, 2008.

[37] M. Elsey and S. Esedoglu. Analogue of the Total Variation Denoising model in the context of Geometry processing. *Multiscale Modeling and Simulation*, 7(4):1549–1573, 2007.

[38] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[39] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2007.

[40] K. Fujiwara. Eigenvalues of Laplacians on a Closed Riemannian Manifold and Its Nets. *Proceedings of the American Mathematical Society*, 123(8):2585–2594, 1995.

[41] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH*, pages 209–216, 1997.

[42] G. Gilboa and S. Osher. Nonlocal Linear image regularization and supervised segmentation. *Multiscale Modeling and Simulation*, 6(2):595–630, 2007.

[43] M. Gopi, P. Diaz-Gutierrez, and K. Das. Sketching free-form surfaces using network of curves. In *Sketch Based Interfaces and Modeling*, pages 127–134, 2005.

[44] R. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[45] H. Hoppe. Progressive meshes. In *23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, 1996.

[46] H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *10th IEEE Visualization Conference (VIS '99)*, VISUALIZATION '99, pages 59–66, 1999.

[47] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH*, pages 19–26, 1993.

[48] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics (TOG)*, 25(3):1126–1134, 2006.

[49] Y. Huang, R. Li, and W. Liu. Preconditioned descent algorithms for p-Laplacian. *Journal of Scientific Computing*, 32(2):343–371, 2007.

[50] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. *ACM SIGGRAPH*, pages 409–416, 2007.

[51] C. Joguet, Y. Caritu, and D. David. Pen-like, Natural Graphic Gesture Capture Disposal, Based on a Microsystem. In *Smart Object Conference*, 2003.

[52] M. Jung, X. Bresson, L. Vese, and T. F. Chan. Nonlocal Mumford-Shah Regularizers for Color Image Restoration. *IEEE Transactions on Image Processsing*, 20(6):1583–1598, 2011.

[53] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[54] L. B. Kara and K. Shimada. Sketch-Based 3D-Shape Creation for Industrial Styling Design. *IEEE Computer Graphics and Applications*, 27(1):60–71, 2007.

[55] O. A. Karpenko and J. F. Hughes. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Transactions on Graphics (TOG)*, 25(3):589–598, 2006.

[56] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[57] S.-J. Kim, C.-H. Kim, and D. Levin. Surface simplification using a discrete curvature norm. *Computers and Graphics*, 26(5):657–663, 2002.

[58] L. Kobbelt, J. Vorsatz, U. Labsik, and H. Seidel. A Shrink Wrapping Approach to Remeshing Polygonal Surfaces. *Computer Graphics Forum*, 18(3):119 – 130, 1999.

[59] M. Laforest. wiiuse - The Wiimote C Library.

[60] R. Lai. *Computational Differential Geometry and Intrinsic Surface Processing*. Ph.d. thesis, University of California Los Angeles, 2010.

[61] B. Lévy and Y. Liu. LP-Centroidal Voronoi Tessellation and its applications. *SIGGRAPH 2010*, 29(4), 2010.

[62] Y. Li and F. Santosa. A computational algorithm for minimizing total variation in image restoration. *IEEE Transactions on Image Processing*, 5(6):987–995, 1996.

[63] Lin, Hongwei, Chen, Wei, Bao, and Hujun. Adaptive patch-based mesh fitting for reverse engineering. *Comput. Aided Des.*, 39(12):1134–1142, 2007.

[64] W. Liu and N. Yan. On Quasi-Norm Interpolation Error Estimation And a Posteriori Error Estimates for p-Laplacian. *SIAM journal on numerical analysis*, 40(5):1780–1895, 2002.

[65] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of detail for 3D graphics*. Elsevier, 2002.

[66] M. Lysaker, S. Osher, and X.-C. Tai. Noise Removal Using Smoothed Normals and Surface Fitting. *IEEE Transactions on Image Processsing*, 13(10):1345–1457, 2004.

[67] T. Mcinerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[68] H. D. McKay and P. J. Besl. A method for registration of 3-D shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[69] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pages 35–57. Springer Verlag, 2003.

[70] J. Mitani, H. Suzuki, and F. Kimura. 3D Sketch: Sketch-based Model Reconstruction and Rendering. *Workshop Geometric Modeling, IFIP*, pages 85–98, 2002.

[71] H. P. Moreton. *Minimum curvature variation curves, networks, and surfaces for fair free-form shape design*. PhD thesis, 1983.

[72] H. P. Moreton and C. H. Séquin. Surface design with minimum energy networks. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 291–301, 1991.

[73] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):113–125, 1992.

[74] S. Morigi. Geometric Surface Evolution with Tangential Contribution. *Journal of Computational and Applied Mathematics*, 233(5):1277–1287, 2010.

[75] S. Morigi and M. Rucci. Adaptive Tangential Remeshing. In *In Proc. Computer Graphics International*, pages 1–5, Ottawa, Canada, 2011.

[76] S. Morigi and M. Rucci. Reconstructing surfaces from sketched 3D irregular curve networks. In T. Hammond and A. Nealen, editors, *In Proc. Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM)*, pages 39–46. ACM Press, Aug. 2011.

[77] S. Morigi and M. Rucci. Multilevel method for mesh simplification. *Submitted*, 2012.

[78] S. Morigi, M. Rucci, and F. Sgallari. Nonlocal Surface Fairing. In A. M. Bruckstein, editor, *LNCS 6667, SSVM 2011*, pages 38–49. Springer-Verlag Berlin Heidelberg, 2011.

[79] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian Mesh Optimization. In *GRAPHITE*, pages 381–389, 2006.

[80] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. FiberMesh: Designing Freefrom Surfaces with 3D Curves. *ACM Transaction on graphics (TOG)*, 26(3), 2007.

[81] M. K. Ng, S. Huanfeng, E. Y. Lam, and Z. Liangpei. A total variation regularization based super-resolution reconstruction algorithm for digital video. *EURASIP Journal on Advances in Signal Processing*, 1, 2007.

[82] A. M. Oberman. Finite difference methods for the Infinity Laplace and p-Laplace equations. page 14, July 2011.

[83] J. Oh, S.-J. Cho, W.-C. Bang, W. Chang, E. Choi, J. Yang, J. Cho, and D. Y. Kim. Inertial Sensor Based Recognition of 3-D Character Gestures with an Ensemble of Classifiers. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 112–117. IEEE, 2004.

[84] Y. Ohtake, A. Belyaeva, and I. Bogaevski. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 33(11):789–800, 2001.

[85] S. Oore, D. Terzopoulos, and G. Hinton. A Desktop Input Device and Interface for Interactive 3D Character Animation. In *Graphics Interface*, pages 133–140, 2002.

[86] G. Orbay and L. B. Kara. Sketch-based surface design using malleable curve networks. *Computers & Graphics*, 36(8):916–929, Dec. 2012.

[87] C. C. Paige and M. A. Saunders. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. *ACM Trans. Math. Softw.*, 8(1):43–71, Mar. 1982.

[88] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[89] J. Rossignac. Specification, representation and construction of nonmanifold geometric structures. *SIGGRAPH Course Note*, 1996.

[90] J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In *Geometric Modeling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

[91] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, Nov. 1992.

[92] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[93] S. Schaefer, J. Warren, and D. Zorin. Lofting curve networks using subdivision surfaces. *Symposium on Geometry Processing (SGP)*, pages 103–114, 2004.

[94] R. Schneider and L. Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer aided geometric design*, 18(4):359–379, 2001.

[95] K. Shimada, A. Yamada, and T. Itoh. Anisotropic Triangulation of Parametric Surfaces via Close Packing of Ellipsoids. *International Journal of Computational Geometry and Applications*, 10(4):400–424, 2000.

[96] O. Sorkine. Differential Representations for Mesh Processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

[97] V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 20–30, 2003.

[98] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. *ACM Transactions on Graphics (TOG)*, 22(4):1012–1033, 2003.

[99] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH*, pages 351–358, New York, New York, USA, 1995. ACM Press.

[100] D. Terzopoulos. Deformable Models: Classic, Topology-Adaptive and Generalized Formulations. In *Geometric Level Set Methods in Imaging, Vision, and Graphics*, pages 21–40. Springer-Verlag, New York, 2003.

[101] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[102] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models:Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, Aug. 1988.

[103] W. Tiller and L. Piegl. *The NURBS book*. Springer Verlag, 1997.

[104] G. Welch and G. Bishop. An introduction to the Kalman filter. In *SIGGRAPH Course Note*, 2001.

[105] W. Welch and A. Witkin. Variational Surface Modeling. *SIGGRAPH*, 26(2):157–166, 1992.

[106] G. Wesche and H.-P. Seidel. Freedrawer - A free-form sketching system on the responsive workbench. In *ACM symposium on Virtual reality software and technology*, pages 167–174, 2001.

[107] G. Xu. Convergent Discrete Laplace-Beltrami Operators over Triangular Surfaces. *Geometric Modeling and Processing*, pages 195–204, 2004.

[108] G. Xu, Q. Pan, and C. L. Bajaj. Discrete surface modelling using partial differential equations. *Computer Aided Geometric Design*, 23(2):125–145, 2006.

[109] D.-M. Yan, B. Levy, Y. Liu, F. Sun, and W. Wang. Isotropic Remeshing with fast and exact computation of restricted Voronoi Diagram. *Eurographics Symposium on Geometry Processing (SGP)*, 28(5):1445–1454, 2009.

[110] S. Yoshizawa. Fair Triangle Mesh Generation with Discrete Elastica. *Geometric Modeling and Processing*, pages 119–123, 2002.

[111] S. Yoshizawa, H.-P. H.-P. Seidel, and A. Belyaev. Smoothing by Example: Mesh Denoising by Averaging with Similarity-based Weights. In *IEEE International Conference on Shape Modeling and Applications (SMI)*, pages 38–44. Ieee, 2006.

[112] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An Interface for Sketching 3D Scenes. *ACM SIGGRAPH*, pages 163–170, 2006.

[113] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[114] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. *ICML Workshop on Statistical Relational Learning*, 2004.

[115] D. Zhou and B. Scholkopf. Regularization on discrete spaces. In *DAGM conference on Pattern Recognition*, pages 361–368, 2005.