ΑΝΑΚΑΤΑΣΚΕΥΗ 3Δ ΜΟΝΤΕΛΩΝ ΣΧΕΔΙΑΣΗΣ ΜΕ ΥΠΟΛΟΓΙΣΤΗ ΒΑΣΙΣΜΕΝΗ ΣΕ ΓΕΩΜΕΤΡΙΚΑ ΠΡΟΣΔΙΟΡΙΣΜΕΝΕΣ ΤΟΜΕΣ

Η
ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Αντώνιο Πρωτοψάλτη

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ

Νοέμβριος 2009

DEDICATION

To my three little angels Gianni, Mika and Eugenia

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Professor Ioannis Fudos for his valuable guidance and advice through my research, his endless patience with me, and his courageous support when it was most needed.

I would like to thank my colleagues in the graphics group for their support and the useful feedback through the entire period of my research.

I would like to especially thank my wife Dimitra, my three children, and my parents for all the love, care, and support they have provided me unconditionally over the years. I credit all my successes to them.

TABLE OF CONTENTS

LIST OF FIGURES

ΕΚΤΕΝΗΣ ΠΕΡΙΛΗΨΗ ΣΤΑ ΕΛΛΗΝΙΚΑ

Αντώνιος Πρωτοψάλτης του Ιωάννη και της Χρυσούλας. PhD, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Νοέμβριος, 2009.
Ανακατασκευή 3Δ μοντέλων σχεδίασης με υπολογιστή βασισμένη σε γεωμετρικά προσδιορισμένες τομές.
Επιβλέποντας: Ιωάννης Φούντος.

Η ανάστροφη μηχανική (reverse engineering) είναι μια διαδικασία μέσω της οποίας ανακατασκευάζουμε μια εύκολα τροποποιήσιμη αναπαράσταση ενός αντικείμενου του οποίου την επιφάνεια έχουμε πάρει με τη μορφή νέφους σημείων. Στην εργασία αυτή μελετάμε την χρήση τομών (cross sections) που στην ανάστροφη μηχανική είναι μια ειδική περίπτωση χαρακτηριστικών (features). Τα μοντέλα αναπαράστασης στερεών τα οποία βασίζοντα σε χαρακτηριστικά και περιορισμούς είναι από τη φύση τους κατάλληλα για την χρήση σε συστήματα σχεδίασης με υπολογιστή και παρέχουν τη δυνατότητα εύκολης τροποποίησης και μπορούν να μοντελοποιήσουν την πρόθεση του χρήστη-σχεδιαστή (design intent).

Το νέφος σημείων αρχικά τεμαχίζεται σε έναν αριθμό από δισδιάστατες τομές οι οποίες περιέχουν ένα σύνολο από σημεία στο επίπεδο. Κατόπιν, επεξεργαζόμαστε το κάθε τέτοιο 2Δ σύνολο λεπταίνοντας το πάχος των συσσωρεύσεων του νέφους σημείων ώστε να περιγράφει μια 2Δ καμπύλη. Αυτό επιτυγχάνεται χρησιμοποιώντας πεδία δυνάμεων προσαρμοσμένα για να έχουν βέλτιστη απόδοση σε νέφη σημείων που περιγράφουν περιβλήματα. Η κάθε τέτοια ακολουθία διατεταγμένων σημείων κατόπιν χωρίζεται αυτόματα σε έναν αριθμό από τμήματα ώστε το κάθε ένα να μπορεί να περιγραφεί ικανοποιητικά από μια ρητή καμπύλη Bezier 2^{ου} βαθμού. Με τον τρόπο αυτό, εξαλείφεται ο θόρυβος και η τομή αυτή του περιβλήματος μπορεί να προσφέρει πολλές πληροφορίες στα μετέπειτα στάδια της ανάθεσης γεωμετρικών περιορισμών και της 3Δ ανακατασκευής.

Κατόπιν εισάγονται αυτόματα αλλά και διαδραστικά γεωμετρικοί περιορισμοί τόσο εντός της ίδιας τομής όσο και μεταξύ διαφορετικών τομών. Λύνοντας το προκύπτον σύστημα γεωμετρικών προσδιορισμών μπορούμε να διαμορφώσουμε τόσο την μορφολογία της κάθε τομής όσο και να αλλάξουμε την σχετική τοποθέτηση των τομών μεταξύ τους.

Οι προκύπτουσες τροποποιημένες τομές ανακατασκευάζονται με μια νέα μέθοδο ανακατασκευής τομών που μπορεί να λειτουργεί ακόμη και αν δύο γειτονικές τομές διαφέρουν πάρα πολύ. Αυτό επιτυγχάνεται χρησιμοποιώντας έναν σκελετό ορίων που ανταποκρίνεται στις δύο τομές, ο οποίος οδηγεί την αυτόματη κατασκευή ενδιάμεσων τομών.

Τέλος προσφέρουμε ποσοτικά και ποιοτικά αποτελέσματα σχετικά με την απόδοση και την ευχρηστία της μεθόδου που παρουσιάσαμε.

ABSTRACT

Protopsaltis, Antonios, A.P. PhD, Computer Science Department, University of Ioannina, Greece. January, 2010.
Reconstructing 3D CAD Models based on geometrically constrained cross sections
Thesis Supervisor: Fudos, Ioannis.

We introduce a novel approach to reconstructing 3D objects from cross sections of point clouds acquired by laser scanning. Cross sections are almost planar clusters of 3D points. We first thin each cluster to obtain an ordered one dimensional set of points. We then partition the point set to subsets that can be approximated adequately by piecewise quadratic or cubic rational Bezier curves using an optimal fitting method. For each curve we select a number of representative points that lie on the fitting curves which are then used for reconstructing the object surface. Inter-cross section and intra-cross section constraints are imposed to support parameterization and editing of the derived model. Shape and topological differences between adjacent object contours pose several issues for the 3D reconstruction process. By using the contour skeleton information we produce intermediate cross sections representing places where ramifications occur to achieve robust covering (meshing) of adjacent slices. Finally we describe a proof of concept implementation of our method and several examples that demonstrate its effectiveness and efficiency.

CHAPTER 1. INTRODUCTION

The creation of an appropriate computer representation of existing objects from vast sets of scanned data points has been an important necessity in many areas of engineering, medical sciences and arts. The process of capturing the geometry of existing physical objects and then using the data obtained as a basis for creating a new design is called Reverse Engineering of solids. Due to recent advances in laser scanning, the process of deriving accurate and topologically consistent models that are ready to use in CAD/CAM systems has become a realistic expectation in the geometric modeling community.

While conventional engineering transforms engineering concepts and models into real parts, in reverse engineering actual parts are transformed into computer models suitable for reproducing or redesigning these parts. In conventional computer-aided design the computer representation of objects is performed by means of operations typically defined interactively using advanced geometric and graphics primitives. The resulting representation is then used for further design, and finally for numerically controlled manufacturing, layered manufacturing or other manufacturing techniques. In reverse engineering, engineering concepts are derived from actual parts when no drawings or documentation are available. The required degree of accuracy may only be obtained if the geometric modeling technique employed precisely represents the shapes being analyzed.

The process of reverse engineering is usually decomposed into the following steps: data acquisition, point cloud segmentation, surface fitting, and model creation (figure 1.1).

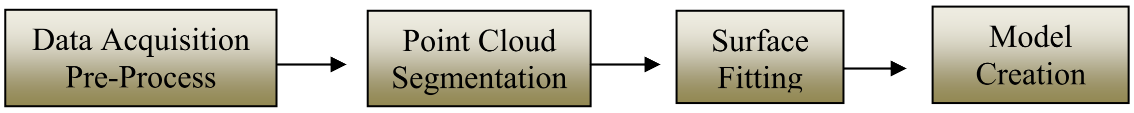


Figure 1.1 Phases of Reverse Engineering

Data acquisition is accomplished by means of 3D laser scanners or other less accurate techniques such as 3D reconstruction from 2D snapshots using correspondence and epipolar geometry. The data acquired is in the form of an unorganized 3D point cloud where each point corresponds to a point on the surface of the object. The measured data is pre-processed before further operations are performed. In many cases where the object is large or very complicated one point cloud is not enough to describe the entire object. In such cases we obtain multiple point clouds each one covering a different part of the object. These point clouds are either merged in one master point cloud or are considered as segments from the beginning.

The object may be anything from a combination of smaller objects to an open surface. Segmentation partitions the point cloud into disjoint subsets each represented by a boundary representation that consists of surfaces. Each derived subset may be classified for its surface types (planar, spherical, conical, etc) [7] or approximated in the fitting step with free-form surfaces.

The fitting surface step fits an appropriate surface to the point set. This is an open research field in CAGD (see e.g. [8]).

Finally, stitching together these surfaces (with appropriate continuity) creates a Boundary Representation that could be used in subsequent phases of CAD/CAM.

Traditionally, the result of this process is a Brep model [42] of the real object that is adequate to describe positional information and therefore it is suitable for reproduction but cannot capture any of the higher-level structure of the object or the designer's intent. Therefore, it is not suitable for redesign. Modification of a part is often a tedious task that requires experienced users and state of the art software and hardware. For instance, a Brep representation might be able to approximate the shape of a cylindrical hole, but the fact that the hole is actually cylindrical is not captured. As a result, it is difficult for a designer to perform a simple modification such as altering the diameter of the hole. Also, the initial model suffers from inaccuracies caused by sensing errors inherited from the data acquisition phase, approximation and numerical errors arising from successive transformations or other geometric manipulations, or possible wear of the actual part. All these errors introduce distortion and may act accumulatively. Redesign may be accomplished through geometric regularities and constraints that have been derived from the original cloud point.

We present a novel computer aided reengineering paradigm based on careful slicing of the 3D point cloud and advanced post processing of the resulting cross sections.
Post processing aims to eliminate noise and partition the point set to point sequences that correspond to low degree curve segments. The curve segments are then approximated using quadratic rational Bezier curves. We then subdivide the curve segments in equal length chord segments and use the corresponding points to perform 3D mesh reconstruction. The final model should be editable which is succeeded by the incorporation of geometric constraints and feature recognition.
Figure 1.2 illustrates the overall process:

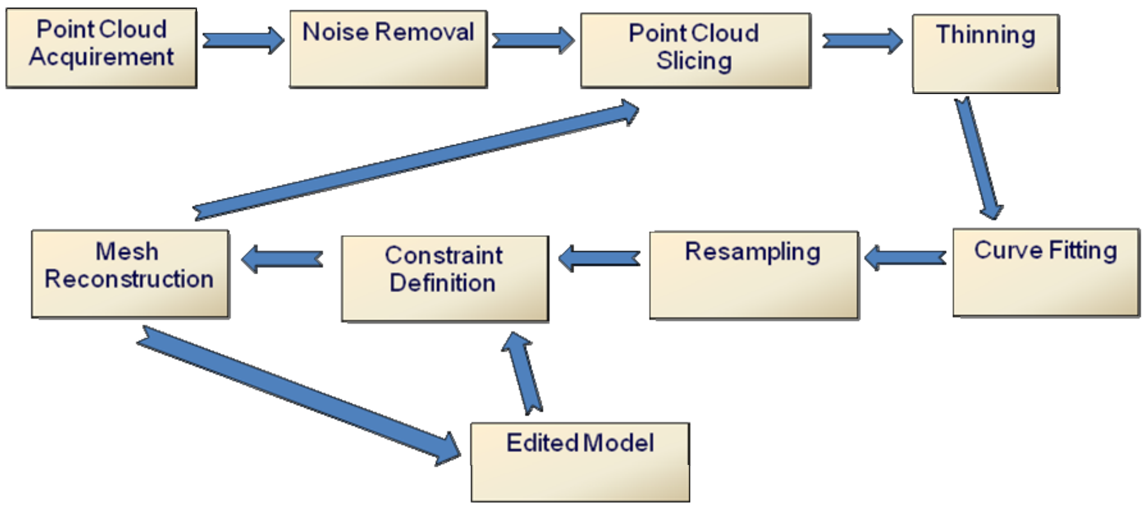


Figure 1.2 Our Reverse Engineering Framework

The rest of this thesis is structured as follows: Chapter 2 provides a review of reverse engineering approaches in computer aided design. In Chapter 3 we present our approach to extracting and processing cross sections from a 3D point cloud. A fast and efficient curve approximation method for fitting rational quadratic Bezier curves to 2D points is presented in Chapter 4. Chapter 5 focuses on incorporating local and global constraints to our model. In Chapter 6 we present our object reconstruction method, based on constrained cross sectional contours, and the model editing methodology. Implementation issues and experimental evaluation of our object reconstruction framework are presented in Chapter 7. Chapter 8 provides conclusions.

CHAPTER 2. REVERSE ENGINEERING IN COMPUTER AIDED DESIGN

2.1. Introduction

Reverse Engineering is a complex process that is central to industry, arts, archaeology and architecture. The creation of CAD models appropriate for computer-aided manufacturing is an expensive and demanding task. In this thesis we focus on re-engineering solid objects for which we have acquired the point cloud of their boundary surface. Subsequently, we wish to obtain a 3D CAD model which is editable and manufacturable. Most related previous approaches have dealt with this problem considering only mechanical parts and employing feature-based knowledge to detect and represent holes, chamfers, extrusions or protrusions. It is important to provide means for editing 3D objects that respect all types of object morphology and topology.

There is a variety of geometric representations that can be used at different levels of CAD applications. The suitable representation scheme for each application depends on the scope of the application and its peculiarities. Some modeling types are simple and aim at providing only an external representation of the object, whereas others aim at encapsulating and providing additional knowledge and data, such as design intent, functionality, and editability.

In this chapter we study common modeling schemes used in CAD applications. An object can be represented in the simple form of raw data, such as a point cloud corresponding to points on the surface of the object. A widespread scheme in solid modeling is the Boundary Representation (B-rep) model where the facets and edges that describe the boundary of a solid are modeled using a connectivity graph and a collection of surface and edge patches. This type of model is not always suitable for redesign because of the lack of expected regularities and constraints. This information is not present because each facet is determined independently. On the other hand, Constructive Solid Geometry (CSG) and volume models handle objects as 3D solids. There are also higher-level representation schemes that capture not only the shape of the object but also provide information pertaining to design intent and functionality, which can be used later on for re-parameterization and modification. We briefly describe each scheme and evaluate its suitability for various CAD applications.

2.2. Related Work

Various authors have considered creating reverse engineered 3D models. Sensor based reverse engineering makes possible the creation of CAD models appropriate for computer-aided manufacturing directly from existing physical prototypes or similar objects for which usable CAD models don't exist. Some researchers have dealt with the tedious task of making their model editable. This is often accomplished by incorporating local and global geometric constraints in the CAD model. In plain solid reconstruction, a geometric model is captured directly from the geometry of the point cloud acquired by 3D laser scanning. This method is commonly used in modeling sculptures in arts. These techniques are quite accurate but do not support large scale modifications, additions or other high level operations to the extracted model.

Ko et al. [38] discuss a method that uses a set of points to model a human face. The discussion focuses on the reorganization of the points, facet modeling and tool path generation. Ma and He [50] present an approach to shape a single B-spline surface by a cloud of points. The discussion concentrates on the parameterization of these unorganized points.

Varady et al [81] compute a “feature skeleton” on the mesh that determines the primary regions of the object. The final surface structure comprises the optimally located boundaries of the connecting features and setback type vertex blends, which are faithfully aligned with the actual geometry of the object. This CAD-like surface structure is sufficient for high-quality surface approximations. Stamati [71] is using an advanced surface analysis technique to extract the morphology of the reconstructed point cloud. This technique is very powerful and accurate but is not suited for rapid reverse engineering since it requires an extensive analysis process.

A feature-based reverse engineering method was also used by Au et. al [2] for reverse engineering a mannequin for garment design. Generic models of mannequin torsos are fit to 3D point clouds of human torsos for garment modelling applications. The basic concept in this method is to create a generic mannequin model of a human torso, which is appropriately aligned with the 3D point cloud of the desired human torso model, and the generic model is fit to the point cloud by matching up characteristic points of the models e.g. peaks. This method creates parameterized models by exploiting the features of the object and by using them to constrain the fitting process. It is an automated approach to reverse engineering human torsos that creates parameterized models with good accuracy.

Researchers such as [77],[78] have focused on creating high accuracy models of manufactured mechanical parts. The REFAB project uses a feature-based and constraint-based method to reverse engineer mechanical parts. REFAB is a human interactive system where after the 3D point cloud is presented to the user, the user selects a feature from a predefined list of features, and specifies the approximate location of the feature in the point cloud. The system then fits the specified feature to the actual point cloud data using a least square means method iteratively. The authors give emphasis on the fitting of pockets, where the user draws a profile of the pocket on the point cloud and the system then fits the profile to the data and the profile is then extruded to create the pocket. This feature-fitting process is made more accurate by using constraints that are detected by the system, verified by the user and then exploited to achieve a better fitting of the features according to the data. The system supports constraints such as parallelism, concentricity, perpendicularity and symmetry. The constraints defined and used in REFAB seek to reduce the degrees of freedom associated with the object as much as possible, so as to achieve high precision models in less time.

Chen and Hoffman [14], define semantics for the creation of generated features. This work is based on a neutral, high-level design representation, called Erep (editable representation), which allows design modifications based on a general design paradigm. This framework considers generated features based on a planar profile and then revolved, swept and extruded in 3D shape.

Dobson et al [23] discuss the fitting of a non-uniform rational B-spline curve to a set of co-planar points. The fitting process uses characteristic points and is demonstrated by fitting a facial 2D profile.

Langbein et al [40] [41] analyze the type of symmetries and shape regularities that may be observed in a Brep model and efficiently apply them in a reverse engineering process to create accurate and aesthetically robust models. The process of model improvement, called beautification, modifies surface parameters to produce a model that is more suitable for redesign.

Sato et al [63] propose a laser projection system and an image processor which are used for determining a fixed set of horizontal cross sections of the recognized object which is placed on a turntable in a stable vertical orientation. For each horizontal cross section they compute the Fourier shape descriptors of the boundary. Constraints between two cross sections may be defined such as horizontal strain, section shape, torsion, and displacement.

Werghi et al [85], suggest a general incremental framework whereby constraints may be added and integrated in the model reconstruction process.

Hoppe et al. [33] propose a method for surface fitting based on polygonal meshes. They produce a surface that approximates the original object surface based considering data points in close.

2.3. Raw data

The most basic and simple way to represent a 3D object is as raw data. By raw data we mean an unstructured collection of geometric primitives such as a point cloud or a range image. Such data are usually produced directly from a 3D object scanning or

3D reconstruction setting. The density of the data sets produced by these methods depends on the sampling rate used to acquire information from the object's surface. Also, very often the point clouds obtained contain noisy data due to physical characteristics of the object or limitations and regulations of the acquisition method used. However, this problem has been dealt with and processing methods have been suggested that overcome this problem. The characteristic of this representation model is that it describes the object as discrete data, i.e. points, without providing any information about the connectivity, the topological relation among geometric primitives or the design intent. This type of representation is mainly used in point-based modeling, i.e. [19][39] and reverse engineering applications [33].

2.4. Boundary Representation (Brep)

The appearance of an object depends largely on the exterior of the object. Boundary representations (Brep) [42] models are commonly used in computer graphics and CAD applications. This type of model consists of a collection of connected surface elements: facets, edges and vertices. A facet is a bounded portion of a surface, an edge is a bounded piece of a curve and a vertex lies at a point. Other elements are the shell (a set of connected facets), the loop (a circuit of edges bounding a facet). Surfaces can capture objects of complex and freeform design. Thanks to advancements in computer graphics hardware we are able to handle efficiently the CPU-intensive processing required by Brep. These factors have resulted in the increased usage of this representation in a wide spectrum of applications. A Brep model is often realized as a mesh of triangular or quadrilateral (and in general polygonal) planar or higher degree surface facets.

Planar polygonal meshes (called polyhedral representations) are mostly suited for rendering and virtual reality and not for CAD applications since they do not provide sufficient detail. Often, other representation schemes are converted to polygonal representations for the purpose of rendering. Polyhedral representations such as triangulations are also used in reverse engineering applications, usually as intermediate representations during the re-engineering process. A drawback of representing a 3D object with a polygonal mesh is that it cannot capture design semantics, such as design intent, inter part relations and overall behavior. Also model editing is only feasible in a local corrective sense. Smooth object surfaces cannot efficiently and accurately be represented by a polygonal mesh, even when a large number of polygons are used, since the polyhedral representation by definition cannot accommodate for G^1 continuity. For example, to render areas of high curvature quite accurately we need to increase the number of polygons and decrease significantly the facet size.

Overall, polyhedral representation is not suitable for describing objects with specific design characteristics and functionality, such as mechanical and industrial parts. Also it is not appropriate for describing complex and detailed objects since the large number of polygons needed to sufficiently approximate the initial object makes the method unaffordable both time-wise and space-wise.

Applications such as aesthetic and industrial engineering, reverse engineering and jewellery design use commonly non-planar surfaces to capture the boundaries of complex objects [7]. A Brep model may be constructed using NURBS (Non-Uniform Rational B-Splines) or other parametric surface patches. This type of representation is useful in applications where free-form surfaces are part of the repertoire of primitive geometric entities. Brep can capture almost any type of object, such as mechanical parts and objects of aesthetic design. Surfaces can be described using appropriate parametric representations. Brep models make editing of local features feasible by interactively placing control points, therefore modifying the shape or curvature of the object's feature. However, Brep models on their own do not capture higher design characteristics of the object such as functionality and part relationships. The information provided through this type of model is limited and does not provide tools for modifying parts of the model that affect the whole design. Therefore, Brep models are used in combination with other techniques (e.g. features, constraints) to obtain higher-level descriptions that correspond to more flexible and useful models that are suitable for CAD applications. For instance, in [40], the authors present a beautification process based on constraints which is performed on B-rep models constructed from reverse engineering range data. B-rep models acquired by re-engineering can present various inaccuracies and errors, therefore the authors suggest the beautification of the models by describing topological regularities in terms of geometric constraints.

2.5. Volume Modeling

While surface raw data and Brep modeling schemes provide data concerning the boundary of a model, constructive solid geometry (CSG) [42] and volumetric models represent the objects as a volume. This type of representation can be used for objects that B-rep cannot sufficiently describe. For example, a Brep model cannot represent unambiguously a sphere containing a hollow, whereas a volume model can easily capture such solids.

Constructive solid geometry (CSG) models are created by performing Boolean operations on solid primitives e.g. spheres, cones, cylinders and cubes. We perceive that CSG models represent objects that can be created from solid primitives. CSG may model higher degree free-form objects using a small number of special free-form primitives. In general, the CSG representation scheme is well suited for mechanical part design and for all applications where the design history can be expressed as a tree of Boolean operations on geometric primitives. Also editing and local shape modification is performed by intervening in the appropriate operation (internal tree node). Converting CSG models to render-able ones is extremely difficult and therefore CSG is commonly used in conjunction to Brep. In this case a Brep model is always maintained and every modification is transformed to an incremental Brep editing operation. Constraints may also be used in conjunction to CSG for performing multiple internal node modifications at a single step.

Volume pixels (voxels) are used in a volumetric approach to 3D object representation. A voxel is a geometric primitive and represents the smallest discrete volume used in this representation scheme. Voxel-based representations are commonly used for visualizing unstructured 3D volume date such as data from scientific computing, medical imaging etc. Although used in early CAD/CAM settings, volumetric representations have been proven to be very inefficient for computer aided editing, rendering and manufacturing. This representation scheme may be used as redundant auxiliary information in CAD applications [35] such as solid modeling, reverse engineering and feature-based and constraint-based modeling for the purposes of physical modeling and simulation.

2.6. Higher-level Representations for CAD

A current promising trend in computer-aided design is to use higher-level structures for model representation. These structures are based on one of the former representation types in combination with additional structural, topological or other information. A feature-based representation scheme describes the object as a combination of features, which are surfaces or solid parts with specific characteristics. A constraint-based representation scheme uses geometric constraints enforced on the model and its features to obtain a more accurate representation that captures designer requirements. The skeleton of a model can also be considered as a higher-level CAD representation that can be used for specific operations such as feature detection and extraction.

More specifically, the feature-based model is a representation scheme that is growing more and more popular. The model is described by defining collections of feature elements and relationships among them. The features are collections of points, surfaces or other features. For example a commonly used feature type is a cross-section of a solid. Constraints are applied to the features to create more accurate and robust models, but also for enforcing global criteria such as tolerance and beautification. This type of model representation has been established initially for manufacturing mechanical parts, where a library of features is created and then relationships among feature elements are enforced. The feature-based scheme is well suited to industrial design in general since it provides for advanced editability. This is due to the knowledge encapsulated by the model concerning tolerances, constraints, relationships and connectivity. For this reason, feature-based methods are often characterized as knowledge-based. Their main objective is to exploit any knowledge and information pertaining to design intent, functionality and construction process. Besides, this representation scheme supports collaborative CAD, reverse engineering and VLSI applications. This type of model also provides the user-designer with the capability of editing, redesigning and reconstructing the original design, depending on her preferences and needs by tailoring the model features [32].

A powerful higher-level structure for representing objects is the constraint-based scheme, which is often used in combination with features [8]. This representation scheme is particularly preferred in CAD applications where the objects being modeled, modified and manufactured are of geometric or freeform design and must conform to constraints determined locally on specific components or globally on the whole model [1]. Constraints defined on a model or its individual components can refer to almost any characteristic, i.e. geometric attributes, such as size and shape, topological characteristics, such as placement and connectivity, functionality and behavior. Constraint-based models are widely used in architecture, mechanical engineering, electronic design, aesthetic and industrial design, for design, modeling or re-engineering. The types of constraints defined depend on the nature of the CAD application. For example, in VLSI CAD a geometric constraint scheme may be used in conjunction to feature-based or other graph-based connectivity modeling. Constraints are imposed on each design feature used in the VLSI circuit referring to the feature's intra-connectivity and its local characteristics (i.e. area, size, geometry). Constraints may also be imposed to express inter-feature connectivity requirements. Finally, constraints are also enforced globally on the circuit, and are targeted to optimize the overall placement and routing of the features on the chip.

An object can also be represented by its skeleton. By skeleton we mean the closure of all points that have more than one closest point on the shape boundary (for example the medial axis transform). This representation provides the topology and shapes that exist in the object and also reflects the symmetries of an object. Depending on the type of application the skeleton is used for, it may be a 2D or 3D representation. For instance, in 3D the medial axis transform produces a medial surface. The exact computation of the 3D skeleton is a computationally intensive problem that returns a skeleton as complex as the object itself. Therefore we usually seek for an approximation. A skeleton representation scheme is used in various CAD applications for object recognition and retrieval [18], animation [9] and other solid modeling operations ([66], [72]). It is widely used in feature-based modeling, where it can be employed to describe the shape of features, in feature detection and extraction applications and shape deformation, for instance refer to [47] and [86].

2.7. Feature-Based / Constraint-Based Models

A product model can be built by using (design) features; this is known as design by features or feature based modeling. One can start either with a more or less complete geometric model and define form features on it, or one starts from scratch by combining form features from a standard library. Design with pre-defined form features can reduce the number of input commands substantially. This is especially advantageous in re-design. The parametric representation of features provides a powerful way to change features with respect to their dimensions.

Features can serve as functional elements to designers. They may be defined interactively. Most often, this is carried out by identifying the faces belonging to a certain feature on the product model that is under consideration. Generic features may be used over and over again in CAD. This type of model representation is very convenient for mechanical engineering and manufacturing, where there is a need for connectivity and continuity between the different elements of the model. Most machined parts are made using a relatively small number of manufacturing operations. Reverse engineering of such parts can be done using a form of parametric fitting where the primitives correspond to these features. Also, feature-based models are ideal for industrial design and manufacturing because the model can be easily modified. This is due to the knowledge provided by the model concerning the tolerances, the constraints, the relationships and connectivity between the features.

Feature-based and constraint-based methods are often characterized also as knowledge-based. Their main objective is to exploit any knowledge and information that is connected to the design intent, functionality and construction process of the object being reverse engineered. Consequently, it is useful to exploit the design intent and feature relationships that exist in models created for industrial use because they justify some of the attributes of the object that might look like they make no sense. These elements are exploited through the usage of geometric constraints.

The main focus in all of the above works is to exploit any knowledge that is available about the initial object and the parameters, features and constraints that it contains. By using this information we can more efficiently create and manipulate part characteristics so as modify and create more advanced models.

CHAPTER 3. CROSS SECTION EXTRACTION AND PROCESSING

3.1. Introduction

The reconstruction of an object from a set of cross-sections has intrigued computer science researchers for the last decades. The need for such reconstructions is a result of the advances in medical imaging technology. Technologies such as magnetic resonance imaging (MRI), computed tomography (CT), ultrasound imaging or other systematic scanning devices, allow measurements of internal properties of objects to be obtained in a nondestructive fashion such that contours representing the boundaries of the objects may be extracted on slices, and then interpolated in order to reconstruct and visualize the analyzed objects. These measurements are usually obtained one slice at a time, where each slice is a 2D array of scalar values corresponding to measurements distributed over a plane passing through the object. The set of planes generating the slices are usually parallel to each other and equispaced along some axis through the object. 3D reconstructions of organs are widely considered to be an important diagnostic aid in the medical world.

3.2. Related Work

In recent years, some novel cloud data modeling approaches take into account a direct manufacturing of cloud data without involving surface reconstruction for more efficient rapid product development.

[67] et al directly slice a point set utilizing implicit quadric surfel, so as to obtain contour curves for Rapid Prototyping (RP).

Most of RP technologies utilize layered manufacturing, which is to make very thin layers and accumulate them [88]. One such approach is to directly slice the point cloud along the part building direction and generate a layer-based model for the use of rapid prototyping technique. Liu [49] developed an automated segmentation method for generating layer-based models from cloud data. The developed algorithm is efficient in terms of computation. However, the main drawback is the difficulty to control the shape error of the final generated model, in comparison with the original cloud data. [89] et al present an adaptive point cloud data slicing method creating a layer-based RP model ready to be fed to RP machines for fabrication. Much emphasis is given on how to control the layer thickness so that a user-specified shape error is met.

Dedieu et al [21] presented an algorithm for ordering unorganized points assuming all points are on the reconstructed curve. Taubin et al [74] reconstructed a planar curve from unorganized data points using an implicit simplicial curve defined by a planar triangular mesh and the values at the vertices of the mesh.

Levin [44] used a method called Moving least squares to thin a point cloud. This method computes a simple regression curve/surface C_i for each data point P_i which locally fits a certain neighborhood of P_i using a weighted regression scheme.

In this work, we present an intuitive method of point cloud segmentation by using the shape error to control the layer thickness so that each layer will yield the same shape error.

3.3. Slice Extraction

Our reconstruction process starts by slicing the point cloud data into a number of cross sections along a user-specified slicing direction. A single slicing direction may not be sufficient for complex objects. For such cases the original point cloud is decomposed into meaningful components using advanced segmentation techniques. Shape decomposition has been studied for decades and there is a large amount of previous work [31][62]. Approximate convex decomposition [46] method decomposes a given component by ‘cutting’ its most concave features based on the convex hull of the input model and a user defined concavity tolerance τ.

Slice thickness is controlled by a user defined thickness threshold value that specifies the maximum allowable width of a projected point set. The thickness threshold value is adapted iteratively until it falls under the user specified levels. Since slice thickness is in general greater than zero, virtually no point is exactly located on a given slicing plane. For this reason, the cloud points in each slice are projected onto a plane perpendicular to the slicing direction cutting the slice in half. Slice selection may be controlled by a user defined parameter called slicing distance. Slicing distance specifies the fixed distance between two adjacent slices. There may be cases where the slicing distance is too large for a certain object. As a consequence, the exact geometry of the object is not recorded accurately. The slicing distance parameter should be set according to the object particular features.

In many cases we obtain adjacent slices that are very similar. This might happen when the sliced object feature is symmetric such as a cylinder or parallelepiped part. Many of these slices may be eliminated from the entire process of reconstruction. If three adjacent slices are of similar shape, then the in between slice is eliminated. Similarity of slices may be detected using principal component analysis and skeleton extraction so as to achieve rotational and translational invariance.

3.4. Preprocessing and Thinning

Depending on the data acquisition and the slicing process a cross-section may contain points that form a shape with thick border. Thinning is the process that identifies the specific points from the data set that are essential to form the actual 2D shape of the cross-section. We call the outcome of this thinning process a thin data set.

The Medial Axis, is a well defined process for extracting a skeleton, but does not always produce a skeleton for the purposes of thinning due to the complexity of the result. Most thinning algorithms work iteratively. The edge pixels are examined against a set of criteria to decide whether they are essential skeleton pixels or not. A common disadvantage of many thinning algorithms is the deformation that is induced on the shape of the skeleton at regions where corners or boundary crossings are formed. Single pixel irregularities may yield unintuitive changes in an otherwise simple skeleton. Furthermore, the extraction of the skeleton does not often preserve the connectivity of the shape. Necking, tailing and spurious projection (line fuzz) are some common flows of many thinning methods [57].

The Force Based thinning algorithm [58] is based on the idea that the boundary should be used to locate the skeletal pixels by exerting a force towards the inner pixels. In that way, the skeleton of the shape lies at pixels where the forces imposed have opposite directions (Figure 3.1).

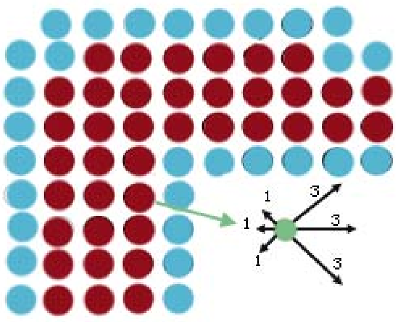


Figure 3.1 Force Based Thinning Strategy

Thinning algorithms need as input a 2D array of points. To convert our unordered set of points to a 2D array we will use an anti-aliasing algorithm. Supersampling [84] (or post-filtering) is the most common form of anti-aliasing. It involves calculating a virtual image at a spatial resolution higher than the pixel resolution and averaging down the high resolution image to a lower pixel resolution. Main advantage of this method is the trivial implementation.

We will define a virtual grid (Figure 3.2) of size G_x *x* G_y. G_x and G_y are the *x* and *y* grid resolutions which depend on the data set density. Each grid cell *G(i,j)* corresponds to a certain area in the cross section specified by the top left and the bottom right positions as follows:

$$(x_{\min} + i\cdot\frac{x_{\max}-x_{\min}}{G_x}, y_{\min} + j\cdot\frac{y_{\max}-y_{\min}}{G_y}) \quad (x_{\min} + (i+1)\cdot\frac{x_{\max}-x_{\min}}{G_x}, y_{\min} + (j+1)\cdot\frac{y_{\max}-y_{\min}}{G_y})$$

x_{min}, x_{max}, y_{min}, and y_{max} are the minimum and maximum coordinates in the original point set. Subsequently, for each point $P(p_x, p_y)$ we increase the intensity of the corresponding grid cell given by

$$G(\left\lfloor \frac{p_x \cdot G_x}{x_{\max} - x_{\min}} \right\rfloor, \left\lfloor \frac{p_y \cdot G_y}{y_{\max} - y_{\min}} \right\rfloor)$$

Each grid cell will play the role of a pixel that is either *on* or *off*. We define a grid cell to be "*on*" if its total intensity is greater than the mean intensity of all cells in the grid. Figure 3.2 shows an example grid and the pixels that are "*on*" and "*off*".

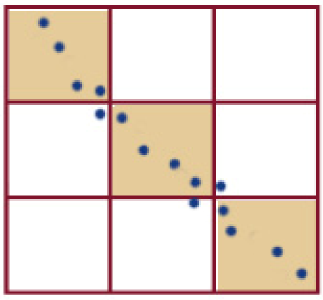
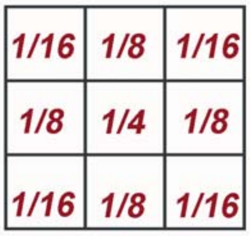


Figure 3.2 A virtual grid

Depending on grid resolution, the above process may produce a grid with a number of non connected pixels. To minimize the chance of occurrence of non connected pixels the anti-aliasing will be performed with the use of weights. The intensity of a cell $AW_{i,j}$ will depend on the weighted sum of its own and its neighbor's intensity W (Figure 3.3). Note that the sum of all weights is 1. The following equation computes the anti-aliased intensity of a cell taking into account its eight neighbors:

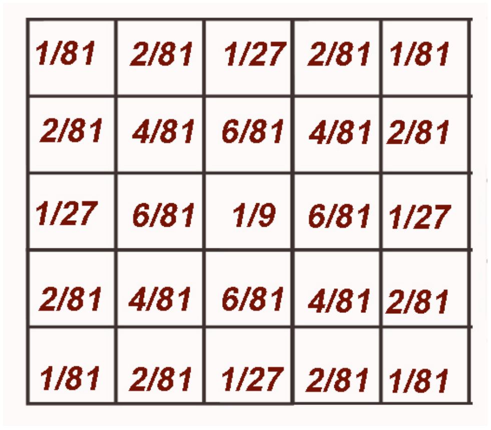
$$AW_{i,j} = \sum_{m=i-1}^{i+1} \sum_{r=j-1}^{j+1} \frac{W_{m,r}}{F_{m,r}}, \qquad \Phi_{m,r} = \begin{cases} 4 & m \neq i \quad \wedge \quad r \neq j \\ 2 & m = i \quad \oplus \quad r = j \\ 1 & m = i \quad \wedge \quad r = j \end{cases}$$



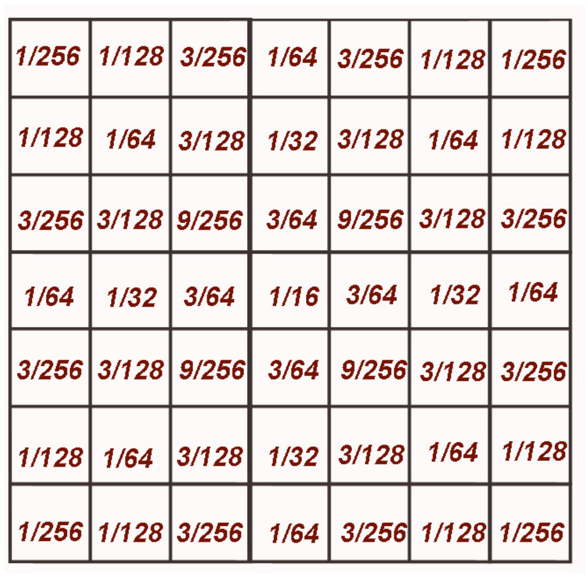
1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Figure 3.3 Anti-aliasing weights *3x3*

In many cases the anti-aliasing with eight neighbors (*3x3* matrix) is not enough to ensure the absence of gaps. For these cases we may perform anti-aliasing using a second level of neighbors also (*5x5* matrix) or even a third level of neighbors (*7x7* matrix). Figure 3.4 summarizes the coefficients of these two cases.



1/81	2/81	1/27	2/81	1/81
2/81	4/81	6/81	4/81	2/81
1/27	6/81	1/9	6/81	1/27
2/81	4/81	6/81	4/81	2/81
1/81	2/81	1/27	2/81	1/81



1/256	1/128	3/256	1/64	3/256	1/128	1/256
1/128	1/64	3/128	1/32	3/128	1/64	1/128
3/256	3/128	9/256	3/64	9/256	3/128	3/256
1/64	1/32	3/64	1/16	3/64	1/32	1/64
3/256	3/128	9/256	3/64	9/256	3/128	3/256
1/128	1/64	3/128	1/32	3/128	1/64	1/128
1/256	1/128	3/256	1/64	3/256	1/128	1/256

Figure 3.4 Anti-aliasing weights *5x5* and *7x7*

Each "*on*" grid cell containing points from the original data set is mapped to the centroid of these points. In the case that the cell does not contain any points (characterized *on* by anti-aliasing) it may be mapped to the centroid of the points of the 8 neighbor cells. The result is then provided as input to the thinning process. Figure 3.5 below depicts an example of a thick cross-section and the result of anti-aliasing and thinning.

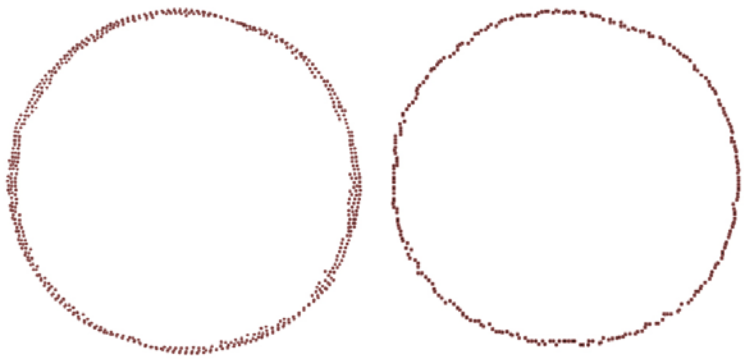


Figure 3.5: The result of quantization and thinning

3.5. Ordering

Thinning a 2D point cloud yields a virtual grid of "*on*" and "*off*" cells where each "*on"* cell represents a point. Figure 3.6a, shows an example virtual grid where all *"on"* cells are colored pink and all *"off"* cells are colored white. The entire grid represents a thin point cloud that is still unordered. At this point we will investigate a 2D Point Cloud Ordering technique that produces an ordered 1D point array which may suit the curve fitting process.

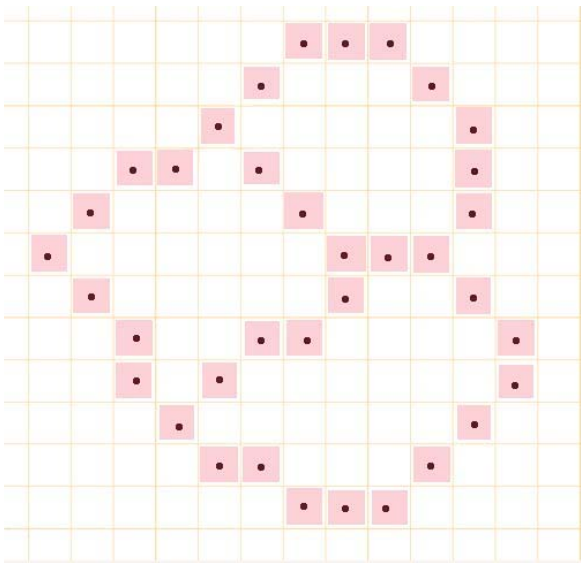


Figure 3.6 Example virtual grid

The virtual grid will first be converted to an undirected graph (Figure 3.8) where each node represents an "*on*" cell and is located at the centroid of the points contained in it. Every node is connected with all of its neighbor nodes in the eight virtual grid directions (Figure 3.7). Since the graph is cyclic we may select any node as graph root, but for simplicity reasons we will select the node that is the closest to the upper left corner of the virtual grid.

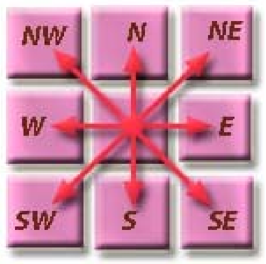


Figure 3.7 Eight neighboring directions

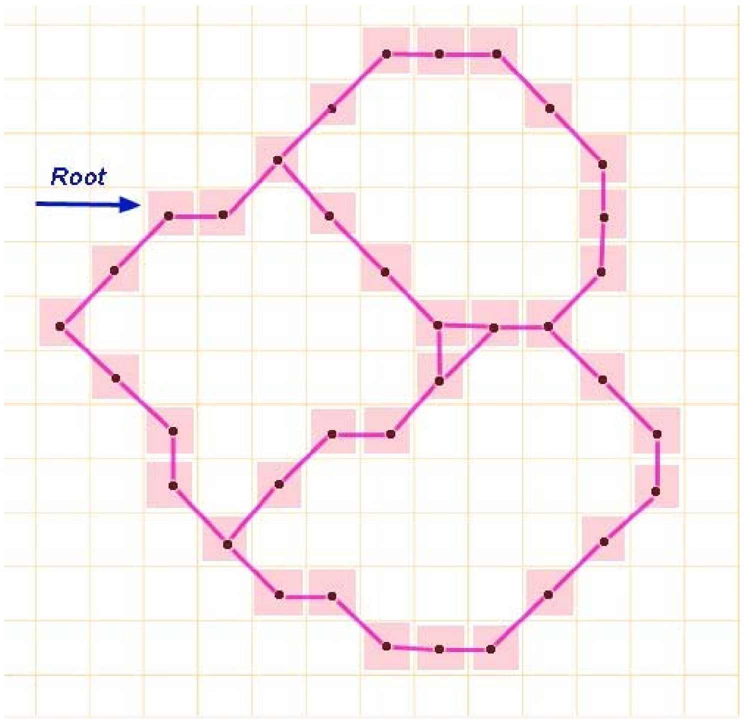


Figure 3.8 Conversion of Virtual Grid to Undirected Graph

To order the 2D point cloud to a 1D point sequence we must traverse the undirected graph using a depth first search DFS [17] algorithm unfolding the entire graph in an ordered sequence of nodes. Depth-first search (DFS) is an algorithm for traversing or searching a tree, or a graph. Intuitively, one starts at the root and explores as far as possible along each branch before backtracking. Formally, DFS is a uniform search that progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hadn't finished exploring. In DFS, each node has three possible colors representing its state:

- White: node is unvisited;
- Gray: node is in process;
- Black: DFS has finished processing the node.

A node with more than two descendant edges is called cross path node and is gray color until all of its descendants are processed. Initially all vertices are white (unvisited). DFS starts from the root node and runs as follows:

1. Mark node *U* as gray (visited).
2. For each edge *(U, V)*, where *U* is white, run DFS for *U* recursively.

3. Mark node *U* as black and backtrack to the parent.

For each performed backtrack to a cross path node, a new sequence of ordered points is created. The final result of DFS process will be a set of independent sequences of ordered points. The union of these sequences of points equals to the initial set of unordered points.

Differentiating from the common DFS algorithm, we will adopt a not fixed traversing order for the descendants of a node, attempting to discover smooth paths through the undirected graph assisting the curve fitting process. As seen in Figure 3.8 most nodes are associated with two edges (ancestor and descendant nodes). Since new sequences of nodes start at cross path nodes when backtracking, it is only important to select the first descendant of a cross path node.

To discover smooth paths through the graph the direction selection of the first descendant of a cross path node will depend on the node's parent direction. These two directions should form the greatest obtuse or acute angle possible (or the smallest reflex angle). Therefore, when the parent of a cross path node is *S* direction, the direction selection should be in the following order: *N, NW, NE, W, E, SW, SE*. In a similar manner, the direction selection of the first descendant of a cross path node could depend not only on the parent node but also on the mean direction of its closest ancestors.

CHAPTER 4. CURVE APPROXIMATION

4.1. Introduction

Approximating a point set by a curve or a set of curve segments is a key problem in reverse engineering of geometric models, pattern recognition, image processing, CAD/CAM, and computer vision over the last three decades. In this work, we focus on an efficient way to approximate a 2D point set, obtained by filleting a 3D point cloud, by a minimum number of quadratic rational Bezier curves. The complexity of the problem is enhanced from the absence of any prior knowledge about the structure of the point set.

The Bezier representation is one that is utilized most frequently in computer graphics and geometric modelling. Quadratic Bezier curves are often used by CAGD developers since they do not require complex computations as other higher degree curves do. However, in practice it is often desirable to approximate conic sections which cannot be represented in Bezier form. Conic sections such as parabolas hyperbolas and ellipses may be adequately represented by Rational Bezier curves.

In this work we propose a method which is suitable for the process of reverse engineering a 3D point cloud by producing a feature-based CAD model. In this context we wish to have a piecewise representation of the curves defining the cross section and satisfying as much as possible the following two criteria:
These criteria result in representations that can be used in feature-based CAD systems for extruding, protruding, and sweeping 2D profiles in 3D solids with satisfactory accuracy, robustness and efficiency (see e.g. [28]).
As we saw in the previous chapter, our method performs a sophisticated thinning of the point set that extracts the significant skeleton. A further examination of this thinned point set will detect subsets of consecutive points that may be fitted by a single quadratic curve. An important property of all quadratic curves is that they exhibit a restricted concavity. This examination will involve the normal vector computation of each line segment formed by two neighboring points. For every subset of points, the first and last points will serve as the end points of the quadratic rational Bezier curves. Finally, by performing an optimization of each curve weight, we will compute the best-fitting quadratic rational Bezier curve for each subset of points.

4.2. Related Work

There is an abundance of research in the literature regarding the general problem of curve reconstruction. Many approaches that have been studied extensively deal with fitting the point data by B-Spline curves [55]. Such methods often require the designer to provide a small number of knots and parameters corresponding to the data points which act as handles to shape the curve. A method [24] based on spring energy minimization approximates an unorganized set of points with a curve which needs a good initial guess of the solution. In SDM [83] a B-spline curve starts from some properly specified initial shape and converges towards the target shape through iterative quadratic minimization of the fitting error. Other approaches [11] use a least square approximation to fit a line segment or a parametric curve of higher degree to the point set. These methods aim at minimizing the sum of squared distances between the set of points and the curve to be fitted. This implies solving a system of linear equations which may require complex and excessive computations while high order polynomials can be highly oscillatory. Several other methods fit point data by rational Bezier and rational B-Spline curves [79] [51] but the problem of setting the weights for good approximation is still a great challenge. Fudos et al [28] describe an interesting representation of conic sections by rational Bezier curves and NURBS. Another iterative algorithm [13] minimizes the sum of squared Euclidean norms with respect to three types of unknowns: the control points, the node values, and the weights. The method uses the projection of the data points on the approximant to improve the node values, and a gradient based technique to update the control point positions and the weight values.

4.3. Point Set Partitioning

First, the ordered point set should be partitioned in subsets of consecutive points that can be fitted by a single rational quadratic Bezier curve. To achieve this, for each point *Pi* we connect all neighbouring points with line segments and compute the average normal vector .

4.3.1. Normal Vector Computation

For a specific point P_i, the average normal vector UR_i is given by:

$$\overrightarrow{UR_i} = \frac{\vec{R}_i}{|\vec{R}_i|} = \frac{\vec{U}_i + \vec{U}_{i+1}}{|\vec{U}_i + \vec{U}_{i+1}|}$$

The above equation averages the normal vectors of the two adjacent line segments $P_{i-1}P_i$ and P_iP_{i+1} (Figure 4.1).

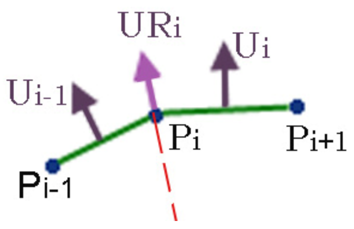


Figure 4.1: Determination of average unit normal vector for a point.

Therefore, for two successive points $P_i(x_i, y_i)$ and $P_{i+1}(x_{i+1},y_{i+1})$ the line segment P_iP_{i+1} that connects the two points is given by

$$P_iP_{i+1}(x_{i+1} - x_{i,}y_{i+1} - y_{i,})$$

The vector normal to P_iP_{i+1} is given by

$$\overrightarrow{N_{i+1}}(y_i - y_{i+1}, x_{i+1} - x_i) \ ,$$

and the unit normal vector to P_iP_{i+1} is

$$\overrightarrow{U_{i+1}}(\frac{y_i - y_{i+1}}{|N_{i+1}|}, \frac{x_{i+1} - x_i}{|N_{i+1}|}), \qquad |N_{i+1}| = \sqrt{(y_i - y_{i+1})^2 + (x_{i+1} - x_i)^2}$$

Every point P_i in the point set P is associated with two unit normal vectors U_i and U_{i+1} derived from the two adjacent line segments $P_{i-1}P_i$, P_iP_{i+1}. The following formula may calculate the average unit normal vector for point P_i (Figure 4.1):

$$\overrightarrow{UR_i} = \frac{\overrightarrow{R_i}}{|R_i|} = \frac{\overrightarrow{U_i} + \overrightarrow{U_{i+1}}}{|\overrightarrow{U_i} + \overrightarrow{U_{i+1}}|}$$

The first and last points (P_0 and P_n) in the point set P are special cases since they do not have two neighbors and therefore, an average unit normal vector may not be computed. For these cases we will use the one neighbor unit normal vector.

In many cases where the data set contains a lot of noise, the average unit normal vector may be computed by averaging a larger number of neighbor line segments. We call this number smoothing neighbors.

4.3.2. Concavity Change Detection

Our aim at this point is to partition the cross section point set into subsets of points that may be fitted by a single quadratic rational curve. Following, we will show how the angle between neighboring normal vectors may drive the partitioning of the point set. Based on the fact that a single quadratic rational curve may approximate correctly a subset of points for which the induced curve exhibits a restricted concavity, indentifying the cross section points where the induced curve changes its concavity (curvature sign) would solve our problem (Figure 4.2).

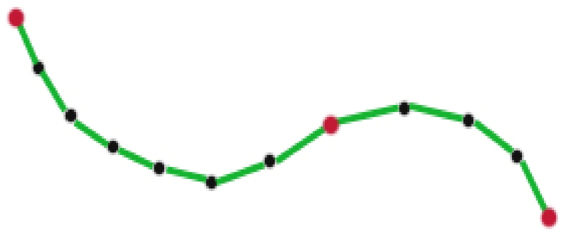


Figure 4.2: Inflection point detection

In differential calculus, point of inflection is defined to be a point on a curve at which the curvature (second derivative) changes sign. This actually means that at an inflection point the curve's concavity changes from convex to concave or vice versa. The curve's normal vector at that point changes rotation direction, which means that the relative rotation of neighboring normal vectors changes sign.

Consequently, to identify the points of the cross section where the induced curve's curvature changes sign we need to compute the relative rotation Φ_i of its average unit normal vectors UR_i with respect to its neighbor's UR_{i+1}.

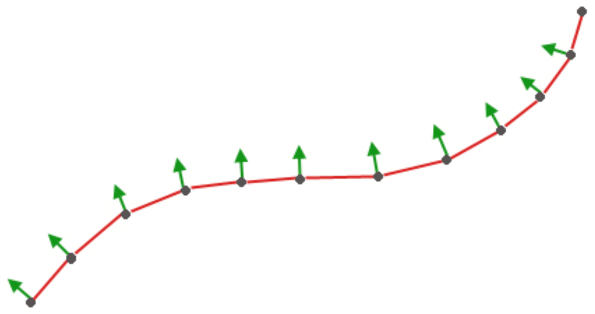


Figure 4.3 Determining the relative rotation

In other words, the algorithm we present is based on the fact that the normal vectors on a concave curve segment turn clockwise, while on a convex curve segment the normal vectors turn counterclockwise (Figure 4.3). On a line segment, the normal vectors have constant direction. The relative rotation Φ_i of vector UR_i with respect to UR_{i-1} is giben by the z-coordinate of the cross product of the two vectors.

$$\Phi_i = (\overrightarrow{UR_{i-1}} \times \overrightarrow{UR_i})_z = (\det\begin{pmatrix} URx_{i-1} & URx_i \\ URy_{i-1} & URy_i \end{pmatrix})_z = (URx_{i-1} \cdot URy_i - URx_i \cdot URy_{i-1})_z$$

Then, in case

- $\Phi_i > 0$, counterclockwise rotation of UR_i with respect to UR_{i-1}
- $\Phi_i < 0$, clockwise rotation of UR_i with respect to UR_{i-1}
- $\Phi_i \simeq 0$, UR_i, UR_{i-1} are almost collinear.

According to the above, each P_i is marked as belonging to a concave or convex or linear curve segment. It is now trivial to detect points where there is concavity change of the induced curve. Each point of inflection may serve as end point for the current processing subset of points B_j and as a start point for the next subset of point B_{j+1}.

For the sake of better curve approximation results, for any point P_i in a certain subset B_j of points, the relative rotation of the average unit normal vector UR_i with respect to the first point's average unit normal vector UR_1 should form no more than $\pi/2$ angle:

$$\forall P_i \in B_j \qquad \cos^{-1}(UR_i \bullet UR_1) \le \pi / 2$$

Our algorithm makes use of a special tree structure where each node represents a point P_i along with all related information about the average unit normal vector on the point and the type of concavity that the induced curve would have. The tree node may also have at most three children that correspond to the three directions of relative rotation that the next point's average unit normal vector could have with respect to the current node.

- *CW* for Clockwise
- *CCW* for Counterclockwise
- *CL* for Collinear

Following is the most important part of the algorithm in pseudocode. *Csp* is the curve's start point, and *Lin* is the last inserted node.

```
Curvature Partitioning
Input: Set of points P
Output: Concavity depending partitioned subsets of points

Set $P_1$ as inflection point
Set Csp=$P_1$, and Lin=$P_1$
Insert $P_1$ at Root
For all points $P_i$ in P
        If slope_difference(UR(Csp), UR($P_i$)) > $\pi/2$
                Mark $P_i$ as inflection point
                Set Csp= $P_i$
        End If
        Insert $P_i$ at Lin wrt (UR(Lin) x UR($P_i$))
End For
End
```

```
Insert
Input: point, Node, Cross product result (point_dir)
Output: Inserted point in the partition tree

If point_dir=Node.dir
        Put point at Node.dir
Else if point_dir<>Node.dir
        If point_dir=Parent(Node).dir
                Put point at Parent(Node).dir
        Else
                Put point at Node.point_dir
        End If
End If
End
```

The relative rotation of the point for insertion drives the partitioning of the point set into subsets. Each subset of points is characterized by a concavity type. A new point for insertion is always appended at the end of the last subset of points provided it preserves the concavity type of the last subset. In the case that the last subset of points is very small (1-2 nodes) and the point for insertion has a different concavity type, the new point is appended at the end of the second to last subset. A new subset of points is started in the case where the relative rotation of a certain point with respect to the first point of the processing subset is greater than $\pi/2$. Consequently, the tree may grow in three directions (CW, CCW, and CL) partitioning the point set into subsets of points that may be approximated by a restricted concavity curve. Any node that is common to two subsets is characterized as inflection point.

Figure 4.4 shows an example point set along with the relative rotations of each average unit normal vector with respect to the first point.

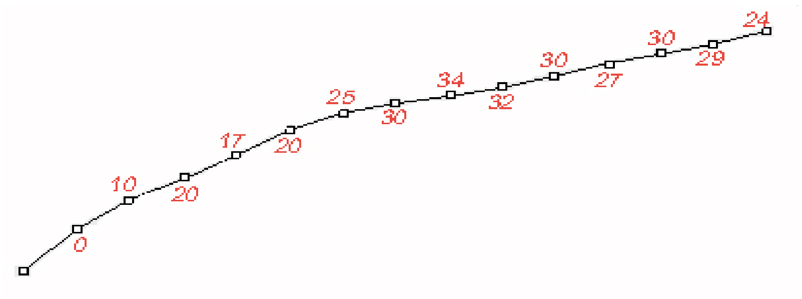


Figure 4.4 Points connected with line segments

Figure 4.5 shows the tree structure created by the algorithm for the point sequence in Figure 4.4. Red arrows show direction *CW*, blue arrows show direction *CCW*, while green arrows show direction *CL*. Inflection points are the light blue squares, and erroneous points are in pink squares.

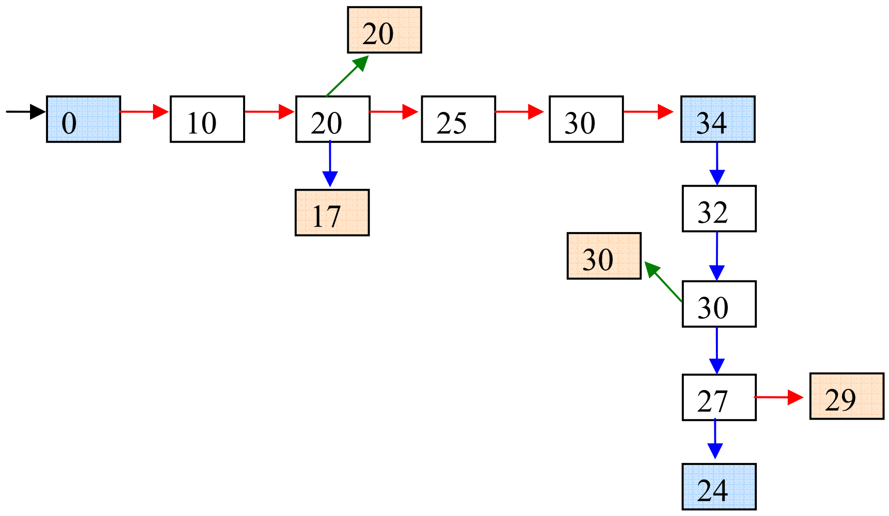


Figure 4.5 Tree structure

In many cases we obtain points with average unit normal vectors that differ significantly from the neighborhood normals. These cases maybe due to point cloud acquiring process or possible wear of the original object. Our algorithm filters out these occasional erroneous points (illustrated in pink in Figure 4.5).

After discarding the erroneous nodes, we end up with a zigzag structure where each branch B_i of it represents a subset of points that may be approximated by a quadratic rational Bezier curve.

Let us denote as L_i the length of branch B_i that is the number of nodes that are between its two inflection points. For instance, in Figure 4.5 we have L_1=4, L_2=3. In some cases, the length of branch may be zero which means that there are no points between two inflection points (Li =0). Figure 4.6 shows an example (L_2=0) of such case where the algorithm was not able to detect that the node with angle 20 is erroneous because its neighborhood tends to rotate slower than it did. A post processing step of the algorithm detects all such cases and eliminates them by characterizing the particular node as erroneous (green color in Figure 4.7), removing the empty branches, and merging where possible branches that split by the erroneous nodes.

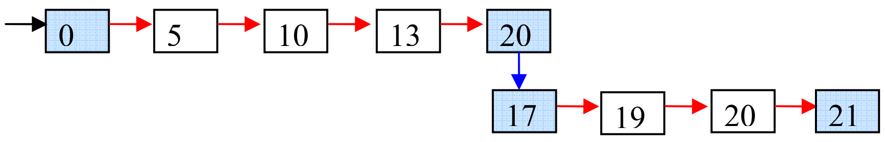


Figure 4.6 Tree with L_2=0

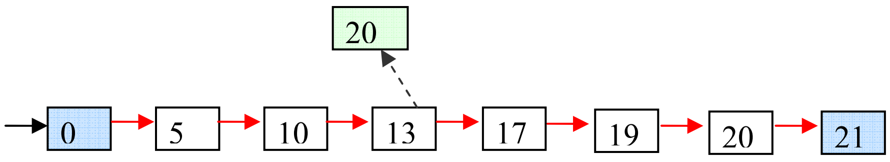


Figure 4.7 Tree resulted by elimination of B_2

In many cases the point set may be very noisy oscillating the average unit normal vectors very frequently and locally. As a result, an abundance of small length point subsets are created. Using a data smoothing algorithm such as the Moving Average could solve the problem but it would increase the time complexity of the entire method. To this effect, we use an alternative that achieves better results by smoothing the normal vectors rather than the actual point set. As described in the previous section, this smoothing is achieved by averaging the normals of a larger number of neighbour line segments. We will call this number smoothing tolerance.

The concavity change detection algorithm divides the ordered set of points into partitions. All points in a certain partition preserve the following relation:

$$P_i < P_k = \begin{cases} UR_1 \bullet URi < UR_1 \bullet UR_k & \textit{partition concave upwards} \\ UR_1 \bullet URi > UR_1 \bullet UR_k & \textit{partition concave downwards} \end{cases} \Bigg\} \text{ , i<k}$$

The above relation is a total order since it preserves:

- Antisymmetry: $(P_i < P_k) \wedge (P_k < P_i) \Rightarrow P_i = P_k$
- Transitivity: $(P_i < P_k) \wedge (P_k < P_m) \Rightarrow P_i < P_m$
- Totality: $(P_i < P_k) \vee (P_k < P_i)$

Also note that if *Si* is the *i*th partition detected

$$\forall S_i,\ \mathrm{S}_{\mathrm{i}_1} = S_{(i+1)_n}$$

where S_{i1} is the first point in the *i*th partition and S_{in} is the last point. In other words the last point of a certain partition coincides with the first point of the next partition. Analysing the running time of the partitioning algorithm, we see that for each point P_i in the point set we perform:

- a computation for the average unit normal vector UR_i of the line segments $P_{i-1}P_i$ and P_iP_{i+1} which takes constant time,
- a computation for the relative rotation of UR_i with respect to UR_{i-1} which takes constant time,
- an insertion of P_i to the tree structure which takes constant time since the last inserted node is always kept track and there is no need to traverse the entire structure.

Consequently, the task of partitioning a cross section is achieved using the above presented algorithm in linear $O(n)$ time for a point set with n points.

4.4. Middle Control Point Computation

The partitioning process gives us a number of subsets of ordered points that may be approximated by a single rational quadratic Bezier curve. Therefore, the start and end points of each approximating curve are already known. Also, note that the end point of each partition coincides with the start point of the next partition. We will now present two methods for determining the middle control point of the fitting curve.

4.4.1. Intersecting the End-point Tangent Lines

The first method makes use of the fact that the middle control point is the intersection of the tangent lines to the Bezier curve on the two end points. These tangent lines may be approximated by the lines that are perpendicular to the average unit normal vectors on the end points, which pass through the end points P_0 and P_2 (Figure 4.8).

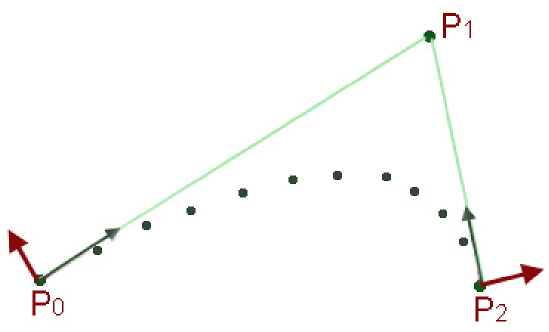


Figure 4.8: Intersecting the end points normals of the average unit normal vectors

Since we already know points P_0 and P_2 and their respective normal vectors *u(ux, uy)* and *v(vx,vy)* we may compute the tangent vectors *u'(-ux, uy)* and *v'(-vx,vy)* which assist us in defining L_1 and L_2:

$$L_1 = P_0 + u's$$
$$L_2 = P_2 + v'r$$

Since control point P_1 is the intersection of lines L_1 and L_2, its coordinates may be computed by solving the following system of linear equations:

$$\left\{\begin{matrix} P_{0x} + u_x's = P_{2x} + v_x'r \\ P_{0y} + u_y's = P_{2y} + v_y'r \end{matrix}\right\} \Rightarrow$$

Therefore, the middle control point is as follows:

$$P_{1,x} = P_{0,x} + \frac{\left(-P_{0,x}\,uy + uy\,P_{2,x} - ux\,P_{0,y} + ux\,P_{2,y}\right)ux}{vx\,uy - vy\,ux}$$
$$P_{1,y} = P_{0,y} + \frac{\left(-P_{0,x}\,uy + uy\,P_{2,x} - ux\,P_{0,y} + ux\,P_{2,y}\right)uy}{vx\,uy - vy\,ux}$$

4.4.2. Approximating the Curve's Maximum Height

Definition: We define as height of a quadratic Bezier curve the normal distance of a – point on the curve from the line segment connecting its end points. The second method involves (Figure 4.9) the fact that for every quadratic Bezier curve the maximum height occurs at *t=0.5*. Also, using De Casteljau algorithm [25] we may derive that at *t=0.5* the tangent to the curve is parallel to the line passing through its end points.

Proof:

If *R(t)* is a quadratic Bezier curve with $t \in [0,1]$ then

$$R(t) = P_0 \cdot (1-t)^2 + 2 \cdot P_1 \cdot t \cdot (1-t) + P_2 \cdot t^2$$

We will compute the value of t for which the tangent to the curve *R'(t)* is parallel to the base of the control triangle P_0P_2.

$$\frac{dR(t)}{dt} = 2 \cdot P_0 \cdot t - 2 \cdot P_0 + 2 \cdot P_1 - 4 \cdot P_1 \cdot t + 2 \cdot P_2 \cdot t$$

Since, the slope of the tangent is equal to the slope of P_0P_2 then solving the following equation for *t* we get

$$\begin{aligned} &\frac{dR(t)}{dt} = P_2 - P_0 \Rightarrow \\ &2 \cdot P_0 \cdot t - 2 \cdot P_0 + 2 \cdot P_1 - 4 \cdot P_1 \cdot t + 2 \cdot P_2 \cdot t = P_2 - P_0 \Rightarrow \\ &t = 1/2 \end{aligned}$$

Therefore, for every quadratic Bezier curve the tangent at t=0.5 is parallel to its control triangle base.

Rotating the curve so that the control triangle base is parallel to the x-axis, makes the tangent to the curve at *t=0.5* equal to zero

$$\frac{dR(0.5)}{dt} = 0$$

This means that *R(0.5)* is a local extreme (either local maximum or local minimum) for the rotated curve *R*. In other words, the quadratic Bezier curve *R* is at the maxium height from the x-axis and the control triangle base at *t=0.5*. The same happens if the curve is not rotated.

Approximating the point on the curve with maximum height *R(0.5)* with the point from the data set that has maximum distance from the line segment $P_0\,P_2$ may give us the location of the middle control point P_1:

$$P_1 = 2 \cdot R(0.5) - \frac{(P_0 + P_2)}{2}$$

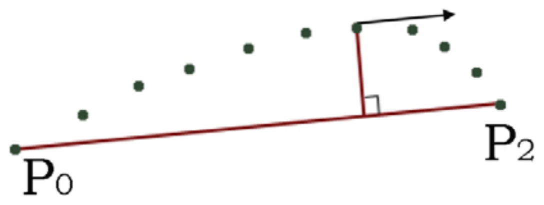
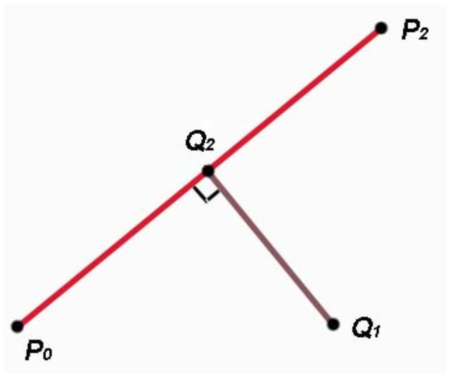


Figure 4.9: Tangent at *t=0.5* is parallel to P_0P_2

Following, we will determine the normal distance of a point Q_1 from a line segment P_0P_2. The equation of a line defined through two points P_0 (x_0,y_0) and P_2 (x_2,y_2) is

$$P = P_0 + u \cdot (P_2 - P_0)$$



The normal distance of point Q_1 (x_3,y_3) from the line P may be determined by computing the point Q_2 of intersection of the normal vector to the line P that passes through Q_1 and the line P. This may be succeded by employing the dot product of the normal line segment and line

$$(Q_1 - P) \bullet (P_2 - P_0) = 0$$

Substituting the equation of the line gives

$$[Q_1 - P_0 - u \cdot (P_2 - P_0)] \bullet [P_2 - P_0] = 0$$

Solving this equation for *u* we get

$$u = \frac{(x_3 - x_0) \cdot (x_2 - x_0) + (y_3 - y_0) \cdot (y_2 - y_0)}{\left\| P_2 - P_0 \right\|^2}$$

Consequently, we may obtain the point of intersection by substituting *u* into the equation of the line *P*

$$x = x_0 + u \cdot (x_2 - x_0)$$
$$y = y_0 + u \cdot (y_2 - y_0)$$

Therefore the distance between the point Q_1 and the line P is the distance between Q_1 and Q_2

4.5. Low Degree Bezier Curve Approximation

The Bezier representation is one that is utilized most frequently in computer graphics and geometric modelling. Quadratic Bezier curves are often used by CAGD scientists since they do not require complex computations as other higher degree curves do. However, in practice it is often desirable to approximate conic sections which cannot be represented in Bezier form. Conic sections such as parabolas hyperbolas and ellipses may be adequately represented by Rational Bezier curves. Non rational Bezier curves are a special case of rational Bezier curves. For these reasons, we will focus on constructing Rational Quadratic Bezier curves. In curve theory, a rational quadratic Bezier curve is defined by

$$P(t) = \frac{\sum_{k=0}^{2} w_k p_k B_k^2(t)}{\sum_{k=0}^{2} w_k B_k^2(t)}, 0 \le t \le 1$$

A 2nd degree Bezier curve requires 3 control points p_k: a start point p_0, an end point p_2, and a 3rd control point p_1 which is obtained by the methods we described in the previous section.

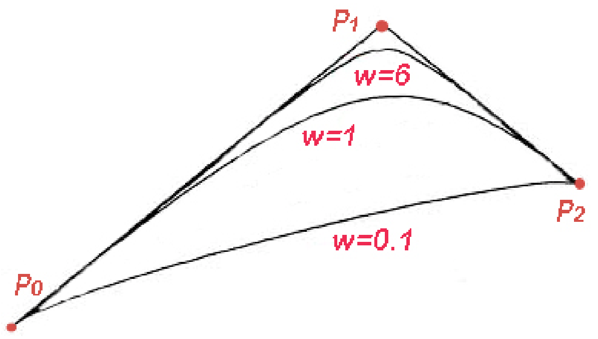


Figure 4.10: Varying weight of a quadratic rational Bezier curve

The B_k terms in the above formula represent the 2nd degree Bernstein polynomials, while the terms w_k are the associated with each control point weights. Setting all weights equal to one to the above formula represents an ordinary non rational Bezier curve. Increasing the weight of a control point causes the curve to move towards the associated control point (Figure 4.10).

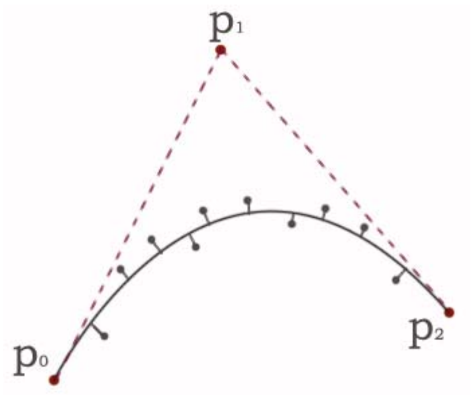


Figure 4.11: Point distances from the curve

The curve fitting process fits equations of approximating curves to the raw field data. Nevertheless, for a given set of data, the fitting curves of a given type are generally not unique. Thus, a curve with a minimal deviation from all data points is desired (Figure 4.11). For cases where a rational Bezier curve is approximated the best-fitting curve can be obtained by varying the control point weights (Figure 4.11). To obtain the best fitting rational Bezier curve we will perform constrained minimization of an objective function subject to a set of constraints.

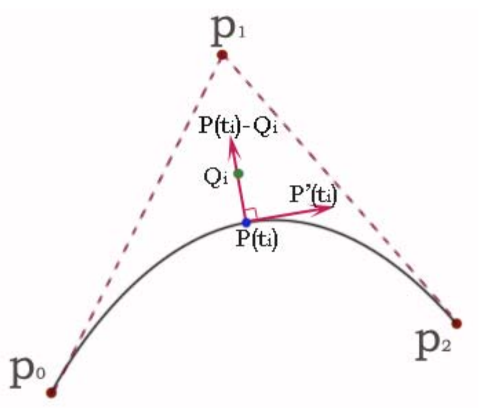


Figure 4.12: The vector Q_i $P(t_i)$ perpendicular to the tangent $P'(t_i)$

A rational Bezier curve *P(t)* that best approximates the given set of 2D points *Q* on a specific cross section is the one that minimizes the sum of the distances of the points from the curve:

$$f(w_1) = \sum_{i=1}^{n} \left\| Q_i - P(t_i) \right\|^2 = \min$$

Also note that each vector $\overrightarrow{Q_i P(t_i)}$ is normal to the tangent of the curve at t_i (Figure 4.12). This means that their inner product is zero. To minimize the sum of square distances, the above equation will serve as the objective function while the equation below will provide *n* constraints.

$$\forall Q_i, i = 0 \ldots n, \quad g_i(w_1) = P'(t_i) \bullet (Q_i - P(t_i)) = 0$$

Without loss of generality we can set $w_0 = w_2 = 1$. The objective function is non linear and twice differentiable. Interior point methods based on a logarithmic barrier function have been widely used for nonlinear programming [65],[80]. To allow convergence from poor starting points, barrier and augmented Lagrangian merit functions may be used [30]. For this reason we will use a NLP solver. A major approach for NLP is the Interior Point method which uses a logarithmic barrier function.

$$\Phi(w_1, \mu) = f(w_1) - \mu \sum_{i=1}^{n} [Log[-g_i(w_1)]]$$

μ is a positive number, known as the penalty number. This method is based on the fact that as we move closer to a constraint boundary, g_i tends to *0* causing a large term to be added to the objective function. Thus, the method keeps the solution away from the constraint boundaries. This method was implemented with IpOpt software [76].

Optimizing the objective function subject to these constraints may give the middle weight value of the rational Bezier curve that best fits the given set of points. Depending on the size of the data set that needs to be fitted, the optimization task could be a long and cumbersome effort. For this reason, we will perform an extra step of evaluating a starting value for the middle weight by making use the barycentric coordinates of each point with respect to the control triangle.

$$w_1 = \frac{\tau_1}{2\sqrt{\tau_0 \tau_2}}$$

Using the above equation [25] we may compute the value of the middle weight of the curve that passes through a certain point of the data set. τ_0, τ_1, τ_{22} are the barycentric coordinates of Q_i with respect to the triangle formed by the three control points P_0, P_1, P_2 of each rational Bezier. Barycentric coordinates using the following equations.

$$\tau_0 = \frac{area(L_i, P_1, P_2)}{area(P_0, P_1, P_2)}, \quad \tau_1 = \frac{area(P_0, L_i, P_2)}{area(P_0, P_1, P_2)}, \quad \tau_2 = \frac{area(P_0, P_1, L_i)}{area(P_0, P_1, P_2)}$$

$$and \quad area(a,b,c) = \frac{1}{2}\begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{vmatrix}$$

Consequently, for each point Q_i in the data set, we may compute a value w_{1_i} for the middle weight of the curve. The value of the middle weight that will be selected is the one that minimizes the sum of square distances of the points in the data set from the curve. This value may be used as a starting value in the optimization process that was described above.

CHAPTER 5. CONSTRAINTS FOR EDITABILITY

5.1. Introduction

A new generation of CAD systems has become available in which geometric constraints can be defined to determine properties of mechanical parts. The new design concept, often called constraint-based design or design by features offers users the capability of easily defining and modifying a design, but introduces the problem of solving complicated, not always well defined, constraint problems. In this chapter, we present the development of a user-friendly interactive system for imposing and solving geometric configurations inside cross-section (intra cross-section constraints) and among two or more cross-section (inter-cross-section) constraints. The system uses a powerful graph-constructive constraint solving method presented in [27], capable of efficiently analyzing certain classes of well-determined, over-determined and under-determined configurations. Minimal systems of geometric constraints that are not solvable by the core constructive method are detected and may either be handled by a numerical method and treated afterwards as rigid bodies, or edited by the user. A main issue pertinent to geometric constraint solving is the solution selection problem. To this end, we have provided an interactive tool for navigating the constraint solver, to the intended solution. Consistent over-determined sub-configurations can be detected, interactively relaxed and solved appropriately. Under-determined subsystems are detected, isolated and subsequently presented to the user annotated with all possible constraint addition choices for interactive editing.

5.2. Related Work

The importance of being able to clearly describe a piece of geometry in a clear and unambiguous way has been realized since ancient times. For the very simple 2D shapes, Euclidean geometry showed us precisely what information was needed to completely define them. However, the difficulties of describing more complex geometries remained for many years until the use of computers became common. With the use of CAD/CAM, it was soon realized that the increased ease of entering more and more complex geometries further necessitated the invention of novel formalisms capable of capturing these geometries in a simple but rigorous manner. A number of authors have conducted research in the field of geometric constraints and regularities for reverse engineered models.

In previous work, feature based design has been approached as an extension of the CSG paradigm [69]. In a CSG construction, a solid is built from standard primitives by regularized Boolean operations. The solid then, is represented by a tree structure in which the leaves are solid primitives and the interior nodes are Boolean operations and rigid-body transformations. Although limited, this approach provides well-defined design semantics.

One of the central research directions in computer vision is 3D object recognition. In 3D object recognition the term of surface characterization refers to the computational process of partitioning surfaces into regions with equivalent characteristics. There are many surface characterization algorithms that make use of differential geometry. The local shape of a surface is central to object recognition. It may be determined by using Gaussian and mean curvatures which combine the first and second fundamental forms of the surface [48] to obtain scalar surface features which are invariant to rotation, translation and re-parameterization. Surface shapes may be characterized by the sign of the mean curvature and Gaussian curvature. These curvatures may be computed directly from the point set acquired from the data acquisition process.

In [63], a laser projection system and an image processor [64] are used for determining a fixed set of horizontal cross sections of the recognized object which is placed on a turntable in a stable vertical orientation. For each horizontal cross section boundary based Fourier shape representations are computed. Constraints between two cross sections may be defined such as horizontal strain, section shape, torsion, and displacement.

In a more recent work [15], semantics for the creation of generated features are defined. This work is based on a neutral, high-level design representation, called *Erep (editable representation*), which allows design modifications based on a general design paradigm. This framework considers generated features based on a planar profile and then revolved, swept and extruded in 3D shape.

Often the surface fitting step is enhanced by imposing a set of constraints and then a simultaneous (as opposed to sequential) fitting is attempted using the constraints as a set of side conditions that must be satisfied by the surface parameters [8].

Another approach [78] is to drive the segmentation and surface fitting phases using pre-defined features like slots and pockets whose abstract location and type has been determined by the user.

[53], [40] describe the importance of a post-processing step, often called *beautification*, which adjusts the model to reflect more closely the intended object. This step involves the analysis of the model to find geometric regularities, the selection of an appropriate consistent set of regularities which renders the original design intent, and finally, the reconstruction of an improved model using geometric constraints without further reference to the point data which avoids the computational expense of constrained fitting.

5.3. Defining Cross Section Features by imposing Geometric constraints

The objective of the entire method is to obtain an editable CAD model that would assist us in redesigning the original object. Editability in CAD is commonly achieved by using geometric constraints. When using the term constraint in CAD we usually refer to geometric dimensions and relations (lengths, angles, tangency, parallelism, perpendicularity, etc.) used to define accurately a particular solid geometry. Even though it is not necessary, object symmetry may provide additional auxiliary information in constraining an object.

Since the result of curve fitting is a set of rational quadratic Bezier curves, the user may define constraints that involve the control triangle – polygon and the curve weights. Taking into account that a certain curve may be approximated by more than one set of control points – weights, defining constraints on curve weights is not always a trivial task. We will provide an alternative way to constrain a rational quadratic curve the curve's maximum height.

The middle weight of a rational quadratic curve specifies its maximum height which, as proved earlier, occurs at $t=0.5$. Therefore, the curve's maximum height may be derived by the normal distance of point $R(0.5)$ from the line segment P_0P_2

In this section we will categorize the geometric constraints and how they are adopted by our method to capture design intent and provide for redesign.

5.3.1. Intra-Cross Section Constraints

The first category of constraints is associated with the geometric and topological relationships among entities in a single cross sections which we will call intra – cross section constraints:

- **Point – line segment coincidence**: special points (curve end points, curve control points, center of circle, etc) or line segments may coincide or be part of the same infinite line.
- **Tangency:** an arc is tangent to a specific curve
- **Distance from a curve or point:** an arc is located at some distance from a specific curve or point
- **Angle with a curve:** an arc (its tangent) forms an angle with another curve or with a line segment at a specific point on the curve.
- **Parallel – Perpendicular line segments or tangents:** a line segment is parallel or perpendicular with another line segment or tangent line.

5.3.2. Inter-Cross Section Constraints

The second category of constraints is associated with the geometric and topological relationships among the contours of different cross sections which we will call inter – cross section constraints:

- **Point co-linearity:** a point from cross section C_A is on the same infinite line with a point from cross section C_B
- **Points on same curve:** a point from cross section C_A is on the curve with a point from cross section C_B
- **Co planar line segments**: a line segment from cross section C_A is on the same infinite plane with a point from cross section C_B
- **Equality or relation of distances:** a specific distance in cross section C_A is equal or related with another distance from cross section C_B
- **Equality or relation of angles:** an angle in cross section C_A is equal or related with an angle in cross section C_B
- **Curve translation**: A curve in cross section C_A is translated by a specific distance and direction in cross section C_B. This constraint may fit cases of slanted or tori objects.
- **Curve scaling**: A curve in cross section C_A is scaled by a certain scaling factor in cross section C_B. This constraint may fit cases of tapered objects.

5.4. Geometric Constraint Solving

We build a system of geometric constraints that captures user intent and at the same time guarantees solid model robustness and accuracy. Symmetry derived geometric constraints are considered to be *strict* with no tolerance allowed. User constraints fall under two categories: (i) *strict*, for which no tolerance is allowed and (ii) *flexible*, for which we wish to acquire the best approximation but we cannot guarantee their strict enforcement. For the purposes of usability we allow only for constraints that can be expressed as equation (e.g. distances, angles, relations of distances and angles, co-planarity, coincidence, tangency). Inequalities can also be handled but they tend to confuse the user with the multiplicity of solutions that they imply. Each flexible constraint has an associated weight which expresses its importance and is derived by two factors: explicitly by a user preference and implicitly by the rank in the user constraint enforcement. Finally the weighted sum of flexible constraint deviations properly normalized is used as the objective function to minimize and the strict constraints are used as the set of constraints for this non-linear optimization problem. To solve this system we employ a local non-linear optimization algorithm from IpOpt [76]. The disadvantage of this method is that it may be trapped in local minima, which makes it depending heavily on the initial configuration. The user is thus advised to make incremental editing. Using global optimization methods or other constraint solving techniques is an interesting research problem [27].

CHAPTER 6. RECONSTRUCTING SOLID PARTS

6.1. Introduction

A novel approach to surface reconstruction from parallel slice contours will be presented in this chapter. The method preserves the topology of the surface without altering the original contours. Main goal of surface reconstruction from contours is to find a best surface consistent with the observed contours. In general, there is an infinite number of surfaces consistent with any set of contours. A surface reconstruction algorithm must choose the best match to the real object. This is accomplished by taking into account the imposed constraints. The quality of the resulting surface depends on the constraints' ability to model the desired solution. In this chapter we will see a specialized point sampling strategy that would assist us in the reconstruction process.

The boundary of the material of interest to be reconstructed is defined by the set of parallel cross sections. Each cross-section contains a set of curves forming a closed contour. As distance separates the sections, information about the region of the object between two cross sections is not recorded. Often, this lost information describes places where ramifications occur in the surface of interest.

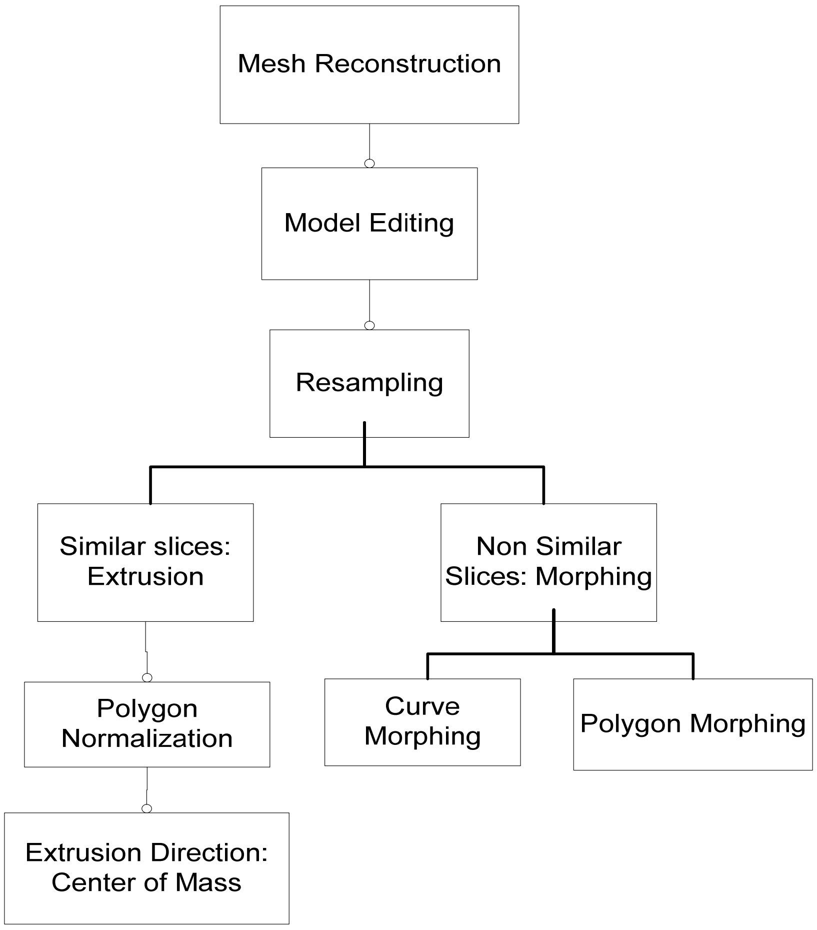


Figure 6.1: Decomposition of Reconstruction process

This causes shape differences or different number of contours between adjacent sections. A way to approach this problem is creating intermediate sections representing the place where the ramifications occur. The rest of the chapter proposes a method (Figure 6.1) that reconstructs solid parts from similar adjacent cross sections using feature construction techniques, and non-similar adjacent cross sections using either a curve based or a polygon based morphing technique.

6.2. Related Work

The problem of reconstructing the surface of a solid object from a series of parallel planar cross sections has been treated by the specialized literature in the past. Since the very early work of Keppel [36] on tiling between parallel polygonal contours, numerous algorithms were introduced for parallel inter-slice interpolation. The problem is considered to be quite difficult because the topology of the contours may change between slices. Some progress was made with the introduction of the Delaunay-based technique of Boissonnat [10], and the method of Bajaj et al. [6]. Both approaches attempt to handle the most general case, in which the geometries and topologies of the contours in every slice are totally unrestricted. Bajaj [6] detects the parts of contours with very different shape and applied a method using edge voronoi diagram to tile them. Another type of skeleton using an approximation of edge voronoi diagram was proposed by Oliva [54]. Subsequently, Barequet and Sharir [5] suggested an interpolation method based on geometric hashing. In this method similar sub-contours are identified first and stitched together, while the remaining contour portions are triangulated so as to minimize the surface area of the reconstructed solid. Later, Barequet et al. [4] suggested another interpolation algorithm that uses the medial axis of the overlay of the two slices. This method generates a smooth and intuitive reconstruction since it inherently captures the differences between the slices. Meyers [52] also uses medial axis to obtain information about the relationships of the vicinity among the regions where ramifications occur. Sloan et al [68] suggest the creation of artificial intermediate sections between adjacent different sections. Levin [45] builds a set of intermediate contours between contours of adjacent sections by calculating the distance field for each point in every section.

6.3. Point Resampling

The construction of the object's surface requires the generation of parts of the surface that lies in between two slices using triangulation. Triangulation may not be based on the thinned point set of each slice because its density would result in creating many small area triangles. Another shortcoming is that since there is a known small distance error between the slice points and the fitted curves, the triangulated model is likely to be rough, containing bumps. To resolve these issues, we must resample the point set to obtain a reduced set of points.

Point sampling is an important intermediate step for a variety of computer graphics applications. Specialized sampling strategies have been developed to satisfy the requirements of each problem. In this section, we present a sampling technique for 2D models. Our sampling domain is the set of points on a single cross section. Aim of the technique is to generate evenly spaced samples by subdividing the sampling domain into non overlapping parts.

Given a data set of points $Q=\{Q_i\}$ for which we have already determined the best fitted set of rational quadratic Bezier curves $P=\{P_k\}$, we suggest replacing the points Q with a reduced set of new points $R=\{R_j: R_j=P(t_j)\}$ that satisfy the curve equations.

In a previous chapter we fitted a rational Bezier curve on the points of each cross section. A Rational Bezier curve is usually defined over the interval *[0, 1]* but it may also be defined over any interval *[0, c]*. The part of the curve that corresponds to *[0, c]* may also be defined by a Bezier polygon. To subdivide the curve [61] to *k* equal length arcs we would first divide the interval *[0, 1]* into *k* subintervals of length *1/k*. The end points of each arc R_i are $P(t_{i-1})$ and $P(t_i)$ where $t_i= i/k$ and $i=0..k$. The length of each chord $\|P(t_i)\ P(t_{i+1})\|$ converges to the length of the arc R_i between t_i and t_{i+1} when *k* is a rather large value:

$$\Lambda = \sum_{i=0}^{k} \left\| P(t_i) - P(t_{i+1}) \right\|$$

Considering that the size of the sample set *S* of points is μ:

$$\forall s_i \in S, \quad i = 0...(\mu - 1), \qquad \left\| s_i s_{i+1} \right\| = \Lambda / \mu$$

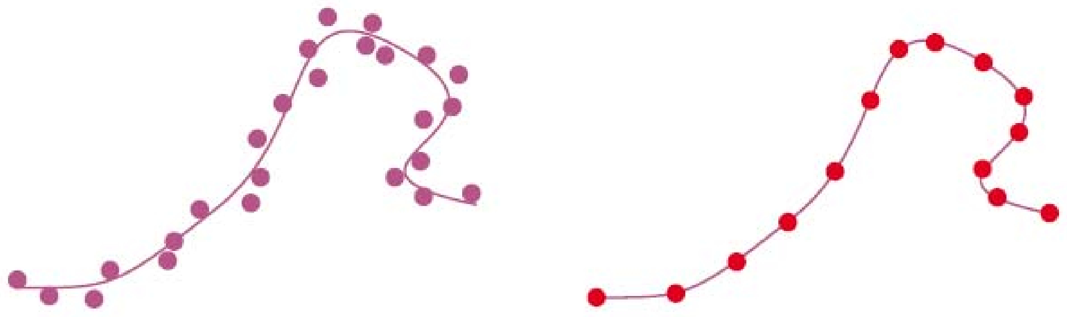


Figure 6.2: Sample points (in red)

The last relation ensures that all points in the sample set *S* are evenly spaced by a distance of Λ/μ. All other points that do not satisfy the above relation are discarded and will not be used in the surface reconstruction process (Figure 6.2).

6.4. Similar Adjacent Cross-Sectional Features

The main design paradigm of CAD systems nowadays is feature-based design. Feature – based systems contain a vocabulary of design elements as long as object operations that are used to create the intended design. By performing operations such as extrusions, protrusions and cuts on the design elements (cylinders, cones, parallelepipeds, pyramids etc) we may generate the desirable design. Our feature based CAD model provides modeling primitives that may be enforced low-level constraints, reducing the number of variables necessary to represent an object. Constraint based techniques apply high-level constraints over these features enforcing the hypothesized design intent.

In this section we will investigate ways for converting 3D point cloud to a set of features describing exactly the original object's geometry and satisfying all imposed constraints. Based on the fact that our method generates a set of planar consecutive curves for each cross section, we will be considering features that are based on a planar profile swept into a 3D shape by an extrusion operation. Consequently, our final CAD model will consist of a set of connected features. This makes our final CAD model easily modifiable since we only need to deal with modifying the geometry of the features.

As shown in Figure 6.3, sweeping a planar profile creates a tubular surface that its bottom base is the planar profile and its top base is the same planar profile translated. Let two planar profiles P_1 and P_2 consisting of a set of quadratic rational Bezier curves. P_1 is said to be similar to P_2 if and only if all curves in P_1 are congruent to all curves in P_2 up to the same affine transformation. In other words, profile congruence requires curve congruence. Bezier curve congruence property implies control triangle congruence and middle weight equality. Therefore, two planar profiles are invariant if and only if all respective control triangles are congruent and all respective weights equal.

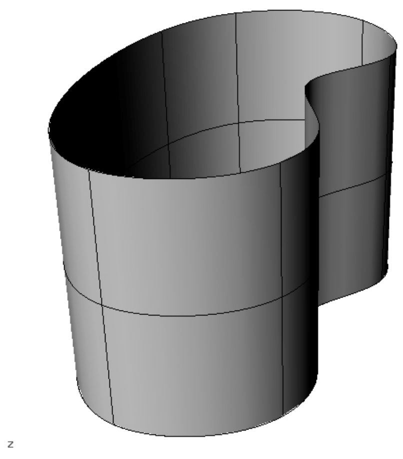


Figure 6.3 Sweeping of a planar profile

As it is defined in Euclidean geometry, triangles are congruent when all corresponding sides and interior angles are equal. These triangles will have the same shape and size. However, they can be in a different location, rotated or flipped over. Consequently, two triangles *R* and *R'* are congruent even if *R* is a mirror of *R'*. In contrast, in our method we are interested in triangles that may not be mirror images of each other because they generate different curves that cannot be interpolated.

Definition 1: We define as topologically congruent in 2D two polygons that are the same up to rotation and translation.

Therefore, two profiles P_1 and P_2 are said to be congruent if and only if all control triangles are topologically congruent.

The extrusion direction vector may be either perpendicular or it may form any angle with the profile plane. As long as the starting and ending cross sections are invariant, computing the center of mass of both cross sections may derive the extrusion direction vector. This is true even for cases where the second polygon is scaled or rotated.

Detecting similarity between two or more polygons is performed based on two key ideas:

- normalizing a shape about its diameter and
- the notion of the ε-envelope.

Normalizing about the diameter. In order to detect whether two or more polygons are similar some kind of "normalization" is applied so that the matching is translation-, rotation-, and scaling-independent. In previous work researchers would normalize each shape about each of its edges: they translate, rotate, and scale the shape so that the edge is positioned at *((0, 0), (1, 0))*. Although this approach gives good results in many cases, it would fail to detect similarity between slightly distorted shapes.

In our method, instead of normalizing about the edges, we normalize about the diameter of the shape, i.e., by translating, rotating, and scaling so that the pair of shape vertices that are farthest apart are positioned at *(0, 0)* and *(1, 0)*. This ensures better results, because the diameter is less susceptible to local distortion which is very common in shapes extracted using thinning and other point-based techniques.

The ε-envelope [29]. Polygon matching works by considering a "fattened" version of the one polygon which is computed by taking lines parallel to the query shape edges at some distance on either side; we call this fattened shape the *ε*-envelope. The good matches are expected to fall inside or at least have most of their vertices inside the ε-envelope even for small *ε*. Therefore, if we start by using a small initial value of ε and keep increasing it, we expect to collect the good matches after a few iterations of this procedure.

The *ε*-envelope can be seen as a collection of trapezoids of height *2ε*, one for each edge of the query shape. (For simplicity, we assume that ε is such that no two trapezoids are overlapping; the method can be extended to handle overlapping trapezoids.)

The center of mass of a planar profile may be approximated by the center of mass of its convex hull polygon. A better approximation could be the minimal control polygon's center of mass. To determine the minimal control polygon of a planar profile all quadratic Bezier middle control points are used as polygon vertices. The minimal control polygon must include all Bezier curves. While a Bezier curve is always inside its control triangle, the minimal control polygon may not always include a curve. The first polygon in Figure 6.4 depicts this case.

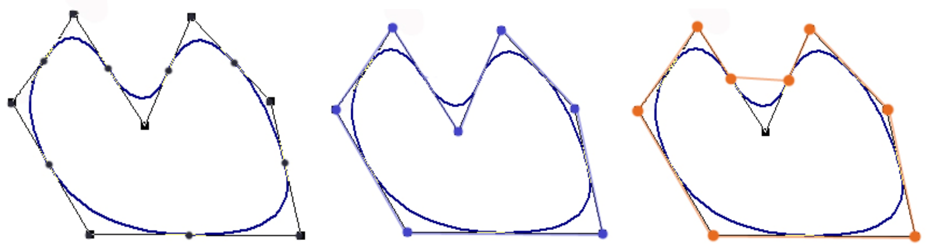


Figure 6.4 Derivation of minimal Control Polygon

In the case where a curve is excluded from the minimal control polygon we edit the list of polygon vertices by replacing the specific middle control point with the respective Bezier curve’s end points. The second and third polygons in Figure 6.4 illustrate this procedure.

If one point of the curve is inside the polygon then the entire curve is inside also. Therefore, we only need to determine if a single point on the curve is inside the polygon. A ray casting algorithm may be used to determine whether a specific point is included in the control polygon. The algorithm is based on a simple observation that if a point moves along a ray from infinity to the probe point and if it crosses the boundary of a polygon, possibly several times, then it alternately goes from the outside to inside, then from the inside to the outside, etc. As a result, after every two "border crossings" the moving point goes outside. Therefore, the number of intersections is an even number if the point is outside the polygon, and it is odd if it is inside (Figure 6.5).

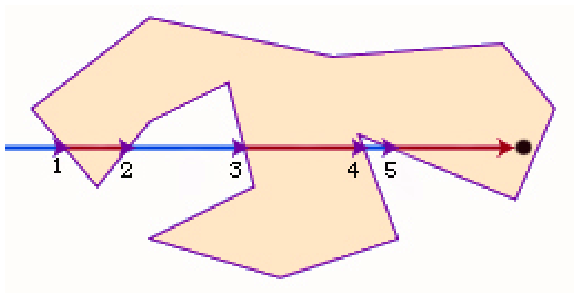


Figure 6.5 Ray casting algorithm: 5 crossings, probe point inside

Both the convex hull and the minimal control polygon of a profile are n-polygons. Computing the center of mass of an n-polygon $\{A_1, A_2, \ldots, A_n\}$ is rather straightforward using the following equation:

$$CM = \frac{1}{n} \cdot \sum_{i=2}^{n} \overrightarrow{A_1 A_i}$$

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. The *orthogonal extrusion* of R is defined to be a solid obtained by sweeping profile *R* in a direction perpendicular to *C* up to the parallel cross section *C'* resulting to one or more tubular surfaces (Figure 6.6).

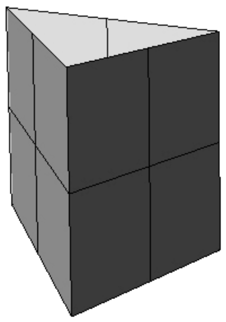


Figure 6.6 Orthogonal Extrusion

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. Also, let *V* be a vector on the line that connects the centers of mass of the profiles *R* and *R'*. The *oblique extrusion* of *R* is defined to be a solid obtained by sweeping profile *R* in a direction specified by vector *V* up to the parallel cross section *R'* resulting to one or more oblique tubular surfaces (Figure 6.7).

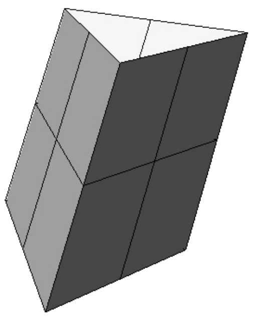


Figure 6.7 Oblique Extrusion

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. Also let profile *R'* be determined by a *θ* angle rotation of profile *R* with the center of rotation being the center of rotation. Let's also denote as *d* the distance between the centers of mass of *R* and *R'*. The *orthogonal rotated sweeping* of *R* is defined to be a solid obtained by sweeping profile *R* in a direction perpendicular to *C* and the same time rotating the profile *R* with a rate of rotation ϑ/d up to the parallel cross section *R'* resulting to one or more rotated tubular surfaces (Figure 6.8).

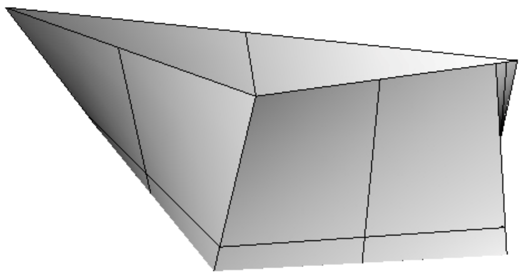


Figure 6.8 Orthogonal Rotated Sweeping

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. Also let profile *R'* be determined by a θ angle rotation of profile *R* with the center of mass being some point *P*. This rotation is equivalent to a θ angle rotation of profile *R* with the center of mass being center of rotation, followed by a translation in the same plane. Therefore, we may denote as *V* the vector on the line that connects the centers of mass of the profiles *R* and *R'*. Let's also denote as *d* the distance between the centers of mass of *R* and *R'*. The *oblique rotated sweeping* of *R* is defined to be a solid obtained by sweeping profile *R* in a direction specified by vector *V* and the same time rotating the profile *R* with a rate of rotation ϑ/d up to the parallel cross section *R'* resulting to one or more oblique rotated tubular surfaces (Figure 6.9).

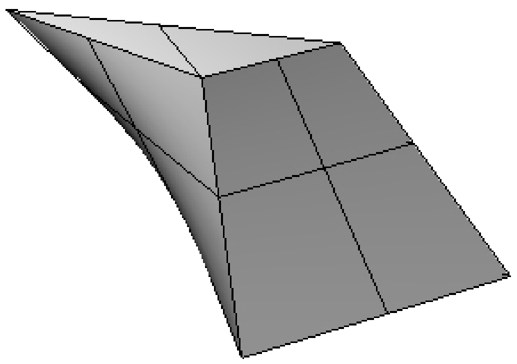


Figure 6.9 Oblique Rotated Sweeping

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. Also, let μ be the linear scaling factor of the two profiles and *d* the distance between the centers of mass of *R* and *R'*. The *orthogonal linear scaled skinning* of *R* is defined to be a solid obtained by skinning profile *R* in a direction perpendicular to *C* and the same time scaling the profile with a scale rate μ/d , up to the parallel cross section *C'* resulting to one or more frustrum surfaces (Figure 6.10).

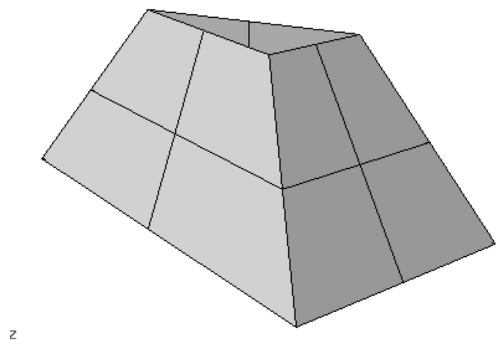


Figure 6.10 Orthogonal Linear Scaled Skinning

Let *R* and *R'* be two congruent closed profiles on the parallel cross sections *C* and *C'* respectively. Also, let μ be the linear scaling factor of the two profiles, *d* the distance between the centers of mass of *R* and *R'*, and *V* the vector on the line that connects the centers of mass of the profiles *R* and *R'*. The *oblique linear scaled skinning* of *R* is defined to be a solid obtained by skinning profile *R* in a direction specified by vector *V* and the same time scaling the profile with a scale rate μ/d, up to the parallel cross section *C'* resulting to one or more frustrum surfaces (Figure 6.11).

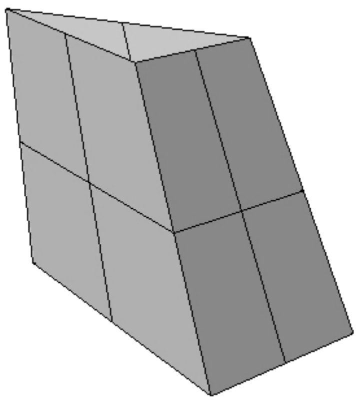


Figure 6.11 Oblique Linear Scaled Skinning

6.5. Non-similar Adjacent Cross-sectional Features

So far, we investigated ways for reconstruction by extrusion of solid parts that are between similar adjacent cross sections. In this section we will investigate ways to reconstruct solid parts that are between non-similar adjacent cross sections.

6.5.1. Curve-based Morphing and Interpolation

When we dealt with similar adjacent cross sections the same sweeping strategy was applied to the entire profile. For cases with non-similar adjacent profiles, we cannot apply the same sweeping strategy to the entire profile. Curve-based morphing is an advanced type of sweep where each curve is being applied a different sweeping strategy.

Two adjacent cross sections C_1 and C_2 are non-similar when there is at least one curve in C_1 that is non-similar (not congruent) to its respective curve in C_2. Figure 6.12 depicts a case with two non-similar adjacent profiles.

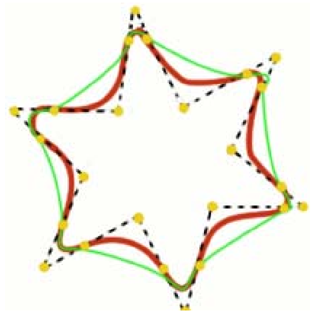


Figure 6.12 Non-Similar profiles

Curve-based morphing breaks down the profile sweeping problem to a number of curve sweeping problems. Since a rational quadratic Bezier curve is fully specified by its control triangle and its middle weight, we only need to sweep the source control triangle to the destination control triangle and gradually change the value of the middle weight from the source to the destination value (see the following equations). Figure 6.13 illustrates this type of sweep. Morphing curve P to Q is equivalent in morphing between their control triangles. The middle weight undergoes a gradual transition from value W_P to W_Q. Let R be a triangle in some ith state of morphing P to Q. Each control point R_i should be on the line segment P_iQ_i such that

$$R_o(t) = P_0 + t(Q_0 - P_0)$$
$$R_1(t) = P_1 + t(Q_1 - P_1)$$
$$R_2(t) = P_2 + t(Q_2 - P_2)$$
$$W_R(t) = W_P + t(W_Q - W_p)$$

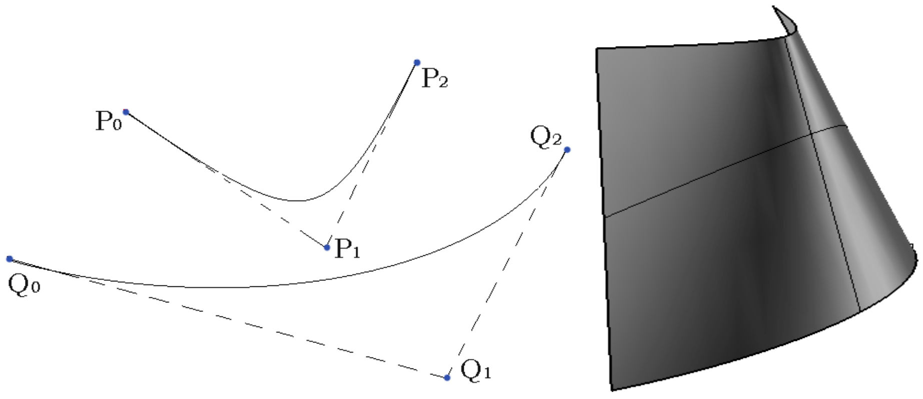


Figure 6.13 Control triangle Linear morphing

The surface constructed by this technique is equivalent to a Ruled Surface. Given two curves *C₁(u)* and *C₂(v)*, the surface generated by connecting line segments between corresponding points, one on each given curve is called ruled surface. More precisely, if *t* is a value in the domain *[0,1]* of both curves, a segment between *C₁(t)* and *C₂(t)* is constructed. This segment is usually referred as a ruling at *t*. As *t* moves from *0* to *1*, the ruling at *t* sweeps out a surface and this is the ruled surface defined by curves *C₁(u)* and *C₂(v)*.

Morphing of two dimensional shapes can be divided into two sub problems that have to be solved. These problems deal with vertex correspondence and vertex path. Common morphing literature is usually concerned with the vertex path problem. However both problems are equally important. A cross sectional profile consists of a set of rational quadratic Bezier curves or a set of control triangles. The vertex correspondence problem deals with the creation of a bijective mapping between the control triangles contained in the source *S*, and target *T* cross sections in a way that for each control triangle in *S* there is exactly one control triangle in T that is mapped to and vice versa. Most morphing algorithms thus enforce manually the correspondence of a few selected points or simply suppose that the problem is solved. This mapping between the two sets of control triangles may be solved by least square distance minimization. It is well known that the correspondence of several vertices

can significantly increase the quality of the morphing. Since the control point sets are ordered, vertex correspondence must maintain this order. Therefore, if S_i is mapped with T_j then S_{i+m} should be mapped with T_{j+m}. 2D morphing techniques pay special attention to the goal that all intermediate shapes are free of self-intersections because apart from some fancy special cases, a morphing sequence that contains self-intersections is considered to be unnatural transition from source to target.
Obviously, such a mapping is not always possible. For cases where two non-similar adjacent profiles consist of a different number of curves, some curves on one profile may not be mapped to any curves on the other profile. Figure 6.14 shows two adjacent non-similar profiles with different number of curves. In the case that two cross sections S, T do not have the same number of curves an extra processing on the set of curves is required. This extra processing involves curve splitting or curve concatenation resulting in two cross sections with the same number of curves.

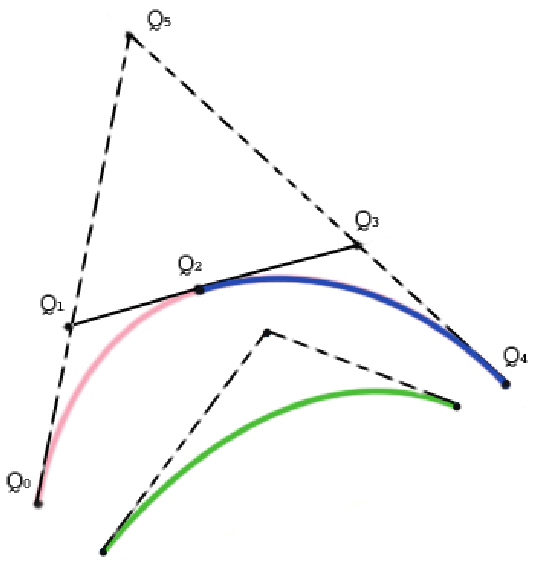


Figure 6.14 Curve Concatenation

To succeed curve concatenation of two rational quadratic Bezier curves $R_1(t)$ and $R_2(t)$ with control points Q_0, Q_1, Q_2 and Q_2, Q_3, Q_4 respectively, we must have $G2$ continuity. That means that the tangent to $R_1(t)$ line segment Q_1Q_2 must be collinear with the tangent to $R_2(t)$ line segment Q_2Q_3. The middle curve weight of each curve must also be equal. The resulted Bezier curve will also be a rational quadratic Bezier curve with control points Q_0, Q_5, Q_4. Q_5 is determined by the intersection of the lines

defined by the line segments Q_0Q_1 and Q_3Q_4 (see figure Figure 6.14). The middle curve weight of the new curve is equal to the original curves $R_1(t)$ and $R_2(t)$ weight.
G2 continuity may not be possible always. For these cases, we choose to split the curve by breaking it at a given point u_0 and create two new Bezier curves that join on u_0. An algorithm for this task was presented by de Casteljau [20], and it uses a geometric construction technique.
The vertex path problem deals with the selection of a path that a control point will travel from the source cross section to its mapped control point in the target cross section. Ruled surfaces use line segments to connect the mapped curve points.

6.5.2. Polygon-based Morphing and Interpolation

The task of surface reconstruction deals with the creation of a ribbon between two adjacent cross sections. This may be accomplished by performing triangulation between the sampled sets of vertices that belong to a pair of adjacent cross sections. In most real cases the material of interest lies in the region that separates the adjacent contours.
A rather simple solution that forces a connection of each vertex of a section with some vertices of the adjacent sections was proposed by the literature in the past. However, as the distance between two cross sections may vary, the chance of missing important information of the places where ramifications occur is rather high. As a result, the reconstructed object does not have the correct shape. To overcome this problem, we propose a method that automatically creates intermediate sections.
The projection of the region, which separates the adjacent cross sections, on an intermediate parallel plane is the region that is not common to both contours. We will denote a cross section as a binary image where the two values represent the background and the object. This intermediate plane projection may be expressed as an *exclusive OR (XOR)* operation on the binary images of the two contours [16]. In the case where the contours of the adjacent sections intercept, it is required to include the pixels of the contour boundary where the interception occurs.

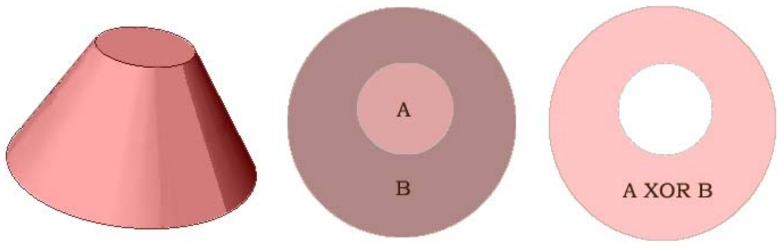


Figure 6.15 *XOR* operation on sections A and B

The result of the *XOR* operation is also a binary image whose boundary is formed by the contours of the contiguous sections. Figure 6.16 shows that the outer border of the binary image is formed by the second contour while the inner border is formed by the first contour. Figure 6.17 shows two slices that their boundaries intersect. The *XOR* operation result is shown in pink while the result of the region thinning is shown by the curves in the pink regions.

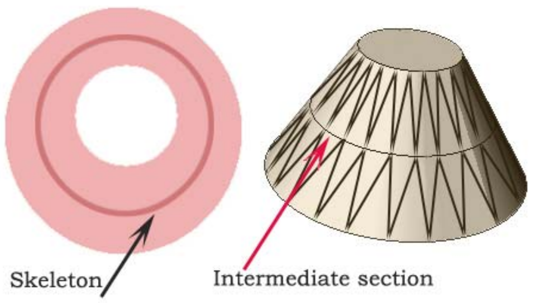


Figure 6.16 Thinning and Ribbon Construction

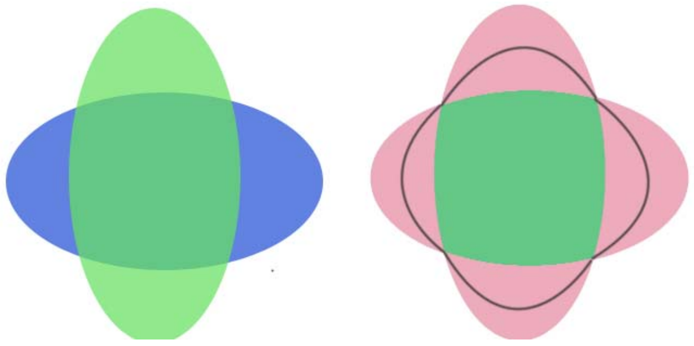


Figure 6.17 Inner and outer boundaries intersecting. *XOR* region in pink

In many cases we may see portions of the binary image to have both inner and outer borders formed by the same contour. This is an indication that in the particular portion of the material of interest there is a ramification. For these cases the skeleton of that portion of the binary image may be used to represent the place where the ramification occurs at an intermediate height of the analyzed sections.
Applying a thinning algorithm on the binary image we may obtain its skeleton (Figure 6.15, Figure 6.16, Figure 6.17). Using the shortest diagonal algorithm [25] we are able to create two ribbons (one with each slice).

6.6. Editing the Reconstructed Model

Reverse engineering is a systematic approach for capturing and analyzing the design of existing objects. One may use it either to study the design, or as an initial step to redesign the object. Ideally, a reversed engineered object should exhibit the same geometric properties that are present in the original design. Most of developed methods in literature create a very detailed and exact description of the physical object's shape suitable for creating an exact copy without providing means for editing. Primary aim of this work is to create a novel reverse engineering CAD model that would represent both the shape of the original object and the design intent.
The phase of constraint definition captures the design intent of the physical object by fitting local and global geometric regularities and symmetries. On completion of the reconstruction process, the user is given the chance to make modifications on the reconstructed CAD model by modifying the imposed constraints.
An object could contain parts that may or may not be dependent of each other. This property is specified by the inter-cross section constraints that have already been imposed. For instance, the screwdriver object has two parts: the handle and the steel shaft. Even though these two parts must be aligned on the same axis, there are no other defined inter-cross-section constraints that relate any of the handle's local properties to any of the steel shaft's. Therefore, an increase of the handle's cavity depth would not affect the shape of the steel shaft.
Definition: The editing area is defined to be the area of the reconstructed model where all cross sections would be re-evaluated based on the defined constraints. The editing area is always specified by two cross sections that bound it.

Depending on the defined inter-cross section constraints, the editing area may be extracted automatically.

Definition: A cross section, which lies inside the editing area, used for the modification of the editing area is called revision slice.

Any user modification (curve control point locations, weight values, maximum heights) may be performed on the revision slice. Modification could be performed also on the inter-cross section constraints. These changes will be propagated to all slices in the editing area defining the new shape of the model according to the imposed intra and inter -cross section constraints. For instance, when two different curves on the revision slice are constrained to have equal maximum height, modifying the maximum height of one curve implies the same modification for the second curve. All such implications, performed during the re-evaluation of the editing area based on revision slice, will affect the geometry of the object part that is inbound the editing area.

The re-evaluation process of all slices in the editing area depends on the intra and inter-cross section constraints that have been defined. After re-evaluation, all constraints must hold unconditionally. There are cases though where Let K_1 be the point on a curve in the revision slice where maximum height occurs and L_1 be the point at maximum height of the modified curve. Also, let K_2 be the point on the respective curve of another slice B where maximum height occurs (Figure 6.18). Our aim is to compute the modified curve of slice B according to the modification in revision slice. To determine the new curve we only need to compute the point of maximum height L_2 using a linear interpolation that involves all three points K_1, K_2, and L_1:

$$L_2 = K_2 + \frac{(L_1 - K_1)K_2}{K_1}$$

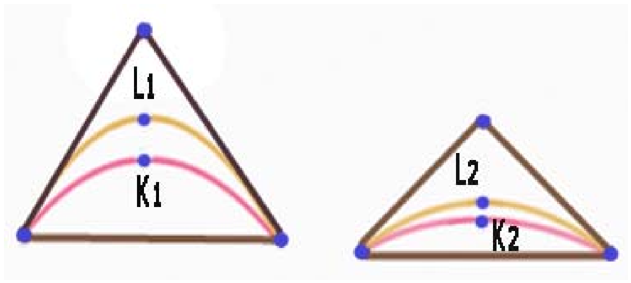


Figure 6.18: (left): Revision slice, (right) Some slice in the editing area

Another approach for the editing area re-evaluating based on the revision slice modifications would involve some geometrical computations: Figure 6.19a shows part of the revision slice. The maximum height of the curve is at point *K* and the modified point is *L* on the line segment MP_1.

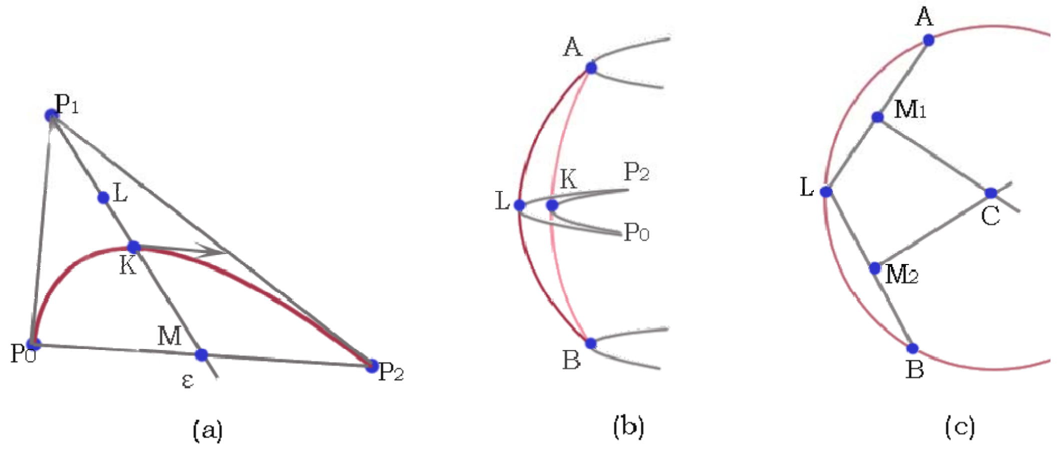


Figure 6.19: (a) Revision slice, (b) Editing area in 3D, (c) Points A, L, B belong to the same circular arc

Figure 6.19b, shows the editing area and revised slice in 3D illustrating how the increase of the curve's maximum height affects the depth of the cavity. Using points *A, B, L* a circular arc is defined for which we may compute the circle equation (center, radius). The perpendicular bisectors (Figure 6.19c) of *AL* and *LB* intersect at point *C* which is the center of the circle while |*CA*|=|*CB*|=|*CL*| is its radius. Then,

$$M_1 = (\frac{x_A + x_L}{2}, \frac{y_A + y_L}{2})$$

$$M_2 = (\frac{x_L + x_B}{2}, \frac{y_L + y_B}{2})$$

The lines connecting *A* and *L, L* and *B* are given by

$$(y - y_A)\cdot(x_L - x_A) = (x - x_A)\cdot(y_L - y_A),$$
$$(y - y_L)\cdot(x_B - x_L) = (x - x_L)\cdot(y_B - y_L)$$

Therefore,

$$Y = \frac{y_L - y_A}{x_L - x_A} X + \frac{y_A x_L - y_L x_A}{x_L - x_A} \qquad slope: \frac{y_L - y_A}{x_L - x_A},$$

$$Y = \frac{y_B - y_L}{x_B - x_L} X + \frac{y_L x_B - y_B x_L}{x_B - x_L} \qquad slope: \frac{y_B - y_L}{x_B - x_L},$$

A line perpendicular to another line has negative reciprocal slope. Consequently, the lines perpendicular to line segments *AL* and LB have slopes:

$$m_1 = \frac{x_A - x_L}{y_L - y_A}, \quad m_2 = \frac{x_L - x_B}{y_B - y_L}$$

Determining the perpendicular line equations we have:

$$Y = \frac{y_A + y_L}{2} + \frac{x_A + x_L}{y_L - y_A}(X - \frac{x_A + x_L}{2}),$$

$$Y = \frac{y_L + y_B}{2} + \frac{x_L + x_B}{y_B - y_L}(X - \frac{x_L + x_B}{2}),$$

The center *C* of the circular arc *AB* is given by the intersection of the above two lines. Solving the set of linear equations we may determine the coordinates of point *C*. Then, for each cross section in the editing area, the point that the rational Bezier curve must pass through is the intersection of the slice plane and the circular arc *AB* (L_1, L_2, L_3, …, L_n).

To determine the weight of each rational Bezier curve that passes through the point L_i we may use the following formula [16]

$$w_1 = \frac{\tau_1}{2\sqrt{\tau_0 \tau_2}}$$

where τ_0, τ_1, τ_2 are the barycentric coordinates of L_i with respect to the triangle formed by the three control points of each rational Bezier P_0, P_1, P_2 which are determined as follows:

$$\tau_0 = \frac{area(L_i, P_1, P_2)}{area(P_0, P_1, P_2)}, \quad \tau_1 = \frac{area(P_0, L_i, P_2)}{area(P_0, P_1, P_2)}, \quad \tau_2 = \frac{area(P_0, P_1, L_i)}{area(P_0, P_1, P_2)}$$

$$and \quad area(a,b,c) = \frac{1}{2}\begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{vmatrix}$$

CHAPTER 7. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

7.1. Implementation Issues

We have implemented and tested a prototype of the proposed method using the

- MS Visual C++ programming language,
- the OpenGL graphics libraries,
- the IpOpt optimization software [76].
- the ACIS solid modeling libraries by Spatial Corporation [75].

The entire system was built using the object oriented design framework. We have used extensive testing with several cloud point sets. For the internal representation of the contours (sequences of G^S rational Bezier patches) we have used NURBS.

Besides the improvements that we have to make to some of the algorithms used in the system for faster execution and better memory management our future aim is to enhance this prototype system to a full function feature based CAD software that would provide visual tools for constraint definition and solving, automatic selection of point cloud slicing direction, splitting and concatenation of slice curves, feature detection during reconstruction, and user editing of the final model by modifying slice entities (curves, control points, weights, max heights, constraints) or even entire features. Major tasks necessary to be tackled in the future is the minimization of user interaction and the enhancement of the already user friendly GUI.

7.2. Experimental Evaluation

We evaluated our methodology framework by testing the implemented prototype with many test point clouds obtained by a 3D laser scanner. In general, our proposed method can reverse engineer and redesign any mechanical or free form object. As we mentioned earlier, the three methods [77], [2], [81], that try to tackle similar problems, have important differences in the aspects of approaching the problem. As a consequence, there are no well-accepted criteria to compare the quality of these CAD models quantitatively, and therefore we do not intend to claim that our results are necessarily better.

Our method may be insufficient for a complex object where different parts of the object may be optimally sliced in different slicing directions. For such cases the object may be decomposed into parts using advanced segmentation techniques.

We intentionally selected point clouds that cannot be trivially reverse engineered constrained and redesigned. Following, we will run through a detailed example to demonstrate the method's effectiveness. We have used a 3D point cloud of a screwdriver object containing *27500* points which was then sliced to equidistant parallel cross sections (Figure 7.1).

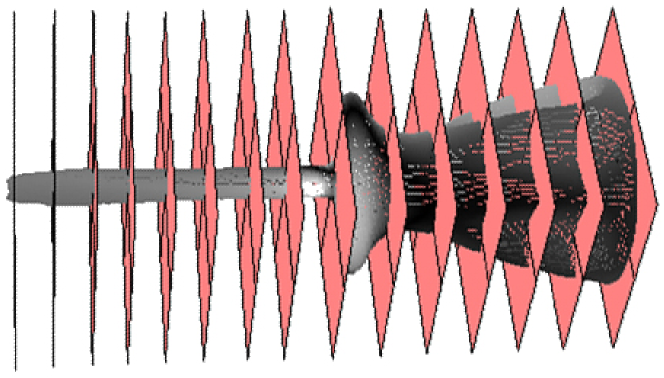


Figure 7.1: Slicing the screwdriver point cloud

Figure 7.2 shows part from a cross section of the screwdriver's handle containing *437* points. Thinning and quantization of this cross section results in a point set with *323* points that form a *1*-point-thick curve boundary.

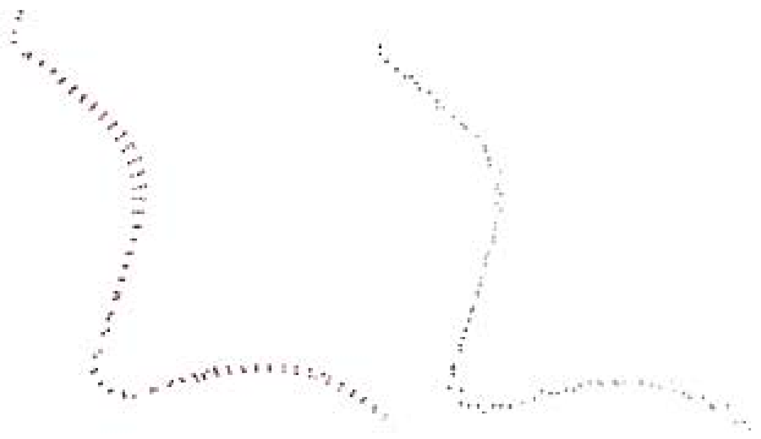


Figure 7.2 Cross Section Thinning

While partitioning the thin slice point set, the algorithm filters out all noisy points (Figure 7.3). The final result of the concavity detection process is a number of ordered subsets of consecutive points that each one may be approximated by a single quadratic rational Bezier curve. The first and last point in each ordered subset will serve as start and end point respectively for the induced curves.

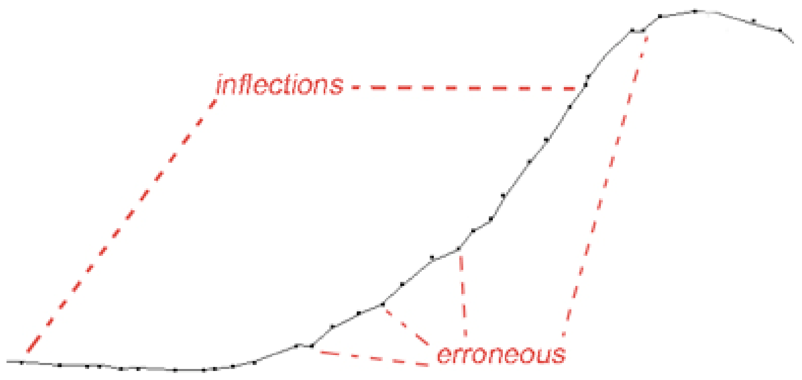


Figure 7.3 Concavity change detection

For the example cross section of Figure 7.2, the method processed 323 points and detected 13 ordered subsets of points. Figure 7.4 shows these subsets in black color and the respective inflection points in blue color. The original point set forms a six peak star shape which is symmetrical. One may notice that the detected inflection points are positioned symmetrically. Ideally, we would expect to have 12 ordered subsets of points divided by 12 symmetrically placed inflection points (*i-th* subset end point coincides with *(i+1)-th* start point). Instead, the method detected 13 subsets of points. Figure 7.4(right), shows that there is one point subset that may be approximated by a line segment. This happens many times on real world data sets because the initial model may suffer from inaccuracies caused by sensing errors propagated from the data acquisition phase, or due to approximation and numerical errors arising from the successive algorithmic steps, or even flaws on the surface of the original object.

Figure 7.4 shows the computation of the middle control point of all quadratic rational Bezier curves that we are going to construct. The left figure shows a slice from the bottom part of the steel shaft while the right figure shows a slice from the screwdriver's handle.

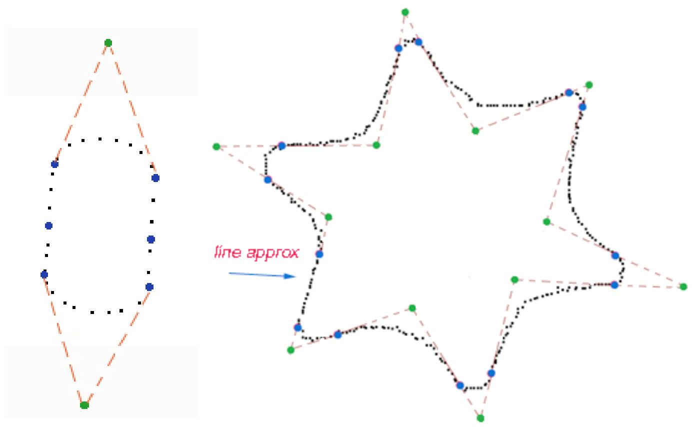


Figure 7.4 Control Point derivation: (a): slice on steel shaft,(b) slice on handle

Depending on the point topology either method may be used to determine the middle control point. The method selection criterion should be the minimum squared distance of the point set from the fitted curve. Figure 7.5 shows a data set with seven points and the curves that are fitted on them by both methods. The minimum squared distance for the first method (green color) is *0.004091* and *0.007578* for the second method (red color). It is clear that for the particular data set the first method produces a better fitted curve.

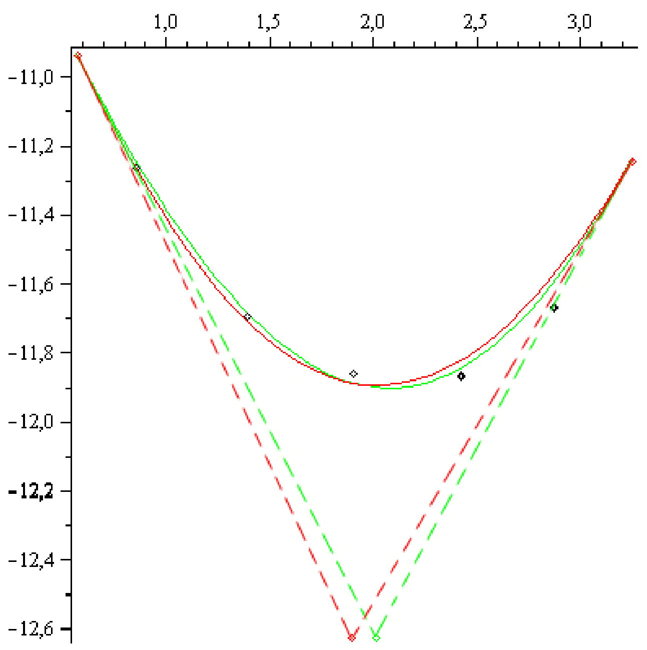


Figure 7.5 Best method for Control point selection through distance minimization

Following, for each partition of points, the computation of the middle control point weight is performed by minimizing the sum of squared distances of all points from the fitted curve (Figure 7.5, Figure 7.6) using the IpOpt libraries. Figure 7.7 shows the set of curves built by the algorithm.

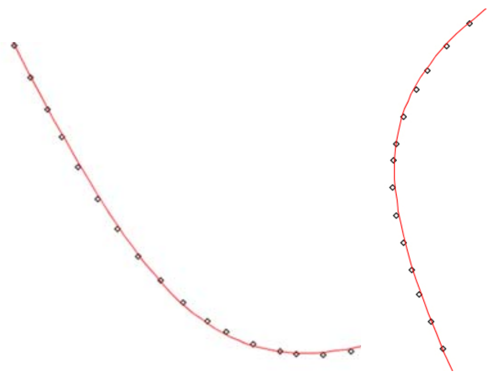


Figure 7.6: Middle weight adjustment minimizes point distances from the curve

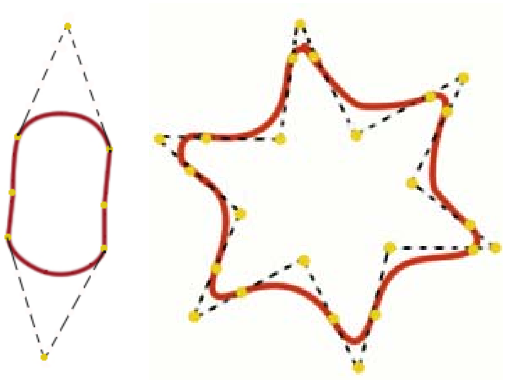


Figure 7.7 Fitting Rational Bezier Curves.

The following diagrams evaluate the effectiveness of the fitting method. We compare *6* different point sets from different slices. Each point set has a different number of points to be fitted (curve1 *24*, curve2 *31*, curve3 *43*, curve4 *17*, curve5 *10*, curve6 *26*). The first diagram (Figure 7.8) shows the relation between the number of points that are to be fitted and the time that the fitting method needed to complete the task. Thus we observe that the time needed is linear on the number of points that are fitted. These experiments were conducted on an average computer system.

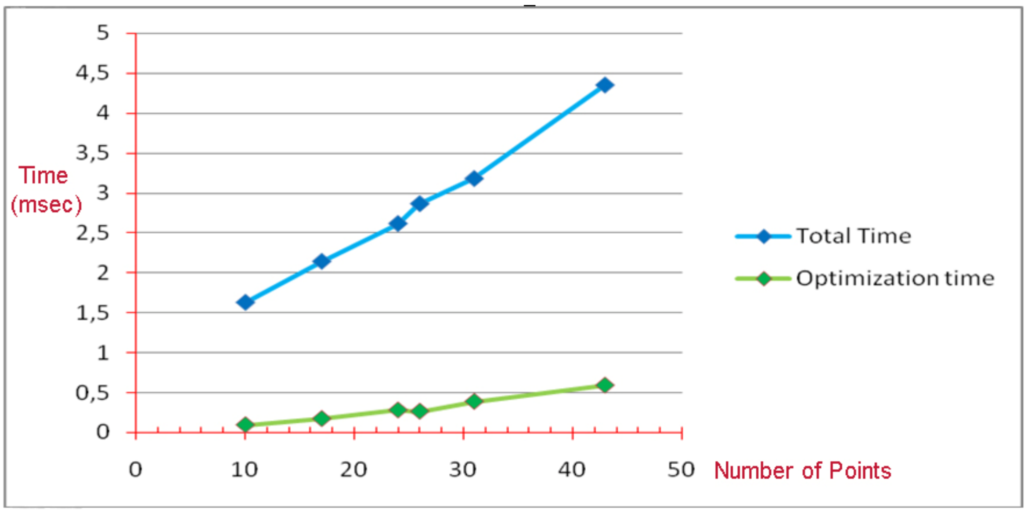


Figure 7.8 Time for fitting point sets

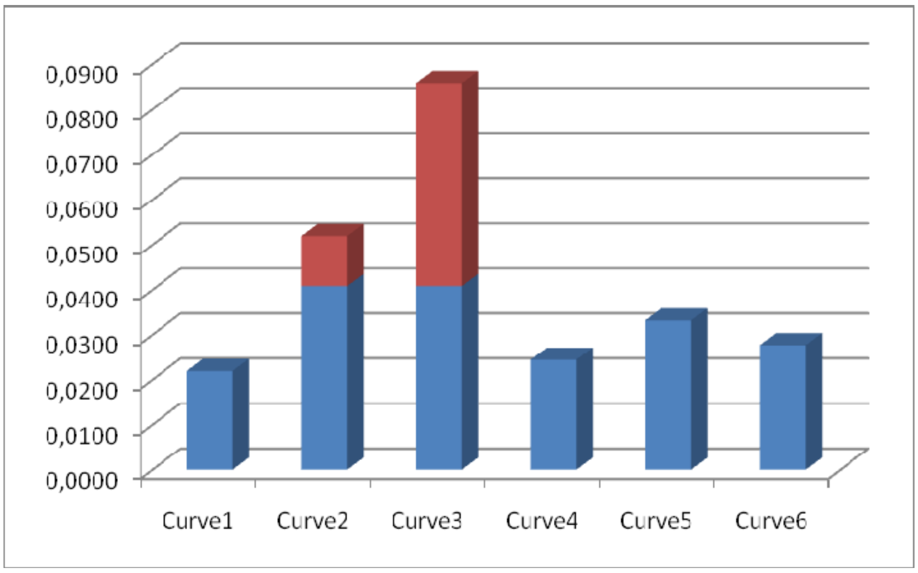


Figure 7.9: Average Error per point

The diagram in Figure 7.9 evaluates the effectiveness of the fitting method. It shows the average error of a curve point from the fitted curve. We notice that curve3 error is a lot above the average error. There are two factors that are responsible for this issue:

- The start point normal vector forms an angle greater than $\pi/2$ with the end point normal vector.
- The smoothing tolerance of the partition process was not used to split the set of points into two partitions.

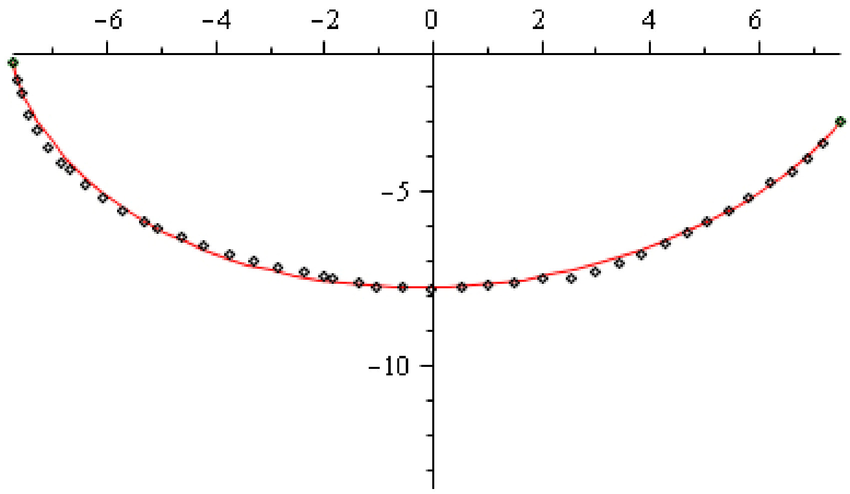


Figure 7.10: Above average fitting error

Despite the above average error, the normalized error values are fairly low even though the original point cloud was very noisy.

Geometric constraints enforcing strong relations between geometric primitives may be defined with the use of Boolean and geometric operations creating a dimension driven solid model. Intra-cross section constraints maintain symmetries and regularities among geometric primitives within a single cross section. A set of system detected intra cross section constraints defined on the two symmetrical cross sections of Figure 7.11 is summarized below:

<u>Screwdriver steel shaft:</u>

- Equality of opposite angles: $\theta_1=\theta_2$, $\theta_3=\theta_4$,
- Equality of opposite sides: $d_1=d_3$, $d_2=d_4$.
- Congruency of opposite control triangles

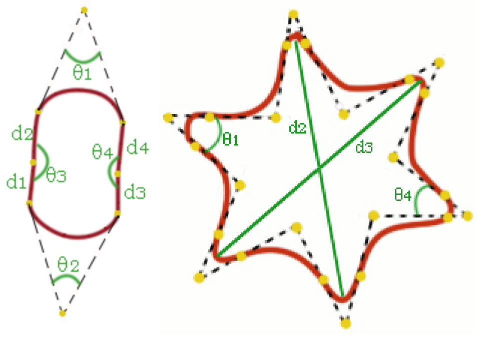


Figure 7.11 Intra-cross section Constraints.

<u>Screwdriver handle (six peak star shape):</u>

- Equality of opposite middle control point angles: $\theta_1=\theta_4$, $\theta_2=\theta_5$, $\theta_3=\theta_6$
- Equality of distances between opposite peaks (diameters): $d_1=d_2=d_3$,
- Congruency of opposite control triangles
- Equality of all cavity curve heights
- Equality of all lump curve heights

Inter-cross section constraints maintain symmetries and regularities among geometric primitives between different cross sections. Figure 7.12 a,b show two adjacent cross sections (red and green) where there is a noticeable change in the shape of their contour. Nevertheless, the respective cross section centroids (centers of mass) are aligned on the same axis perpendicular to the slice plane. The respective control points are also aligned on the same perpendicular to the slice plane axis. Therefore, the shape difference between the two cross section contours is due to the different values of curve weights (maximum height). Note that for all respective lump curves in the star shape the maximum height points are aligned on the same perpendicular to the slice plane axis (Figure 7.12c).

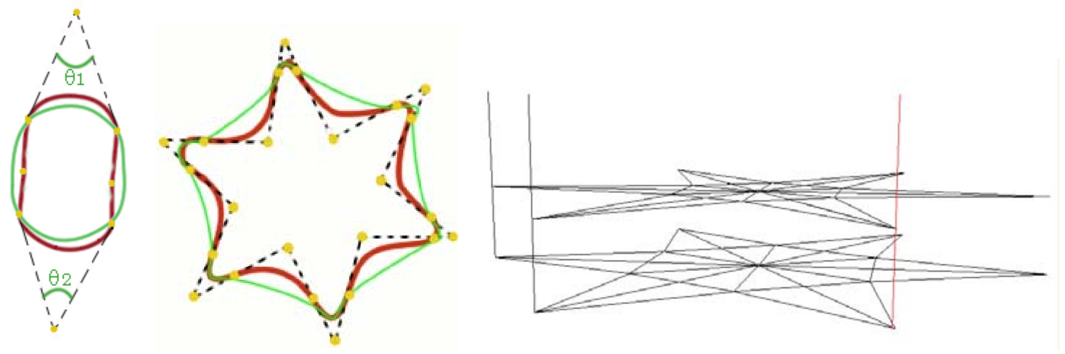


Figure 7.12 Inter-cross section Constraints: Slice 1 in red, slice 2 in green.

We will now constrain the handle of the screwdriver by user defined constraints. We will consider the editing area to be the part of the screwdriver's handle between cross section *A* and *B* (Figure 7.14). Intra Cross Section Constraints will be defined on a user selected cross section *M* where maximum cavity depth occurs.

Figure 7.13a shows the control polygon that results from the curve fitting process. We define a normal hexagon which is centered on the center of mass of the cross section. Each side of the hexagon is the base of an isosceles triangle (Figure 7.13b). All peaks *(S1, S2,.., S6)* are equidistant from the center of mass *H=14*. All hexagon sides are equal *d=7* while all cavity peaks (*E1, E2, ...,E6)* are equidistant from the center of mass *d=7*. The values *H* and *d* are completely independent of each other. In other words, the value of *H* controls the diameter of the handle while the value of *d* controls the radius of the normal hexagon, the height of the isosceles triangles and therefore, the depth of the handle cavities.

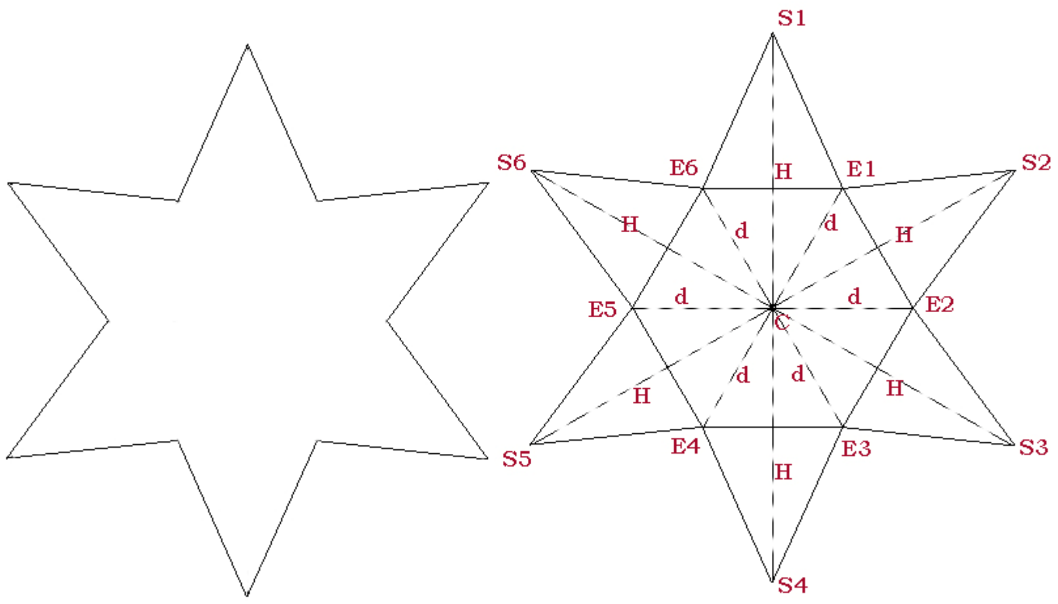


Figure 7.13 (a) Control polygon (b) Intra Constraints

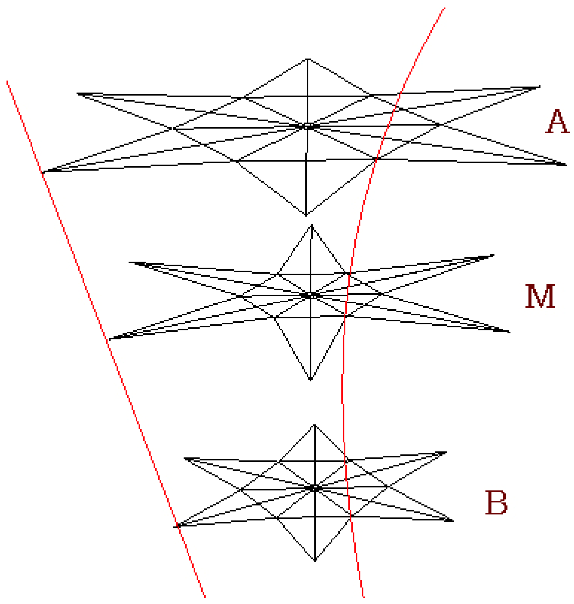


Figure 7.14 Inter Constraints

For inter cross section constraints we define that all slices between cross sections *A* and *B* must have their centers of mass on the same *z*-axis. The diameter of the handle changes linearly. Therefore, all corresponding slice peaks belong to the same line. Consequently the value *H* may be easily evaluated by the line equation. Furthermore, we determined a quadratic Bezier curve that best fits all corresponding cavity peaks among different slices. Therefore, the value d (radius of normal hexagon) may be easily evaluated using the Bezier curve (Figure 7.14).

The resampling step computes the length of each rational Bezier segment in the slice. The approximate length of the entire contour is $\Lambda=95.76$. Setting $\mu=60$, we obtain the distance Λ/μ of each point from its neighbors to be around 1.596. Figure 7.15 shows the set of representative points that were selected.

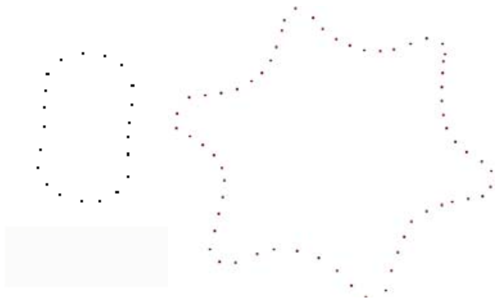


Figure 7.15 Resampling result.

Figure 7.16 shows the intermediate slice generation using the *XOR* operation. Figure 7.17 shows the result of the intermediate slice generation by curve morphing. The reconstructed part of the object between the two adjacent cross sections is shown in Figure 7.18.

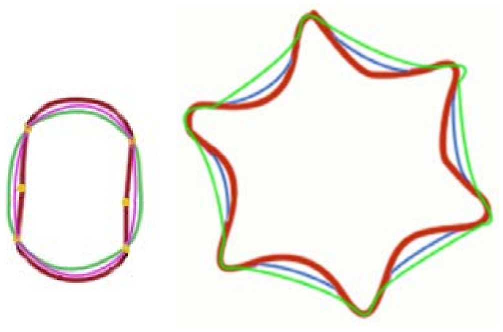


Figure 7.16 Auto slice generation by *XOR*. Slice 1 (red), slice 2 (green), slice S_1 *XOR* S_2 (purple)

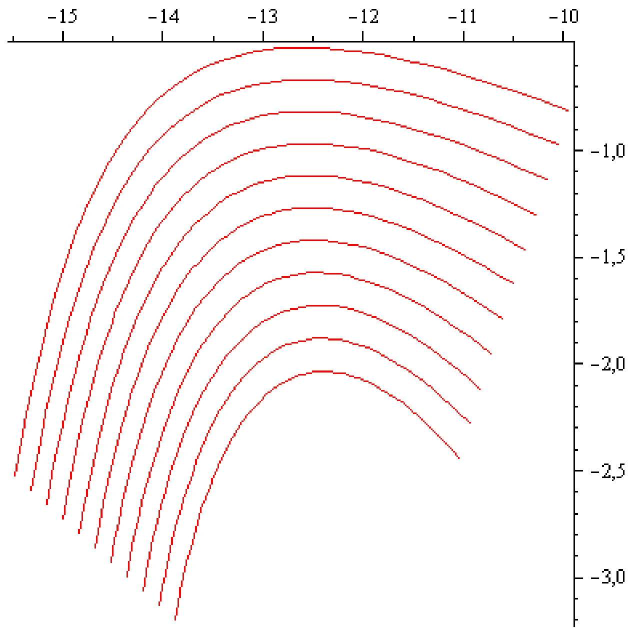


Figure 7.17 Auto slice generation by curve morphing

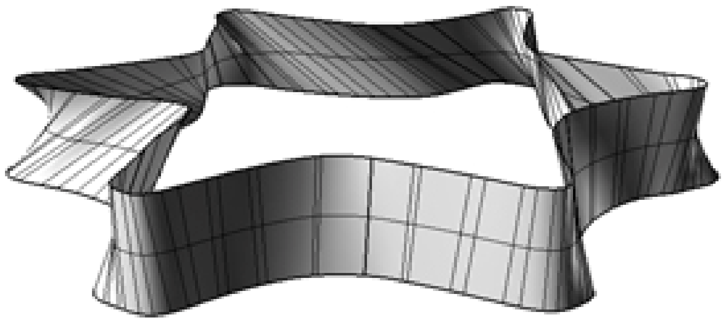


Figure 7.18 Reconstruction of part using intermediate slice generation

Figure 7.19a shows the fully reconstructed object (exact copy). Figure 7.19b shows the modified reconstructed object with the cylindrical part of the steel shaft longer. To accomplish this modification we increased the distance between the cross sections on the steel shaft by a factor of *1.4*. Figure 7.19c shows the modified reconstructed object with the lower part of its handle wider. The modification actually made was an increase of the diameter of the lower handle by a factor of *1.3*.

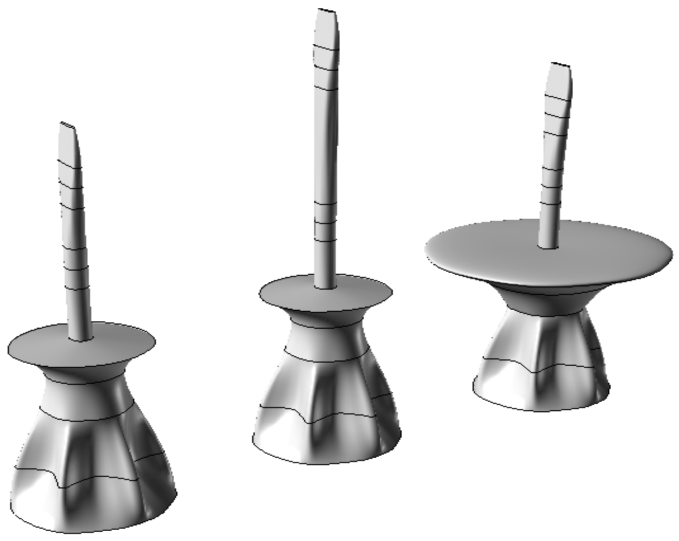


Figure 7.19 (a) Reconstruction result (b,c) Reconstruction result after Editing.

Figure 7.20 illustrates a constraint modification in the cavities of the screwdriver handle. The designer decreased the value of *d* by *25%* in the revision slice. This decrease propagated to all slices in the editing area automatically by the reevaluation of the Bezier curve that constrains the value of d in the editing area. As a result, the depth of all cavities in the screwdriver handle were increased accordingly. Notice that the other constraint value *H* remained constant (line equation did not change) and therefore, the diameter of the handle does not change. The difference in the cavity depths is clearly seen in Figure 7.21. The original and the modified object may be seen in Figure 7.22 and Figure 7.23.

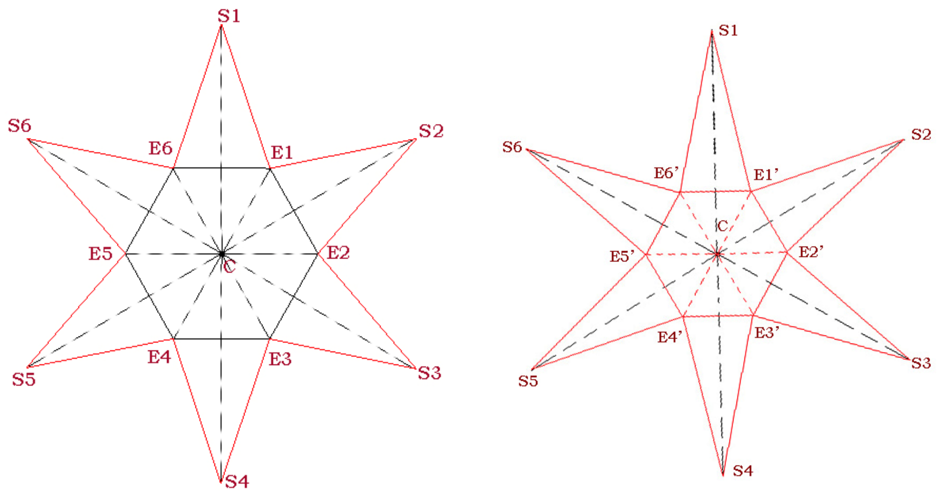


Figure 7.20 Original and Modified Slice Constraints

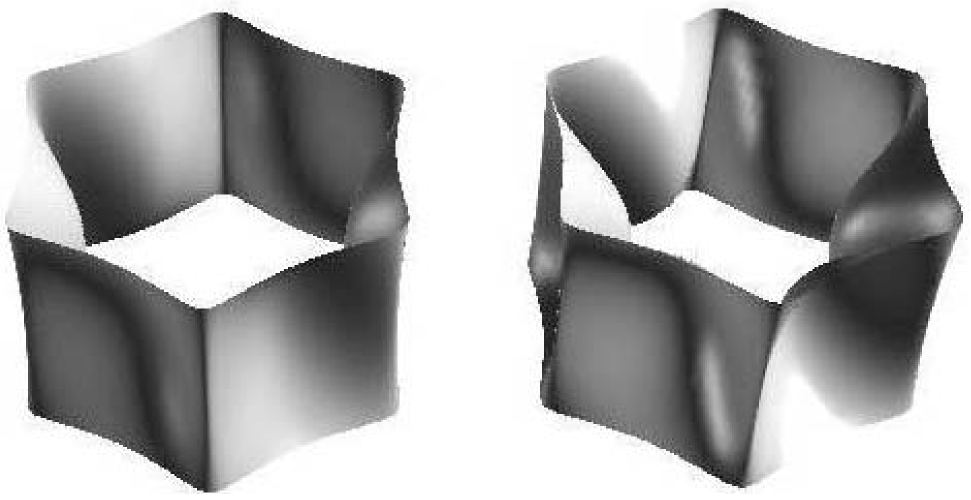


Figure 7.21 Original and Modified object part

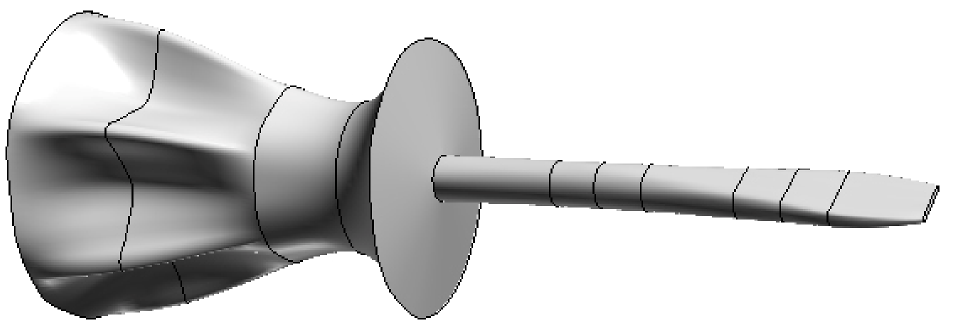


Figure 7.22 Original Object

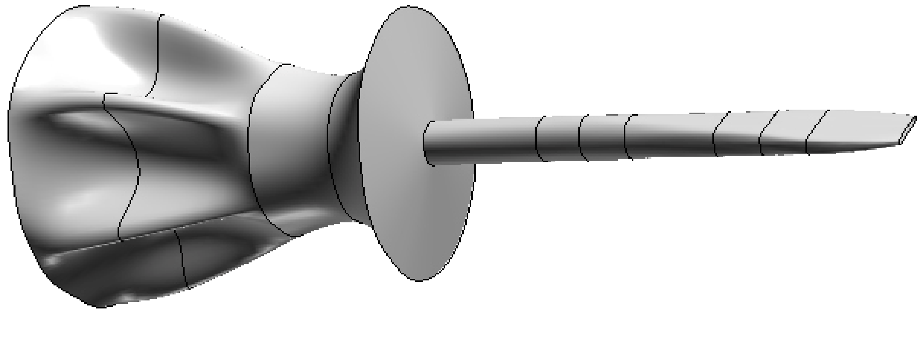


Figure 7.23 Modified object

CHAPTER 8. CONCLUSIONS

We have presented an effective and efficient method to build a 3D CAD model from a given point cloud representing the surface of an object.
Our approach to re-engineering uses point cloud slices along a principal axis. These slices are then processed to obtain a thinned, ordered set of planar points. Subsequently, this set is used to obtain a fully functional cross section represented by a number of constrained rational Bezier curves.

We have introduced inter-cross-section and intra-cross-section geometric constraints for supporting editability.

3D contour-based reconstruction has been extensively studied, and we have employed and tested several slice morphing and slice insertion techniques for covering between non similar adjacent cross sections. Model editability is also supported at this level by defining parameters for the 3D reconstruction of user defined slice groups.

We have performed a preliminary evaluation of the usability of our method with very good results even for users with no former CAD software experience. Our method provides the tools for robust and accurate editing of the produced CAD model prior to remanufacturing.

Automated detection of an optimal slicing direction is an addition that can save users a lot of effort. Finally, the effectiveness of the reconstruction process could be improved for complicated objects by first decomposing the object by employing sophisticated decomposition methods such as the one presented in [47].

REFERENCES

[1] Anderl R., Mendgen R.: Modelling with constraints: theoretical foundation and application, Artificial Intelligence in Computer-Aided Design, 1996, 28(3), 155-168.

[2] Au C.K., Yuen M.M.F.: Feature-Based Reverse Engineering of Mannequin for Garment Design, Computer-Aided Design, 1999, 31, 751-759.

[3] Barequet G., Goodrich M.T, Levi-Steiner A., Steiner D.: Contour Interpolation by straight skeletons, Graphical Models, 66 (2004), 245-260.

[4] Barequet G., Shapiro D., Tal A.: Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices, The Visual Computer, 16(2) (2000) 116–133.

[5] Barequet G., Sharir M.: Piecewise-linear interpolation between polygonal slices, Computer Vision and Image Understanding, 63 (1996), 251-272.

[6] Bajaj C.L., Coyle E.J., Lin K.N.: Arbitrary topology shape reconstruction from planar cross sections, Graphical Models and Image Processing, 58 (1996) 524–543.

[7] Benko P., Martin R.R., Varady T.: Algorithms for reverse engineering boundary representation models, Computer-Aided Design 2001, 33(11), 839-851.

[8] Benko P., Kos G., Varady T., Andor L., Martin R.R.: Constrained fitting in reverse engineering, Computer-Aided Design 2002, 19(3), 173-205.

[9] Bloomenthal J.: Medial-based Vertex Deformation, Proc. of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2002, San Antonio, TX, USA.

[10] Boissonnat J.D.: Shape reconstruction from planar cross sections, Computer Vision, Graphics and Image Processing, 44 (1988).

[11] Borgefors G., Sanniti di Baja G.: Analyzing nonconvex 2D and 3D patterns, Computer Vision and Image Understanding, v.63 n.1, p.145-157, Jan. 1996.

[12] Borges C.F., Pastva T.: Total least squares fitting of Bézier and B-spline curves to ordered data, Computer Aided Geometric Design, 19 (2002), 275–289.

[13] Chambelland J.C., Daniel M., Brun J.M.: A robust iterative method devoted to pole curve fitting, In CAD/GRAPHICS 2005 conference (Hong-Kong, Chine, December 7-10, 2005), Proc. ISBN 0-7695-2473-7, pages 22–27. IEEE Computer Society, 2005.

[14] Chen X., Hoffmann C. M.: On Editability of Feature – Based Design. Computer Aided Design, 27(12):905-914, 1995.

[15] Chen X. Hoffmann C. M.: Towards Feature Attachment, Computer Aided Design, 27(9):695-702, 1995.

[16] Christiansen H., Sederberg T.: Conversion of complex contour line definitions into polygonal element mosaics, Computer Graphics, 13 (1978) 187–192.

[17] Cormen T., Leiserson C., Rivest R.: Introduction to Algorithms, McGraw-Hill, 1990. pp. 477-85.

[18] Cornea N.D., Demirci M.F., Silver D., Shokoufandeh A., Dickinson S.J., Kantor P.B.: 3D Object Retrieval using Many-to-many Matching of Curve Skeletons, IEEE International Conference on Shape Modeling and Applications (SMI) 2005, Boston USA, 368-373.

[19] Cripps R.J.: Algorithms to support point-based cadcam, International Journal of Machine Tools and Manufacture 2003, 43(4), 425-432.

[20] de Casteljau F.: Outillage Methodes Calcul, Anre Citroen Automobiles SA, Paris 1959.

[21] Dedieu J.P., Favardin Ch.: Algorithms for ordering unorganized points along parametrized curves, Numerical Algorithms 6 (1994) 169–200.

[22] De St. Germain H.J., Stark S.R., Thompson W.B., Henderson T.C.: Constraint Optimization and Feature-Based Model Construction for Reverse Engineering, in Proc. of the ARPA Image Understanding Workshop 1996.

[23] Dobson G.T., Waggenspack Jr. W.N., Lamousin H.J.: Feature based models for anatomical data fitting, Computer Aided Design 1995; 27(2):139–46.

[24] Fang L., Gossard D.C.: Fitting 3D curves to unorganized data points using deformable curves, Visual Computing (Proc. of CG International '92), Springer, Berlin, 535–543.

[25] Farin G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Boston: Academic Press, 1997.

[26] Floater M.S.: Derivatives of rational Bezier curves, Computer Aided Geometric Design 9, 161-174, 1992.

[27] Fudos I., Hoffmann C. M.: A Graph-constructive Method to Solving systems of Geometric Constraints, ACM Transactions of Graphics, Vol. 16(2), pp. 179-216.

[28] Fudos I., Hoffmann C. M.: Constraint-Based Parametric Conics for CAD, Computer Aided Design, Vol. 28, No. 2, pp. 91-100 [01 Jan 1996].

[29] Fudos I., Palios L.: An Efficient Shaped-based Approach to Image Retrieval, Discrete and Applied Mathematics, 2000.

[30] Gould N.I.M., Orban D., Sartenaer A., Toint P.L.: Superlinear convergence of primal-dual interior point algorithms for nonlinear programming, SIAM Journal on Optimization, 11, 2001, 974–1002.

[31] Hoffman D., Richards W.: Parts of recognition. Cognition, 18:65-96,1984.

[32] Hoffmann C.M., Joan-Arinyo R.: Erep An editable high-level representation for geometric design, Geometric Modeling for Product Realization, Wilson P., Wozny M., Pratt M., eds., North Holland, 1993, 129-164.

[33] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface Reconstruction from Unorganized Points, Proc. of SIGGRAPH 92, pp.71-78, 1992.

[34] Hoschek J., Lasser D.: Fundamentals of Computer Aided Geometric Design, Peters A.K., Ltd, 1993, ISBN 1-56881-007-5

[35] Jense G.J., Voxel-based methods for CAD, Computer Aided Design 1989, 21(10), 528-533.

[36] Keppel E., Approximating complex surfaces by triangulation of contour lines, IBM Journal of research and developement, 19 (1975), 2-11.

[37] Klein R., Schilling A., Straßen, W.: Reconstruction and simplification of surface from contours, Graphical Models, 62(6) (2000) 429–443.

[38] Ko H., Kim M.S., Park H.G., Kim S.W.: Face sculpturing robot with recognition capability, Computer Aided Design 1994;26(11):814–21.

[39] Kobbelt L., Botsch M.: A survey of point-based techniques in computer graphic, Computers and Graphics 2004, 28(6), 801-814.

[40] Langbein F.C., Marshall A.D., Martin R.R.: Choosing Consistent Constraints for Beautification of Reverse Engineered Geometric Models, Computer-Aided Design, 2004. 36: 261-278.

[41] Langbein F.C., Mills B.I., Marshall A.D., Martin R.R.: Finding Approximate Shape Regularities in Reverse Engineered Solid Models Bounded by Simple Surfaces. in Proc. of the 6th Symp. Solid Modeling & Applications, ACM. 2001.

[42] Lee K.: Principles of CAD /CAM / CAE Systems. Addison Wesley, 1999, ISBN 0-13-178454.

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

[44] Levin D.: Mesh-independent surface interpolation, Geometric Modeling for Scientific Visualization, Edited by Brunnett, Hamann and Mueller, Springer-Verlag, 2003, 37-49.

[45] Levin D.: Multidimensional reconstruction by set-valued approximation, IMA J. Numerical Analysis, (6) (1986) 173–184.

[46] Lien J.M., Amato N.M.: Approximate Convex Decomposition of Polygons, in Proc. 20th Annual ACM Symp. Computat. Geom (SoCG), pages 17-26, June 2004.

[47] Lien J.M., Keyser J., Amato N.M.: Simultaneous Shape Decomposition and Skeletonization, In Proc. ACM Solid and Physical Modeling Symp. (SPM), pp. 219-228, Cardiff, Wales, UK, Jun 2006.

[48] Lipschutz M.M.: Differential Geometry, Mc Graw Hill, New York 1969.

[49] Liu S., Ma W.: Seed-growing segmentation of 3D surfaces from CT-contour data, Computer-Aided Design, 31 (1999) 517–536.

[50] Ma W, He P.: B-spline surface local updating with unorganised points, Computer Aided Design 1998;30(11):853–62.

[51] Ma W., Kruth J.P.: NURBS curve and surface fitting for reverse engineering, Int J Adv Manufact Technol 1998;14:918–27.

[52] Meyers D.: Reconstruction of Surfaces from Planar Contours, PhD thesis, University of Washington, July 1994.

[53] Mills B. I., Langbein F. C., Marshall A. D., Martin R. R.: Estimate Frequencies of Geometric Regularities for Use in Reverse Engineering of Simple Mechanical Components, Technical Report GVG 2001 – 1.

[54] Oliva J.M., Perrin M., Coquillart S.: 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram, Comp. Graphics Forum, 15(3) (1996) C397–C408.

[55] Park H.: Choosing nodes and knots in closed B-spline curve interpolation to a point data, Computer-Aided Design 2001;33(13):967–74.

[56] Parker J.R.: Algorithms for Image Processing and Computer Vision, Wiley, 1996, ISBN-10 0471140562.

[57] Parker J.R., Jennings C.: Defining the Digital Skeleton, in Proc. of SPIE Vision Geometry, vol. 1832 (Boston, Mass., 15-16 Nov. 1992).

[58] Parker J.R., Jennings C., Molaro D.: A Force Based Thinning Strategy with Sub-Pixel Precision, in Proc. of Vision Interface 94 (Banff, AB, 18-20 May 1994).

[59] Pina J., Alquezar R.: Reconstruction of Surfaces from Cross Sections Using Skeleton Information, CIARP 2003, LNCS 2905, pp. 188−195, 2003.

[60] Pina J., Alquézar R.: A new method to solve the branching problem in surfaces of three dimensional models reconstructed from parallel cross sections, Revista Cubana de Investigación Operacional, Universidad de La Habana, Cuba.

[61] Randrianarivony M.: Arc Length of Rational Bezier Curves and Use for CAD Reparametrization, World Academy of Science Engineering Technology 34, Oct 2008, ISSN 2070-3740.

[62] Rom H., Medioni G.: Part decomposition and description of 3d shapes, in Proc. International Conference of Pattern Recognition, pages 629-632, June 1994.

[63] Sato Y., Honda I.: Pseudodistance measures for recognition of curved objects, IEEE Trans. Pattern Anal. Machine Intell. PAMI-5, 4 (July), 362-373.

[64] Sato Y., Kitagawa H., Fujita H.: Shape measurement of curved objects using multiple slit ray projections, IEEE Trans Pattern Anal. Machine Intell. PAMI-4,6 (Nov), 641-646.

[65] Shanno D.F., Vanderbei R.J.: Interior-point methods for non convex nonlinear programming: orderings and higher-order methods, Mathematical Programming B, 87, 2000, 303–316.

[66] Sheehy D., Armstrong C., Robinson D.: Shape description by medial axis construction, IEEE Transactions on Visualization and Computer Graphics 1996, 2(1), 62-72.

[67] Shin H., Park S., Park E., Choi B.: Direct Slicing of a Point Set Model for Rapid Prototyping, Computer-Aided Design and Applications, Vol 1, 2004.

[68] Sloan K., Hrechanyk L.: Surface Reconstruction from Sparse Data, Proc IEEE conf on Pattern Recognition and Image Processing, 1981.

[69] Solano L., Brunet P.: A system for constructive constraint – based modeling. in Falcidieno B. and Kunii T., editors, Modelling in Computer Graphics, pages 61-84, Springer Verlag, 1993.

[70] Stamati V., Fudos I. : Constraint-based and Feature-based CAD Systems and Applications - Computer-Aided Design and other Computing Research Developments , De Smet C. M. and Peeters J. A. eds., Nova Publishers, N.Y., ISBN: 978-1-60456-860-8 [12 Dec 2008] [Book Chapter].

[71] Stamati V.: Reconstructing feature-based CAD models based on point cloud morphology, Dept. of Computer Science, University of Ioannina [Oct 2008] [PhD Thesis].

[72] Storti D.W., Turkiyyah G.M., Ganter M.A., Lim C.T., Stal D.M.: Skeleton-based modeling operations on solids, Solid Modeling '97 1997, Atlanta GA USA.

[73] Surazhsky, T., Surazhsky, V., Barequet, G., Tal, A.: Blending polygonal shapes with different topologies, Computers & Graphics, 25 (2001) 29–39.

[74] Taubin G., Ronfard R.: Implicit simplicial models for adaptive curve reconstruction, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18 (1996) 321–325.

[75] The 3D ACIS Modeler. ACIS Corporation, HTTP://WWW.SPATIAL.COM.

[76] The Ipopt - Interior Point Optimizer Project, HTTPS://PROJECTS.COIN-OR.ORG/IPOPT

[77] Thompson W.B., De St. Germain H. J., Henderson T.C., Owen J.C.: Constructing High-Precision Geometric Models from Sensed Position Data, in Proceedings 1996 ARPA Image Understanding Workshop. 1996.

[78] Thompson W.B., Owen J.C., De St. Germain H.J., Stark S.R., Henderson T.C.: Feature-Based Reverse Engineering of Mechanical Parts, IEEE Transactions on Robotics and Automation, 1999. 15(1): 57-66.

[79] Tiller W.: Rational B-splines for curve and surface representation, IEEE Comput Graph Appln 1983;3(6):61–9.

[80] Vanderbei R.J., Shanno D.F.: An interior-point algorithm for non convex nonlinear programming, Computational Optimization and Applications, 13, 1999, 231–252.

[81] Varady T.; Facello, M.; Terek, Z.: Automatic Extraction of Surface Structures in Digital Shape Reconstruction, Computer Aided Design, 39, 2007, 379-388.

[82] Varady T., Martin R. R., Cox J.: Reverse Engineering of Geometric Models—An Introduction, Computer Aided Design 29 (4), 255-268, 1997.

[83] Wang W., Pottmann H., Liu Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization, ACM Trans. Graph. 25, 2 (Apr. 2006), 214-238.

[84] Watt A.: 3D Computer Graphics, Addison Wesley, Third Edition, 2000, ISBN 0-201-39855-9.

[85] Werghi N., Fisher R., Robertson C., Ashbrook A.: Object Reconstruction by Incorporating Geometric Constraints in Reverse Engineering, Computer – Aided Design, 31(6): 363-399,1999.

[86] Yoshizawa S., Belyaev A.G., Seidel H.P.: Free-form skeleton-driven mesh deformation, Proc. of the eighth ACM Symposium on Solid Modeling and applications 2003, Seattle, Washington U.S.A., 247-253

[87] Fu Y., Zhou B.: Direct sampling on surfaces for high quality remeshing. Computer Aided Geometric Design , Volume 26 Issue 6, 711-723, August 2009.

[88] Yan X., Gu P.: A review of rapid prototyping technologies and systems, Computer-Aided Des. 28 (4) (1996) 307–318.

[89] Zhang Y.F., Wong Y.S., Loh H.T., Wu Y.F.: An adaptive slicing approach to modelling cloud data for rapid prototyping, Journal of Materials Processing Technology 140 (2003), 105–109.

PUBLISHED WORK

[1] Protopsaltis A., Fudos I.: A Feature-Based Approach to Re-engineering CAD Models from Cross Sections, Computer Aided Design and Applications. To appear. June 2010.

[2] Protopsaltou A., Fudos I.: Creating Editable 3D CAD Models from Point cloud slices, GraVisMa 2009

[3] Fudos I., Stamati V., Protopsaltou A.:, An Approach to Geometric Constraint Solving for CAD Representations, Technical Report TR 2004-17, Computer Science Dept., University of Ioannina, 2004

SHORT VITA

Antonis Protopsaltis was born on November 30, 1971 in Veria, Greece. He received his B. Sc. in Computer Science (Software Systems option) from the dept. of Computer Science in Concordia University, Montreal Quebec, Canada in 1994 and his M. Sc in Computer Science (Software Engineering) from the same University in 1996. He was twice on the Dean's Honour's list in 1991 and 1992, and earned twice the Hewlett Packard Calculator Award in 1991 and 1992 for his high achievements. In 1992 he was also awarded with a scholarship from the Hellenic Scholarships Foundation in Montreal for his outstanding performance.

He worked for a research grant in Concordia University in collaboration with Northern Telecom formally specifying in Larch/C++ the Rogue Wave commercial libraries “Tools.h++”, as System Manager in Aristotle University in Thessaloniki Greece, as Software Engineer in Pulse Microsystems in Thessaloniki Greece developing CAD software for sewing machines, and as Software Engineer in Geoanalysis in Thessaloniki Greece developing Oracle web applications.

His research interests include Parametric and Feature-Based Design, Reverse Engineering, Solid Modeling and Computer – Aided Design.