Inspection and Feature Extraction of Marine Propellers

by

Michael Oliver Jastram

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Ocean Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1996

Author .....
Department of Ocean Engineering
December 13, 1996

Certified by .....
Nicholas M. Patrikalakis
Professor of Ocean Engineering
Thesis Supervisor

Accepted by .....
J. Kim Vandiver
Chairman, Departmental Graduate Committee
Department of Ocean Engineering

Inspection and Feature Extraction of Marine Propellers

by

Michael Oliver Jastram

Submitted to the Department of Ocean Engineering
on December 13, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Ocean Engineering

Abstract

Localization Localization is the process of determining the rigid-body translations and rotations that must be performed on a set of points measured on a manufactured surface (like a propeller blade) to move those points into the closest correspondence with the ideal design surface. An additional parameter is an offset distance, such that the Euclidean motion brings the measured points as close as possible to an offset of the design surface.

An algorithm to determine the seven parameters (three rotations, three translations, one offset) was developed in 1991 by R. A. Jinkerson. But that algorithm makes some assumptions about the surface and the measured points, which are sometimes not fulfilled. Specifically, it assumes, that a measured point has always an orthogonal projection on the offset surface, regardless of the translation and rotation parameters.

This thesis extends Jinkerson's algorithm, so that these assumptions are not necessary any longer. This involves the development of a new objective function and its gradient.

Feature extraction During the manufacturing process, a propeller blade surface is subject to manufacturing inaccuracies, that result in small changes to the data describing its features. It is therefore desirable to recompute these features for comparison with the original design data. Most of the characteristics of a propeller blade are embedded in the camber lines of its hydrofoil sections. The objective of this part of the thesis is to recompute the camber line from a hydrofoil shape curve.

An algorithm for this task has already been developed, but it makes the assumption that the blade thickness has a single maximum, which is often not fulfilled, especially, if the hydrofoil has been generated from measured data.

In this thesis, a new algorithm has been developed. It generates a highly accurate camber line by using a two pass iteration method: The first pass generates an approximation of the camber line, and the second pass refines this approximation to the desired accuracy.

Thesis Supervisor: Nicholas M. Patrikalakis
Title: Professor of Ocean Engineering

Acknowledgments

This thesis is dedicated to the many people who have made it possible.

Starting back in Germany, I want to thank the staff of the "Institut für Schiffbau" in Hamburg for preparing me for survival at MIT.

During my tenure in the Master's program, several people and organizations I have been in contact with have provided tremendous support. First and foremost, my advisor, Professor N. M. Patrikalakis, whose support made it possible for me to finish the program within a short period of time.

I also got a lot of support from the whole staff of the design laboratory. Specifically I want to thank Mr. Stephen Abrams, who always was willing to take time to give technical support. Dr. Seamus Tuohy helped me to get started with the work and Dr. Takashi Maekawa accompanied me to the finishing line. Many people inspired me throughout the project, like Professor André Clément, who made useful comments and suggestions on the thesis.

Finally I want to thank the U.S. Navy for helping me cover my expenses. Funding for this work was obtained in part from the Naval Sea Systems Command of the U.S. Navy under grant number N0002496WR10553.

Contents

Abstract 2

Acknowledgements 3

Contents 4

List of Figures 7

List of Tables 9

1 Introduction and outline **10**

1.1 Localization 10

1.2 Camber line extraction 12

1.3 Outline 13

2 Literature review **14**

2.1 Localization 14

2.2 Feature extraction 16

3 General localization algorithm **18**

3.1 Introduction 18

3.2 Problem formulation 19

3.2.1 Distance function 21

3.2.2 Objective function 24

3.3 The gradient vector 24

3.3.1 Derivatives with respect to ψ, θ, ϕ 25

3.3.2 Derivatives with respect to t_x, t_y, t_z 26
3.3.3 Derivative with respect to h 27
3.4 Interpretation of the gradient vector 29
3.4.1 "Corner" point 29
3.4.2 "Border" point 30
3.5 Relation to the earlier formulation of the gradient 33
3.6 Constrained gradient function 34
3.6.1 Introduction 34
3.6.2 Selecting a constraint function 35
3.6.3 Constrained localization algorithm 36
3.6.4 Constrained function and gradient 36
3.7 Tangent plane distance 38
3.8 Conclusion 39

4 Examples of the general localization algorithm **40**
4.1 Introduction 40
4.2 Example 1: Biquadratic B-spline patch 41
4.2.1 Measured points with orthogonal projections 41
4.2.2 Measured points with non-orthogonal projections 42
4.3 Example 2: Westinghouse turbine blade 45
4.3.1 Influence of the starting position of the data points 48
4.4 Example 3: ARL propeller blade 50
4.5 Constrained localization 52
4.5.1 Examples 53
4.6 Conclusion 54

5 Camber line extraction **55**
5.1 Introduction 55
5.2 Problem formulation 56

5.3 Intersection curve 58
5.4 Properties of Brooks ribbons 58
5.4.1 Multiple Brooks ribbons define one shape 59
5.4.2 Slope of thickness function at the leading edge 60
5.5 PRAXITELES 63
5.6 Camber line generation 64
5.7 Accuracy analysis 66
5.7.1 Error dependency 67
5.7.2 Error in terms of spine curvature 67
5.7.3 Interpretation of accuracy 71
5.8 Camber line refinement 75
5.9 Approximation 76
5.9.1 Approximating the camber line 77
5.9.2 Approximating the thickness function 77
5.10 Conclusion 78

6 Examples of camber line extraction **80**
6.1 Introduction 80
6.2 ARL fan blade 81
6.2.1 PRAXITELES 8.0 (from design) 81
6.2.2 Bisector method 82
6.2.3 PRAXITELES 9.0 (From design surface) 83
6.2.4 From measured surface 86
6.3 Praxiteles logo blade 87
6.4 Conclusion 88

A Additional tables **90**

Bibliography **93**

List of Figures

3-1 Interior, border and corner points and a design surface 20
3-2 Relation between a measured point and the surface 22
3-3 Distance to tangent plane 23

4-1 The experimental patch 41
4-2 Measured points near the border of the patch 42
4-3 Westinghouse turbine blade 45
4-4 Localization results on the Westinghouse turbine blade 46
4-5 Westinghouse turbine blade, localization results from a "bad" starting position 48
4-6 Wrong distance measure 50
4-7 ARL fan blade 51

5-1 Two Brooks ribbons, generating the same shape 59
5-2 Leading edge and camber line 60
5-3 Brooks ribbon representing a hydrofoil 64
5-4 Generation of the camber line 64
5-5 Accuracy analysis 68
5-6 Error in thickness function for `NACA 24xx` 74
5-7 Error in camber position for `NACA 24xx` 74
5-8 Refinement of the camber line 75

6-1 Camber line from design generated by PRAXITELES 8.0 82

6-2 Camber line from design generated by PRAXITELES 8.0 82
6-3 Camber line generated by PRAXITELES 8.0 83
6-4 Camber line from ARL fan blade (from design surface) 83
6-5 Leading edge region of ARL fan blade 84
6-6 ARL blade thickness function 85
6-7 Camber line from ARL fan blade, using PRAXITELES 9.0 (from measured surface) 86
6-8 Camber lines at leading edge 86
6-9 Camber line from PRAXITELES logo blade (from design) 87
6-10 PRAXITELES logo blade thickness function 88

List of Tables

4.1 Localization with centered, critical data points. 43

4.2 Localization result of a set of data points with critical points 44

4.3 Localization performed on the Westinghouse turbine blade 47

4.4 Localization with measured points bad starting condition 49

4.5 Localization performed on the ARL propeller blade 52

4.6 Constrained localization on sample patch 53

A.1 The input file of the sample patch 91

A.2 A set of data points 91

A.3 Localization result of a set of measured points with orthogonal projections on the patch 92

A.4 A set of data points 92

Chapter 1

Introduction and outline

1.1 Localization

A fundamental problem in manufacturing is the determination whether a manufactured piece meets the requirements of its original design specifications. The evaluation of positional tolerances to ensure that a manufactured item is an acceptable rendering of the original design is a basic element of manufacturing inspection.

In few areas of manufacturing is the need for precise inspection and evaluation of tolerances more important than in the area of marine propellers, especially naval propellers. The manufactured item is an exceedingly complex sculptured surface which must be produced with extremely high fidelity to the original design. Very strict positional tolerances must be achieved by a difficult manufacturing process to prevent severe compromise of the propeller's performance.

The inspection of marine propellers has traditionally involved highly skilled technicians checking the surface of a manufactured propeller with numerous mechanical gauges. The blades are cut to specified dimensions under the direction of a manufacturing engineer who interprets the specifications of the propeller designer. Although rigid guidelines are provided for placement of the gauges on the blade, errors in measurement can result from inappropriate decisions of the technician or inaccurate

alignment of the gauge on the manufactured blade. Moreover, the direct gauge measurements only evaluate the blade at the local site where the measurement is made. A fully satisfactory method for evaluating global compliance with specified tolerances is not yet available. Often expensive and inefficient rework of propellers has been necessary because it was not possible to quickly and confidently ascertain whether the manufactured product would satisfy the requirements of the designer.

The recent development of automated methods of inspection using coordinate measuring machines (CMMs) and laser-based measuring devices has made it possible to obtain voluminous quantities of highly accurate spatial measurements of manufactured propellers. These robotic devices have provided a reliable source of measurement data, but methods for best using that data are still in the process of development.

The first part of this thesis will be built on existing approaches to utilize measured data from manufactured propellers for the automated inspection of those propellers. It will deal with the problem of optimally positioning a set of measured points from a manufactured propeller blade relative to the design surface from which the blade was manufactured. The first problem investigated in this thesis may be simply stated as follows:

> Given a set of measured data points from a manufactured surface, determine the rigid body translations and rotations which must be applied to the set of measured data points to bring those points into closest correspondence with the design surface from which the measured surface was manufactured.
>
> If all measured points contribute equally to the determination of the set of rigid body transformations, then the problem is one of *unconstrained localization.*
>
> If some measured points have greater effect on the determination of the set of rigid body transformations than other points, then the problem is

one of *constrained localization.*

1.2 Camber line extraction

Once a propeller blade section has been designed, it is desirable to recompute the features of the blade. Important features include camber line, section thickness function, pitch, rake, skew, chord length, maximum thickness, and leading edge. During various design steps, such as section and surface fairing, the propeller blade surface undergoes slight modifications resulting in small changes of the value of its features. It is desirable to recompute these features and compare them with design values.

Given that manufacturing always involves error, it is even more important to recompute the features of a manufactured blade. The real features of the manufactured propeller always differ slightly from the design features. Using the design surface of the blade and a set of measured points, it is possible to create a procedural surface representing the manufactured blade from which features can be extracted.

The design process of a propeller blade usually involves a set of intersection curves of the blade with a cylinder, cone or plane. Thus, to compare the manufactured blade with the design blade, it is desirable to generate the intersection curves for the manufactured blade. The intersection curve can be further analyzed, by determining leading and trailing edge and generation of the camber line. From this, other important features like pitch, rake, skew, etc. and the related features may be extracted.

Traditionally, the blade surface is defined in terms of sections with cylindrical surfaces coaxial with the propeller axis of rotation. Two-dimensional hydrofoil sections are first produced using a camber line and a thickness function. They are subsequently translated and rotated in three-dimensional space and mapped onto a cylinder, to define the surface by lofting through those sections. Thus, the first step of extracting features from a mathematical surface is the regeneration of those hydrofoil sections. Once the section is generated, the camber line and thickness function have

to be extracted to determine all other features.

The second part of this thesis deals with the problem of finding the camber line and thickness function of a given hydrofoil-like intersection curve.

1.3 Outline

The thesis is structured as follows:

Chapter 2 will present a review of current literature relevant to the problem of localization and feature extraction.

Chapter 3 will discuss the theoretical basis for the localization algorithm (both, unconstrained and constrained). It will describe the optimization problem and how it has been solved. An extension to an existing localization algorithm will be presented, which improves the reliability of the localization results.

Chapter 4 will present experimental results which demonstrate the applicability to the localization on artificial test surfaces and existing propeller blades.

Chapter 5 will present the process of generating the camber line and thickness function of a given intersection curve. The intersection curve might be generated from a design surface or from a procedural surface representing a manufactured blade and produced from measured data.

Chapter 6 will present some examples of camber lines and thickness functions generated from intersection curves from propeller blades.

Chapter 2

Literature review

2.1 Localization

There has been much interest in the problem of localization in recent years. This chapter will review the literature that is relevant to the localization problem and the work of this investigation. The intent is to provide some pertinent background information which will give the reader a broader perspective concerning this particular work.

Localization of surfaces was accomplished by Thorne et al. using a "least squares matching of associated boundary edges" [29]. Gunnarsson and Prinz developed a localization method between "a set of points and parametric surfaces" by "dynamically faceting the design surface" and finding a rigid body transformation matrix which minimizes the sum of squared distances from the data points to associated planar faces [14, 15]. This formulation required the solution of a constrained minimization problem with twelve unknowns, which are the nine elements of the rotation matrix with three translations, and six constraints which are the necessary relations between elements of the Euclidean rotation matrix.

One basic question in the formulation of the localization problem is the selection of an appropriate norm to use in distance minimization. Bourdet and Clément analyzed

this problem in [7]. The general guidelines of the paper can be summarized by saying that "for point sets of fewer than 12 points the L_∞ (or minmax) norm should be used, while the L_2 (or least-squares) norm should be used for point sets of greater than 12 points." An interesting algorithm which allows for computation of both the L_2 and the L_∞ norms at the same time is presented in [13].

The work of this thesis, as applied to the unconstrained localization problem, is based on earlier work presented by Bardis and Patrikalakis [25, 4].

This work has been continued by Jinkerson et al. [18, 19] and being implemented in the *Interrogation System* PRAXITELES[1]. To speed up the localization process, a modified Newton algorithm, provided by *Numerical Algorithms Group* (NAG) [24], is used. This algorithm uses the gradient vector of the sum of the squared oriented distances (the objective function) to find the "optimal rigid body movement of the measured points to the design surface." So the objective function has to be analytical (so the gradient vector can be provided). Jinkerson uses the "oriented distance", which is an important concept in his work. The idea was developed by Kriezis et al. [21, 22] for use in the solution of surface intersection problems. It is the oriented distance function which is defined as "the inner product between a vector from a given point to its nearest point on a surface, and the unit normal vector of the surface at that point." The derivative of the oriented distance has been developed by Jinkerson et al. [19].

Another extension incorporated by Jinkerson et al. [19] was to apply the localization on an offset of the design surface instead of the design surface itself. The offset distance is a seventh independent variable, besides the three Euler angles and the three translations. This results in much better localizations for real world problems, as this method is adapted from the manufacturing process with machining tools.

At about the same time Sahoo and Menq [28] developed a universal localization

[1]PRAXITELES is an interactive geometric modeling system for sculptured curves and surfaces. It has been developed in the Ocean Engineering Design Laboratory at MIT with funding from various U. S. government agencies [17, 30, 2]

algorithm, which extends the manufacturer's software of CMM systems. Their system processes several different types of parametric and implicit surfaces. For every surface type a distance calculation algorithm is given. In contrast to the algorithm by Jinkerson et al. [19], Sahoo and Menq do not use the analytic gradient function of the distance function.

Choi and Kurfess [11] developed, like Sahoo and Menq, a localization algorithm which works on several different surface types.

2.2 Feature extraction

The alluded hydrofoil-like curve defined as the intersection of the blade surface with a cylinder coaxial to the axis of rotation can be computed and expanded on this cylinder. The camber line of the resulting two-dimensional curve on the expanded cylinder was defined by Patrikalakis and Bardis [26] as the spine of a Brooks ribbon [9, 27]. A Brooks ribbon is generated by a planar curve called the "spine" or "directrix", and straight-line segments of varying length called the "generators" or "bones", which are "swept along the spine while being normal to the spine with their midpoints on the spine" [9]. Thus the hydrofoil section, generated by a camber line and a thickness function, is a Brooks ribbon. Other classes of ribbon-like structures are Blum ribbons [6] and Brady ribbons [8]. "While a Blum ribbon is completely determined by the boundary of a planar shape, the spine of a Brooks ribbon or Brady ribbon is not uniquely determined by the boundary of the region alone" [27]. Each Brooks ribbon also depends on the initial point chosen as its starting point.

Patrikalakis and Bardis [26] developed algorithms for the "extraction of gross geometrical features" of ideal design geometries of marine propeller blades represented by B-spline surfaces. They used an extension of a Brooks ribbon to describe the camber line, and they derived a system of nonlinear differential equations to define the camber line. The approximation of the camber line with a B-spline curve uses a

complicated error evaluation scheme.

The features extracted from the design of a propeller blade were highly accurate. Also, the surface of a manufactured propeller blade could be approximated by using measured data and the corresponding design surface. However, this surface could not be used directly for feature extraction. The problem was embedded in the integration scheme developed by Patrikalakis and Bardis [26]. The algorithm required some specific characteristics of the hydrofoil shape intersection curve from the surface (i.e. that the blade section thickness has only one maximum). Those requirements were not reliably fulfilled for intersection curves generated from measured data.

This work has been extended by Jinkerson et al. [19] and Abrams et al. [2]. Their objective was to recompute camber lines from manufactured propeller blades. From a set of measured points they generate a faceted surface, from which they extract hydrofoil shaped curves by "intersecting the faceted surface with a cylinder." For this hydrofoil, they compute the camber line. As the algorithm developed by Patrikalakis and Bardis [26] is not reliable on hydrofoil shaped curves generated from measured data, Abrams et al. [2] uses a simple marching scheme to approximate the camber line.

This marching scheme, using a bisection method, has been implemented in PRAXITELES 8.0 for the case of a hydrofoil curve generated from measured data. By using a different approach to compute the camber line from a hydrofoil section, this thesis tries to find a stable, highly accurate algorithm, which can be applied on both, design and measured data surfaces.

Chapter 3

General localization algorithm

3.1 Introduction

Localization can be defined as *the problem of determining the optimal positioning of a set of measured points on a manufactured artifact relative to its design surface.*

Unconstrained localization addresses the problem of determining the optimal positioning of a set of measured points relative to a design surface, when each measured point has an equal effect on the determination of positioning. Constrained localization, on the other hand, involves the problem of determining a feasible, but not necessarily optimal, positioning of a set of measured points relative to a design surface, when subsets of the measured points can have unequal effects on the determination of positioning.

These problems have been successfully solved [19]. The localization problem can be formulated as an optimal parameter estimation problem involving seven parameters. Those parameters are the three translations and three Euler angles, which correspond to a general three-dimensional translation and rotation of a rigid body in 3D-space. We introduce a seventh parameter, the offset distance to the design surface.

In this context the measured points represent physical points in space that are

determined by direct measurement of the surface of a manufactured part. The introduction of the offset parameter produces much better results for real manufactured surfaces like propeller blades. The reason is the manufacturing process, which uses NC milling tools. The tool moves along an offset surface to produce the design surface. If the tool is not perfectly adjusted, a small offset remains on the surface.

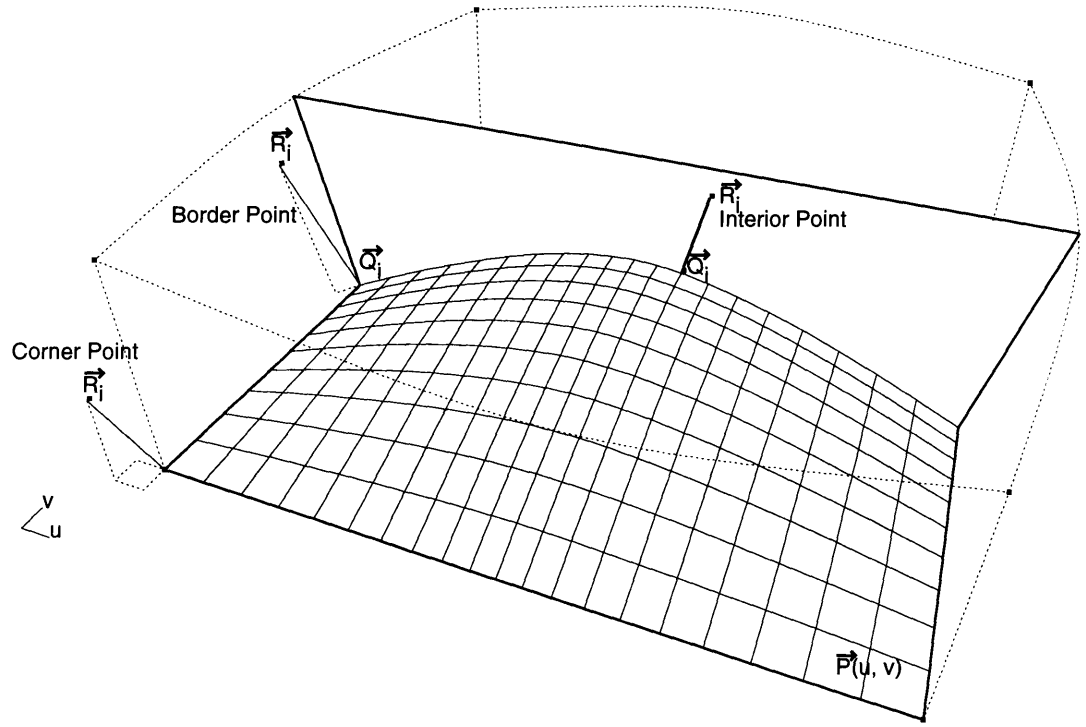
The algorithm developed by Jinkerson et al. [19] puts some requirements on the design surface and the data points, which restrict its application. The algorithm requires, that the shortest distance of a data point to the design surface is the point's orthogonal projection on the surface. Fig. 3-1 shows the three possible cases we can encounter. We call them "interior point", "border point" and "corner point", respectively. Jinkerson's algorithm requires all points to be interior points.

The localization algorithm developed by Jinkerson et al. [19] has been implemented in the interrogation system PRAXITELES 8.0[1]. The problem described above has been suppressed by ignoring points that have no orthogonal projection. This is not an appropriate solution, as we assume that a unique solution of the problem of finding the nearest position of a point on the surface exists. If the objective function changes during the localization, as a point can have an orthogonal projection in one iteration step and in the next step it does not any longer, we are not sure that we will find the optimal global solution.

3.2 Problem formulation

Consider a parametric surface $\vec{P}(u,v)$, which will be called the *design surface*, representing the desired design geometry. Consider also a set of m points $\vec{R}_i$, $1 \leq i \leq m$, $\vec{R}_i \in \Re^3$, which will be called *measured points*. Finally, consider another set of m points $\vec{Q}_i$, $1 \leq i \leq m$, on the design surface $\vec{P}(u,v)$ which are the nearest points to

[1] PRAXITELES is an interactive geometric modeling system for sculptured curves and surfaces. It has been developed in the Ocean Engineering Design Laboratory at MIT with funding from various U. S. government agencies [17, 30, 2].



The figure shows a patch with three measured points. To show the orthogonality properties better, the patch is cut by a plane, which is orthogonal to a $u = const$ isoparameter line at some value v. The dotted frame indicates the region, where points have an orthogonal projection (this region is not limited above or underneath the patch). The "interior point" illustrated has an orthogonal projection (and thus, lies in the cutting plane). The "border point" illustrated has an orthogonal projection on the $u = 0$ isoparameter line only. Finally, the "corner point" has no orthogonal projection on the patch or its four boundary curves.

Figure 3-1: Interior, border and corner points and a design surface

each measured point $\vec{R}_i$. It is assumed that the nearest points $\vec{Q}_i$ of $\vec{R}_i$ on $\vec{P}(u, v)$ are *unique*. The points $\vec{Q}_i$ will be subsequently called *footpoints*.

It can be shown that under some reasonable conditions, the nearest point will be unique [22]. Furthermore, in most cases the nearest point can be expected to be an orthogonal projection of the measured point onto the patch. Jinkerson makes this assumption in his approach [18].

3.2.1 Distance function

The objective function for minimization is the sum of squared distances of each point from the design surface. Each measured point $\vec{R}_i$ has a nearest point $\vec{Q}_i$ on the design surface. The minimum distance from each measured point to the design surface is the distance between the points $\vec{R}_i$ and $\vec{Q}_i$. If the distance between two points is denoted by $d(\vec{P}_1, \vec{P}_2)$, then the minimum distance from a measured point $\vec{R}_i$ to the design surface $\vec{P}(u, v)$ is defined as

$$d(\vec{R}_i, \vec{Q}_i) = |\vec{R}_i - \vec{Q}_i| = min_{u,v}(|\vec{R}_i - \vec{P}(u, v)|) \tag{3.1}$$

But we are not interested in the minimum distance to the design surface, but in the minimum distance to the *offset surface* with distance h to the design surface. The offset to a given footpoint $\vec{Q}_i$ is $\vec{Q}_i + \vec{n}_i h$, where $\vec{n}_i$ is the unit normal vector of $\vec{P}(u, v)$ at $\vec{Q}_i$. Thus, the minimum distance d of a measured point to the offset h of the design surface is

$$d(\vec{R}_i, \vec{Q}_i, h) = \sqrt{(\vec{R}_i - \vec{Q}_i - \vec{n}_i h)^2} \tag{3.2}$$

The normal vector $\vec{n}_i$ can be computed by

$$\vec{n}_i = \frac{\vec{P}_u(u, v) \times \vec{P}_v(u, v)}{|\vec{P}_u(u, v) \times \vec{P}_v(u, v)|} \tag{3.3}$$

The minimum distance function (3.2) describes the Euclidean distance between two points. Still, there is another way to calculate the same distance. If $\vec{R}_i$ has an orthogonal projection on $\vec{P}(u, v)$, then the *oriented distance* gives the same value:

$$d(\vec{R}_i, \vec{Q}_i, h) = \vec{n}_i \cdot (\vec{R}_i - \vec{Q}_i) - h \tag{3.4}$$

Jinkerson uses this distance function in [18]. It is important to remember that this function is only valid, if an orthogonal projection of $\vec{R}_i$ on $\vec{P}(u, v)$ exists!

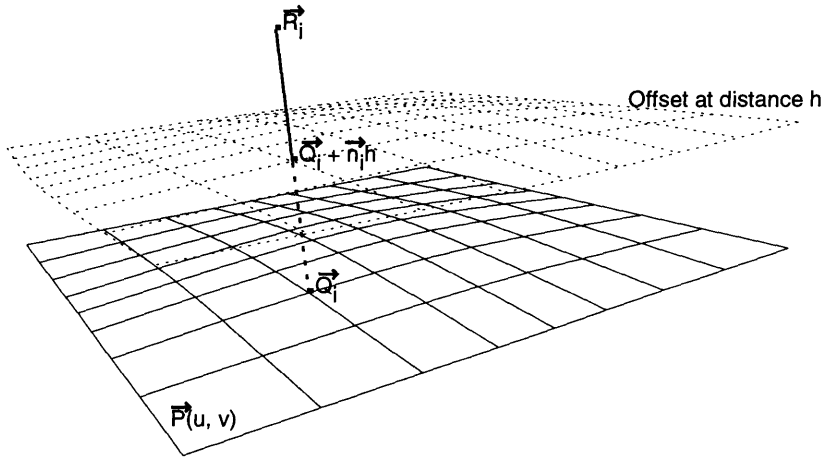


Figure 3-2: Relation between a measured point and the surface

To gain consistency with the algorithm described here, we use a slightly modified version of (3.4):

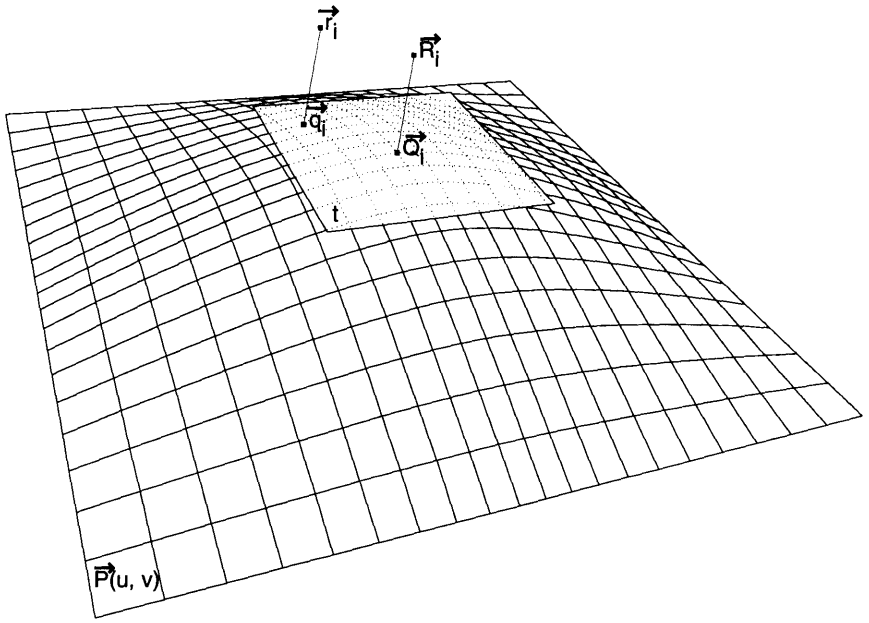
$$d(\vec{R}_i, \vec{Q}_i, h) = \vec{n}_i \cdot (\vec{Q}_i - \vec{R}_i) - h \tag{3.5}$$

The difference is the definition of whether a negative distance is on one or on the other side of the patch.

The process of finding the footpoint $\vec{Q}_i$ is described in [18]. A modified Newton method is used, and the process is stable even if no orthogonal projection of the measured point on the surface exists.

The two distance measures, the signed distance and the Euclidean distance, are measures which we can understand intuitively. But one might wonder whether there are better measurements for the distance, where "distance" does not necessarily has to represent the Euclidean distance any longer. One such distance measure has been suggested by Professor André Clément. In this new approach we do not measure the distance to the footpoint, but the distance to the tangent plane of the surface at the initial footpoint and shifted by the offset (see Fig. 3-3).

In this approach, we refer not only to the measured points $\vec{R}_i$ with their corresponding footpoints $\vec{Q}_i$; we also refer to the rotated, translated and offset points $\vec{r}_i$



In this approach the distance of a translated and rotated point $\vec{r}_i$ depends on its original position $\vec{R}_i$, before the localization process started. $\vec{Q}_i$ is the footpoint to the measured point $\vec{R}_i$. In this approach, we measure the minimum distance to the plane t, which is tangent to $\vec{P}(u,v)$ at $\vec{Q}_i$, instead of the minimum distance to the patch $\vec{P}(u,v)$. This plane t is determined by the footpoint $\vec{Q}_i$ (which does not change during the localization process) and the offset distance h, which might change. (The offset is not shown in this sketch, to keep it simple.)

Figure 3-3: Distance to tangent plane

with their new footpoints $\vec{q}_i$. For their exact definition, refer to section 3.2.2.

The approach suggested by Professor André Clément is similar to Jinkerson's. In fact, it is identical to (3.5), except that we always refer to the original footpoint $\vec{Q}_i$ (instead to the moved footpoint $\vec{q}_i$). With the notation suggested above, we use the distance measure

$$d(\vec{r}_i, \vec{Q}_i, h) = \vec{n}_i \cdot (\vec{Q}_i - \vec{r}_i) - h \tag{3.6}$$

where $\vec{n}_i$ is the unit normal vector of $\vec{P}(u,v)$ at $\vec{Q}_i$.

In the following we will concentrate on developing a framework to implement the localization algorithm using the Euclidean distance. In the end of this chapter we will investigate the tangent plane distance further.

3.2.2 Objective function

Localization is an iterative procedure searching for improved positions for the measured points. During each iteration the initial measured points $\vec{R}_i$ are rotated by angles ψ, θ and ϕ about the x, y and z axis, respectively, and are translated by the vector $\vec{t} = [t_x, t_y, t_z]$. The angles are the Euler angles of rotation. Let $\vec{r}_i$ be the positions attained by the measured points after the completion of an iteration step.

$$\vec{r}_i = [C]\vec{R}_i + \vec{t} \tag{3.7}$$

We use the notation $\vec{q}_i$ for the point nearest to the transformed points $\vec{r}_i$ on the offset surface with distance h from the design surface $\vec{P}(u, v)$. The position of the point $\vec{q}_i$ varies together with the motion of the point $\vec{r}_i$ depending on the six motion parameters. Therefore $\vec{q}_i$ can be viewed as a vector-valued function depending on the variables ψ, θ, ϕ, t_x, t_y, t_z and h. The rotation matrix $[C] = [C](\psi, \theta, \phi)$ was introduced by Jinkerson in [18] and has been proved to be orthogonal:

$$[C] = \begin{bmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ \cos\psi\sin\phi + \sin\theta\sin\psi\cos\phi & \cos\psi\cos\phi - \sin\theta\sin\phi\sin\psi & -\sin\psi\cos\theta \\ \sin\psi\sin\phi - \sin\theta\cos\phi\cos\psi & \sin\psi\cos\phi + \sin\theta\sin\phi\cos\psi & \cos\theta\cos\psi \end{bmatrix} \tag{3.8}$$

Altogether the objective function is

$$F(\psi, \theta, \phi, t_x, t_y, t_z, h) = \sum_{i=1}^{m} d_i^2(\vec{r}_i, \vec{q}_i, h) \tag{3.9}$$

3.3 The gradient vector

The algorithm used to find the optimal rigid body transformation of the measured points to the design surface uses the gradient of the objective function (3.9) to speed

up the minimization process.

Using the gradient vector is not strictly necessary; there are other approaches, which perform the localization by using a numerical estimate of the gradient vector, but they are slower. The advantage of not using the symbolic gradient is generality. Choi and Kurfess [11], for example, developed a localization algorithm which works on several different types of surfaces, which can be implicit or parametric. If a new surface type has to be added, the algorithm requires only the implementation of a distance function for the new surface type.

As the unit normal vector $\vec{n}_i$ and the distance vector $\vec{d}_i$ do not necessarily have the same direction (as required by Jinkerson in [18]), we have to use the Euclidean distance function (3.2) instead of the oriented distance function (3.5). So we use the distance function

$$d_i = \sqrt{(\vec{r}_i - \vec{q}_i - \vec{n}_i h)^2} \tag{3.10}$$

It is helpful to define the distance vector

$$\vec{d}_i = \vec{r}_i - \vec{q}_i - \vec{n}_i h \tag{3.11}$$

The objective function (3.9) has to be minimized. We want to provide its gradient vector, which consists of the following seven components:

$$\frac{\partial d_i^2}{\partial \psi}, \frac{\partial d_i^2}{\partial \theta}, \frac{\partial d_i^2}{\partial \phi}, \frac{\partial d_i^2}{\partial t_x}, \frac{\partial d_i^2}{\partial t_y}, \frac{\partial d_i^2}{\partial t_z}, \frac{\partial d_i^2}{\partial h}$$

3.3.1 Derivatives with respect to ψ, θ, ϕ

First we want to take the derivative of (3.9) with respect to the angles. This process is equivalent for all three angles, so it is shown here only for ψ. We omit subscript i for d for simplicity.

$$\begin{aligned}
\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial\psi} &= 2d(\vec{r}_i,\vec{q}_i,h)d_\psi(\vec{r}_i,\vec{q}_i,h) \\
&= 2d(\vec{r}_i,\vec{q}_i,h)\frac{\partial}{\partial\psi}\sqrt{(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2} \\
&= \frac{2d(\vec{r}_i,\vec{q}_i,h)}{2\sqrt{(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2}}\frac{\partial}{\partial\psi}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2 \\
&= \frac{2d(\vec{r}_i,\vec{q}_i,h)}{2d(\vec{r}_i,\vec{q}_i,h)}\frac{\partial}{\partial\psi}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2 \\
&= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial\psi}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)
\end{aligned}$$

substituting (3.7) yields:

$$\begin{aligned}
\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial\psi} &= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial\psi}([C]\vec{R}_i+\vec{t}-\vec{q}_i-\vec{n}_ih) \\
&= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot([C]_\psi\vec{R}_i+\vec{t}_\psi-(\vec{q}_i)_\psi-(\vec{n}_i)_\psi h)
\end{aligned}$$

$\frac{\partial d_i^2}{\partial\theta}$ and $\frac{\partial d_i^2}{\partial\phi}$ can be developed in the same way. With $\vec{t}_\psi=\vec{t}_\theta=\vec{t}_\phi=\vec{0}$ we get finally

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial\psi} = 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot([C]_\psi\vec{R}_i-(\vec{q}_i)_\psi-(\vec{n}_i)_\psi h) \tag{3.12}$$

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial\theta} = 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot([C]_\theta\vec{R}_i-(\vec{q}_i)_\theta-(\vec{n}_i)_\theta h) \tag{3.13}$$

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial\phi} = 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot([C]_\phi\vec{R}_i-(\vec{q}_i)_\phi-(\vec{n}_i)_\phi h) \tag{3.14}$$

3.3.2 Derivatives with respect to t_x, t_y, t_z

We take the derivatives of (3.9) with respect to the translations in a similar way. This process is equivalent for all three translations, so it is shown here only for t_x.

$$\begin{aligned}
\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial t_x} &= 2d(\vec{r}_i,\vec{q}_i,h)d_{t_x}(\vec{r}_i,\vec{q}_i,h) \\
&= 2d(\vec{r}_i,\vec{q}_i,h)\frac{\partial}{\partial t_x}\sqrt{(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2} \\
&= \frac{2d(\vec{r}_i,\vec{q}_i,h)}{2\sqrt{(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2}}\frac{\partial}{\partial t_x}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2 \\
&= \frac{2d(\vec{r}_i,\vec{q}_i,h)}{2d(\vec{r}_i,\vec{q}_i,h)}\frac{\partial}{\partial t_x}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2 \\
&= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial t_x}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)
\end{aligned}$$

substituting (3.7) yields:

$$\begin{aligned}
\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial t_x} &= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial t_x}([C]\vec{R}_i+\vec{t}-\vec{q}_i-\vec{n}_ih) \\
&= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot([C]_{t_x}\vec{R}_i+\vec{t}_{t_x}-(\vec{q}_i)_{t_x}-(\vec{n}_i)_{t_x}h)
\end{aligned}$$

$\frac{\partial d_i^2}{\partial t_y}$ and $\frac{\partial d_i^2}{\partial t_z}$ can be developed in the same way. With $[C]_{t_x}=[C]_{t_y}=[C]_{t_z}=[0]$ we finally get

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial t_x} = 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot(\vec{t}_{t_x}-(\vec{q}_i)_{t_x}-(\vec{n}_i)_{t_x}h) \tag{3.15}$$

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial t_y} = 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot(\vec{t}_{t_y}-(\vec{q}_i)_{t_y}-(\vec{n}_i)_{t_y}h) \tag{3.16}$$

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial t_z} = 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot(\vec{t}_{t_z}-(\vec{q}_i)_{t_z}-(\vec{n}_i)_{t_z}h) \tag{3.17}$$

3.3.3 Derivative with respect to h

Similarly, the derivative of (3.9) with respect to the offset h can be found:

$$\begin{aligned}
\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial h} &= 2d(\vec{r}_i,\vec{q}_i,h)d_h(\vec{r}_i,\vec{q}_i,h) \\
&= 2d(\vec{r}_i,\vec{q}_i,h)\frac{\partial}{\partial h}\sqrt{(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2} \\
&= \frac{2d(\vec{r}_i,\vec{q}_i,h)}{2\sqrt{(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2}}\frac{\partial}{\partial h}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2 \\
&= \frac{2d(\vec{r}_i,\vec{q}_i,h)}{2d(\vec{r}_i,\vec{q}_i,h)}\frac{\partial}{\partial h}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)^2 \\
&= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial h}(\vec{r}_i-\vec{q}_i-\vec{n}_ih)
\end{aligned}$$

substituting (3.7) yields:

$$\begin{aligned}
\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial h} &= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial h}([C]\vec{R}_i+\vec{t}-\vec{q}_i-\vec{n}_ih) \\
&= 2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot([C]_h\vec{R}_i+\vec{t}_h-(\vec{q}_i)_h-(\vec{n}_i)_hh)
\end{aligned}$$

But $[C]$, $\vec{t}$ and $\vec{q}_i$ do not depend on h, and therefore their derivatives with respect to h vanish. Consequently:

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial h} = -2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\frac{\partial}{\partial h}\vec{n}_ih$$

But $\frac{\partial}{\partial h}h = 1$, and $\vec{n}_i$ is independent from h, so we get finally

$$\frac{\partial d^2(\vec{r}_i,\vec{q}_i,h)}{\partial h} = -2(\vec{r}_i-\vec{q}_i-\vec{n}_ih)\cdot\vec{n}_i \tag{3.18}$$

We assemble the gradient vector from (3.12) to (3.18) and get

$$\nabla F = \sum_{i=1}^{m} \begin{bmatrix} 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot ([C]_\psi \vec{R}_i - (\vec{q}_i)_\psi - (\vec{n}_i)_\psi h) \\ 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot ([C]_\theta \vec{R}_i - (\vec{q}_i)_\theta - (\vec{n}_i)_\theta h) \\ 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot ([C]_\phi \vec{R}_i - (\vec{q}_i)_\phi - (\vec{n}_i)_\phi h) \\ 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot (\vec{t}_{t_x} - (\vec{q}_i)_{t_x} - (\vec{n}_i)_{t_x} h) \\ 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot (\vec{t}_{t_y} - (\vec{q}_i)_{t_y} - (\vec{n}_i)_{t_y} h) \\ 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot (\vec{t}_{t_z} - (\vec{q}_i)_{t_z} - (\vec{n}_i)_{t_z} h) \\ -2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot \vec{n}_i \end{bmatrix} \tag{3.19}$$

It should be noted that the expression $(\vec{r}_i - \vec{q}_i - \vec{n}_i h)$ appears in all components and has already been defined in (3.11) as the distance vector $\vec{d}_i$.

3.4 Interpretation of the gradient vector

Measured points can be classified into three categories, depending on their location:

Interior point: The point has an orthogonal projection on the surface.

Border point: The point has an orthogonal projection on one of the four border curves.

Corner point: All remaining points.

An example of each case is shown in Fig. 3-1.

In the following we will see, that all cases simplify to one simplified gradient vector.

3.4.1 "Corner" point

This case can be visualized as the "Corner point" in Fig. 3-1. Suppose the measured point $\vec{p}_i$ is moved by a small amount in either direction. If the movement is small enough, the position of the footpoint $\vec{q}_i$ will not change at all, and neither the position

of the normal $\vec{n}_i$. So

$$(\vec{q}_i)_\psi = (\vec{q}_i)_\theta = (\vec{q}_i)_\phi = (\vec{n}_i)_\psi = (\vec{n}_i)_\theta = (\vec{n}_i)_\phi = 0$$

$$(\vec{q}_i)_{t_x} = (\vec{q}_i)_{t_y} = (\vec{q}_i)_{t_z} = (\vec{n}_i)_{t_x} = (\vec{n}_i)_{t_y} = (\vec{n}_i)_{t_z} = 0$$

and (3.19) simplifies to

$$\nabla F = \sum_{i=1}^{m} \begin{bmatrix} 2\vec{d}_i \cdot [C]_\psi \vec{R}_i \\ 2\vec{d}_i \cdot [C]_\theta \vec{R}_i \\ 2\vec{d}_i \cdot [C]_\phi \vec{R}_i \\ 2d_i^x \\ 2d_i^y \\ 2d_i^z \\ -2\vec{d}_i \cdot \vec{n}_i \end{bmatrix} \tag{3.20}$$

where superscripts x, y, z denote components. All unknowns disappear, so all information is available to compute this gradient vector.

3.4.2 "Border" point

A border point is a measured point in a position, where the footpoint $\vec{q}_i$ is located on the border of the patch, and the projection of the measured point $\vec{p}_i$ is not orthogonal (like the "Border point" in Fig. 3-1). As it will turn out, for the resulting gradient vector it does not matter which of the four borders we observe.

Equation (3.19) can be simplified by using orthogonality properties. The key idea is, that the $\vec{q}_i$ and $\vec{n}_i$ can only slide along the border. That means, all derivatives of $\vec{q}_i$ and $\vec{n}_i$ have the direction of the tangent to the border curve.

First we rewrite (3.19) by multiplying some of the components out. F_ψ can be written as follows:

$$\begin{aligned} F_\psi &= \sum_{i=1}^{m} 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot ([C]_\psi \vec{R}_i - (\vec{q}_i)_\psi - (\vec{n}_i)_\psi h) \\ &= \sum_{i=1}^{m} 2[(\vec{r}_i - \vec{q}_i) \cdot [C]_\psi \vec{R}_i - (\vec{r}_i - \vec{q}_i) \cdot (\vec{q}_i)_\psi - (\vec{r}_i - \vec{q}_i) \cdot (\vec{n}_i)_\psi h \\ &\quad -\vec{n}_i \cdot [C]_\psi \vec{R}_i h + \vec{n}_i \cdot (\vec{q}_i)_\psi h + \vec{n}_i \cdot (\vec{n}_i)_\psi h^2] \end{aligned} \tag{3.21}$$

As sliding is only possible along the border, all derived vectors are linear dependent. Assuming that $\vec{v}$ is the direction of the border, we have the parallel properties $\vec{v} \parallel (\vec{q}_i)_\psi \parallel (\vec{n}_i)_\psi$ and the orthogonality properties $(\vec{r}_i - \vec{q}_i) \perp \vec{v}$ and $\vec{n}_i \perp \vec{v}$. From this follows

$$\begin{aligned} (\vec{r}_i - \vec{q}_i) \cdot (\vec{q}_i)_\psi &= 0 \\ (\vec{r}_i - \vec{q}_i) \cdot (\vec{n}_i)_\psi h &= 0 \\ \vec{n}_i \cdot (\vec{q}_i)_\psi h &= 0 \\ \vec{n}_i \cdot (\vec{n}_i)_\psi h^2 &= 0 \end{aligned}$$

So (3.21) simplifies to

$$F_\psi = \sum_{i=1}^{m} 2\left[(\vec{r}_i - \vec{q}_i) \cdot [C]_\psi \vec{R}_i - \vec{n}_i \cdot [C]_\psi \vec{R}_i h\right] = 2\vec{d}_i \cdot [C]_\psi \vec{R}_i \tag{3.22}$$

The same orthogonality properties can be applied on the other partial derivatives:

$$F_\theta = 2\vec{d}_i \cdot [C]_\theta \vec{R}_i \tag{3.23}$$

$$F_\phi = 2\vec{d}_i \cdot [C]_\phi \vec{R}_i \tag{3.24}$$

Similar conditions can be found for the translations:

$$\begin{aligned} F_{tx} &= \sum_{i=1}^{m} 2(\vec{r}_i - \vec{q}_i - \vec{n}_i h) \cdot (t_{tx} - (\vec{q}_i)_{tx} - (\vec{n}_i)_{tx} h) \\ &= \sum_{i=1}^{m} 2[(\vec{r}_i - \vec{q}_i) \cdot \vec{t}_{tx} - (\vec{r}_i - \vec{q}_i) \cdot (\vec{q}_i)_{tx} - (\vec{r}_i - \vec{q}_i) \cdot (\vec{n}_i)_{tx} h \\ &\quad -\vec{n}_i \cdot \vec{t}_{tx} h + \vec{n}_i \cdot (\vec{q}_i)_{tx} h + \vec{n}_i \cdot (\vec{n}_i)_{tx} h^2] \end{aligned} \tag{3.25}$$

By using the orthogonality properties, we obtain

$$\begin{aligned} F_{tx} &= \sum_{i=1}^{m} 2d_i^x \\ F_{ty} &= \sum_{i=1}^{m} 2d_i^y \\ F_{tz} &= \sum_{i=1}^{m} 2d_i^z \end{aligned}$$

The component F_h can not be simplified any more. Altogether the gradient vector is

$$\nabla F = \sum_{i=1}^{m} \begin{bmatrix} 2\vec{d}_i \cdot [C]_\psi \vec{R}_i \\ 2\vec{d}_i \cdot [C]_\theta \vec{R}_i \\ 2\vec{d}_i \cdot [C]_\phi \vec{R}_i \\ 2d_i^x \\ 2d_i^y \\ 2d_i^z \\ -2\vec{d}_i \cdot \vec{n}_i \end{bmatrix} \tag{3.26}$$

Surprisingly, this vector is identical with (3.20), the gradient vector of the "Corner point".

3.5 Relation to the earlier formulation of the gradient

The gradient vector developed by Jinkerson in [18] is

$$\nabla F = \sum_{i=1}^{m} \begin{bmatrix} 2d_i(\vec{n}_i \cdot [C]_\psi \vec{R}_i) \\ 2d_i(\vec{n}_i \cdot [C]_\theta \vec{R}_i) \\ 2d_i(\vec{n}_i \cdot [C]_\phi \vec{R}_i) \\ 2d_i n_i^x \\ 2d_i n_i^y \\ 2d_i n_i^z \\ -2d_i \end{bmatrix} \tag{3.27}$$

This gradient is only valid, if the measured points $\vec{r}_i$ have orthogonal projections $\vec{q}_i$ on the patch. If (3.26) is compared with (3.27) for measured points with orthogonal projections on the patch (e. g. the scope of Jinkerson's gradient), the following terms are identical:

$$\begin{aligned} \vec{d_i} &= d_i \vec{n}_i \\ d_i^x &= d_i n_i^x \\ d_i^y &= d_i n_i^y \\ d_i^z &= d_i n_i^z \end{aligned}$$

If the above four expressions are substituted in (3.26), it yields (3.27). That means that Jinkerson's gradient vector can be substituted by (3.26) and that again means, that we do not have to distinguish between the three different cases at all.

3.6 Constrained gradient function

3.6.1 Introduction

Unconstrained localization addresses the problem of determining the optimal positioning of a set of measured points relative to a design surface when each measured point has an equal effect on the determination of positioning. *Constrained localization*, on the other hand, involves the problem of determining a *feasible*, but not necessarily optimal, positioning of a set of measured points relative to a design surface when subsets of the measured points can have *unequal* effects on the determination of positioning.

The unconstrained localization problem seeks to minimize one global objective function so the measured points are all collectively brought as close to the design surface as possible. In this problem each point contributes with the same weight to the minimization of the objective function. In contrast, the constrained localization problem starts with the rotations and translation produced by the minimized objective function of the unconstrained localization problem and determines a rigid body transformation which will allow the measured points to satisfy a set of nonlinear constraints. The constrained localization problem does not minimize an objective function, but rather uses minimization techniques to find a feasible transformation that will satisfy the constraints imposed by a set of constraint functions. Satisfying the constraint functions has the effect of changing the importance of each measured point.

The need for constrained localization of marine propellers is evident because of the tougher positional tolerances that are required near the leading edge of a blade. A method is required which will provide for greater influence on the localization by measured points that are close to the leading edge.

3.6.2 Selecting a constraint function

The selection of an appropriate constraint function is fundamental to the formulation of the constrained localization problem. The constraint function must certainly be a distance measure, but careful definition of this measure may facilitate the solution of the problem.

Jinkerson [18] discusses the choice of the constraint function. It was considered to choose the squared distance as the constraint function. A drawback of that approach is less precision. Furthermore, the squared distance function as an *unsigned* function causes all sense of position of a measured point relative to the design surface normal vector orientation to be lost. The loss of positional sense of a measured point relative to the design surface is undesirable in the context of using the process as an inspection tool.

Using the oriented distance as the constraint function, the localization has m constraining functions c_i, one for each transformed measured point $\vec{r}_i$, so that

$$c_i = d(\vec{r}_i, \vec{q}_i, h) \tag{3.28}$$

where d is the oriented distance (3.5). The bounds for the constraining function c_i depend upon the u, v parameters of the initial footpoint $\vec{Q}_i$. Assume chordwise parametrization of the design surface in the parameter u and that u_{min} and u_{max} are the isoparameter lines describing the leading-edge region boundaries, $u_{min} < u_{max}$. Let ϵ be the constraining bounds for c_i if the u parameter of the corresponding footpoint $\vec{Q}_i$ is inside the leading-edge region and let δ be the bounds otherwise. Typically, $\epsilon \ll \delta$. Then the constrains are:

$$-\epsilon \leq c_i \leq \epsilon \quad \text{if } u_{min} \leq u_i \leq u_{max} \tag{3.29}$$

$$-\delta \leq c_i \leq \delta \quad \text{otherwise} \tag{3.30}$$

3.6.3 Constrained localization algorithm

The constrained localization is the problem of determining the rotations and translations which must be performed on the set of measured points so that they will satisfy the required localization constraints. The problem of determining the set of six parameters which will allow the set of measured points to satisfy the localization constraints is solved using the routine E04UCF for nonlinear constrained optimization problems provided by Numerical Algorithms Group (NAG). The routine uses an iterative sequential quadratic programming algorithm in which the search direction is the solution of a quadratic programming problem [12, 24].

The nonlinear constrained optimization routine estimates gradients of user-supplied functions with difference quotient unless the user can also supply those gradients. The latter situation produces a great improvement in computational accuracy and efficiency. For this reason, part of the implementation of this algorithm involves providing symbolic gradients for each of the functions that are supplied to the NAG routine.

3.6.4 Constrained function and gradient

As the algorithm requires a signed distance, we can not just substitute the Euclidean distance (3.10) in (3.28). To solve this problem, we calculate both, the oriented distance (3.5) and the Euclidean distance (3.10). We use the sign of the oriented distance and assign it to the Euclidean distance.

For the gradient vector we have to distinguish two cases. Depending on the sign of d_i we have to use the right gradient of the following two:

$$\begin{aligned} \nabla d_i &= \nabla\sqrt{(\vec{r}_i - \vec{q}_i - \vec{n}_i h)^2} \\ \nabla(-d_i) &= \nabla - \sqrt{(\vec{r}_i - \vec{q}_i - \vec{n}_i h)^2} \end{aligned}$$

The process of taking derivatives is almost identical to the steps shown in section 3.3.1 and 3.3.2 and it yields

$$\nabla c_i = \pm \begin{bmatrix} \vec{u}_i \cdot ([C]_\psi \vec{R}_i - (\vec{q}_i)_\psi - (\vec{n}_i)_\psi h) \\ \vec{u}_i \cdot ([C]_\theta \vec{R}_i - (\vec{q}_i)_\theta - (\vec{n}_i)_\theta h) \\ \vec{u}_i \cdot ([C]_\phi \vec{R}_i - (\vec{q}_i)_\phi - (\vec{n}_i)_\phi h) \\ \vec{u}_i \cdot (\vec{t}_{t_x} - (\vec{q}_i)_{t_x} - (\vec{n}_i)_{t_x} h) \\ \vec{u}_i \cdot (\vec{t}_{t_y} - (\vec{q}_i)_{t_y} - (\vec{n}_i)_{t_y} h) \\ \vec{u}_i \cdot (\vec{t}_{t_z} - (\vec{q}_i)_{t_z} - (\vec{n}_i)_{t_z} h) \\ -\vec{u}_i \cdot \vec{n}_i \end{bmatrix} \tag{3.31}$$

where

$$\vec{u}_i = \frac{(\vec{r}_i - \vec{q}_i - \vec{n}_i h)}{|\vec{r}_i - \vec{q}_i - \vec{n}_i h|} \tag{3.32}$$

$\vec{u}_i$ is simply the unit vector in the direction of $\vec{d}_i$. The sign of d influences the whole gradient vector by changing the sign of all components.

As for the gradient vector of the objective function, three cases can be distinguished. The measured point as in Fig. 3-1 can be an interior, a border or a corner point. Again, all cases simplify to one case which includes the gradient introduced earlier by Jinkerson et al. [19]. As the calculations are quite similar to the way shown for the objective function, only the result is presented here:

$$\nabla c_i = \pm \begin{bmatrix} 2\vec{u}_i \cdot [C]_\psi \vec{R}_i \\ 2\vec{u}_i \cdot [C]_\theta \vec{R}_i \\ 2\vec{u}_i \cdot [C]_\phi \vec{R}_i \\ 2u_i^x \\ 2u_i^y \\ 2u_i^z \\ -2\vec{u}_i \cdot \vec{n}_i \end{bmatrix} \tag{3.33}$$

Compared to the earlier approach, this algorithm is computationally more expensive, as the distance is calculated twice (oriented distance, to get the sign, and Euclidean distance, to get the value).

3.7 Tangent plane distance

As mentioned earlier, an approach similar to Jinkerson's is the use of the tangent plane distance, which is given by (3.6). For this distance function we do not have to derive the gradient vector, as it is identical with the one developed by Jinkerson, as we just have to substitute $\vec{q}_i$ by $\vec{Q}_i$.

What can we expect from such a distance function? First, as the initial position of the point set determines the footpoints used to create the tangent plane, we have to expect that the localization results depend on the initial position of the measured point set. Second, we can also expect a performance increase. Using tangent planes assigns a plane to every measured point, and this plane does not change during the localization process. Thus, the distance of a measured point to its plane is now a *linear* function with respect to a translation t_x, t_y, t_z and the offset h of this point (with respect to the rotations, the distance is well approximated by a linear function, if we assume small angles). The numerical solver we use to minimize the objective function uses a modified Newton method [24]. But the Newton method converges very fast on a linear function (in fact, on a linear function with one unknown it finds the solution in one iteration step).

And last, we must expect the "real" RMS (defined by (3.9) and using the Euclidean distance) to be larger or equal to the RMS found by the localization using the Euclidean distance to the tangent planes. As we do not localize to the curved patch any more (but to a faceted approximation), we cannot expect an exact result. And as we assume that the Euclidean distance method finds a global minimum of the objective function, every approximation to that solution can only create a larger

RMS.

3.8 Conclusion

A new gradient vector (3.26) for the objective function has been developed. It turns out that this new gradient is valid for all measured points, including interior, border and corner points. Inside the patch it simplifies to the earlier formulation of the gradient vector, that was introduced by Jinkerson et al. [18]. So this approach extends the existing localization algorithm, rather than substituting it. As the objective function does not change during the localization process, even if points do not have orthogonal projections, the localization process has been stabilized and guaranteed to find a unique best solution.

Also a new constraint function (3.28) and its gradient (3.33) have been developed, which are generally valid, no matter whether an orthogonal projection of the measured point on the design surface exists.

We also studied the "tangent plane distance function", which is strongly related to the signed distance function. It is worthwhile exploring this approach, as it might lead to a method to speed up the localization procedure.

Chapter 4

Examples of the general localization algorithm

4.1 Introduction

The focus of chapter 3 was to extend the method developed by Jinkerson et al. [18], from now on called *old algorithm*, to a more general methodology, to cover also exceptional points, where no orthogonal projection of a measured point on the design surface exists. Jinkerson's algorithm has been proved to be reliable, so the improvement presented here would be worthless if it can not be shown to be stable and consistent.

First it will be shown that the new algorithm delivers the same rigid body transformations as the one introduced by Jinkerson, as long as all measured points have orthogonal projections on the design surface. Then some examples from existing applications will be shown.

The algorithm has been implemented in PRAXITELES 9.0[1]. To compare the lo-

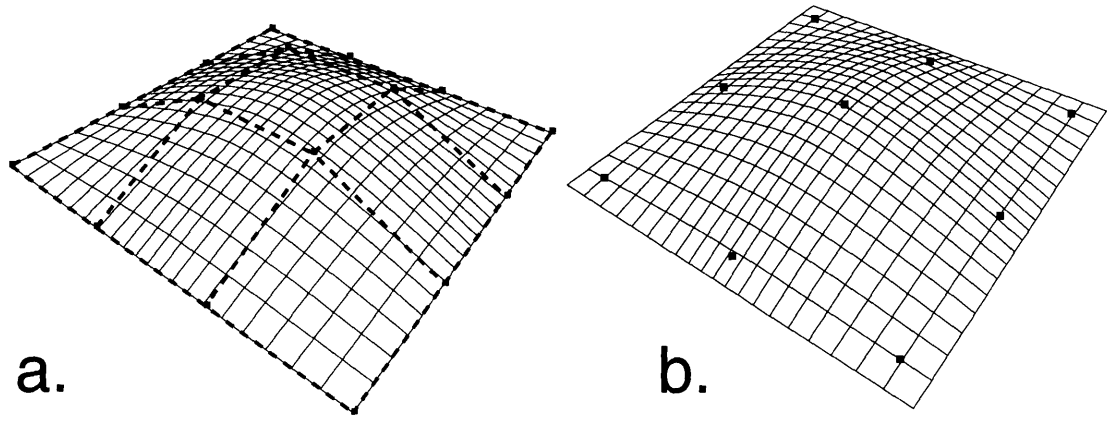
[1] PRAXITELES is an interactive geometric modeling system for sculptured curves and surfaces. It has been developed in the Ocean Engineering Design Laboratory at MIT with funding from various U. S. government agencies [17, 30, 2]. PRAXITELES 9.0 was in the alpha testing phase during the development of the new localization algorithm.

calization performance with the earlier localization algorithm, the same localizations have been performed with PRAXITELES 8.0, which has the implementation of Jinkerson's algorithm.

Although it has not been investigated in detail, the localizations have also been performed using the tangent plane distance function.

4.2 Example 1: Biquadratic B-spline patch

To check systematically the properties of the new localization algorithm, a simple patch has been defined (Fig. 4-1a). The patch is a biquadratic B-spline patch and has a four by four control polyhedron.



a. A biquadratic B-spline patch with a four by four control polyhedron. The control points are listed in Table A.1.
b. If a small displacement is performed on the point set shown here, all points will still have an orthogonal projection on the patch. The data points are listed in Table A.2, the result of the localization in Table A.3.

Figure 4-1: The experimental patch

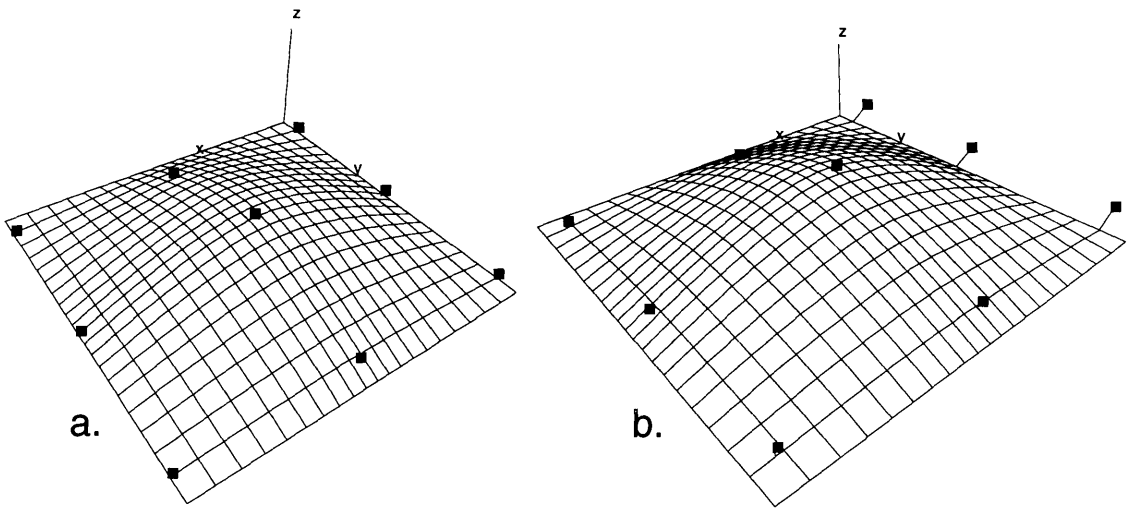
4.2.1 Measured points with orthogonal projections

The first sample point set consists of nine points, which are distributed regularly over the patch (Fig. 4-1b). The control points are chosen in a way, that the optimal position

requires a displacement in z-direction, no displacement in x- or y-direction and no rotations. As expected, the localization results of PRAXITELES 8.0 and PRAXITELES 9.0 are identical. During the localization process the projections of the measured points were always orthogonal. Both algorithms need exactly the same number of iteration steps and deliver the same result (Table A.3). This shows that the new localization algorithm has the same capabilities as the earlier algorithm.

4.2.2 Measured points with non-orthogonal projections

If a small displacement of a measured point can cause the projection not to be orthogonal any more, (Fig. 4-2 a), the old algorithm will start to disable those points during the localization process. The number of disabled points can vary from iteration step to iteration step.



a. Six of the nine measured points are in a critical position, as a small rigid body transformation can change the orthogonality property. The set of measured points is documented in Table A.4.
b. Localization result of PRAXITELES 8.0. Although the program reported an improvement, it is clearly visible that three measured points are far away from the patch. As those points have no orthogonal projection, PRAXITELES 8.0 simply ignores them. The localization results are listed in Table 4.1.

Figure 4-2: Measured points near the border of the patch

	PRAXITELES 8.0	PRAXITELES 9.0	Tangent Plane
Points:	6 (of 9)	9 (of 9)	9 (of 9)
Iterations:	50	39	17
Results:	"Successful"	"Very good solution"	"Very good solution"
RMS, before:	0.05318910	0.05318910	0.05318910
RMS reported:	0.02512345 (52.8%)	0.04383439 (17.6%)	0.04383603 (17.6%)
RMS all Points:	0.10507048 (-97.5%)	0.04383439 (17.6%)	0.043834389 (17.6%)
Tx, Ty, Tz:	$\begin{bmatrix} -0.14698086 \\ 0.00000000 \\ 0.1012604 \end{bmatrix}$	$\begin{bmatrix} 0.00000000 \\ 0.00000000 \\ 0.03158004 \end{bmatrix}$	$\begin{bmatrix} 0.00000001 \\ -0.00000001 \\ 0.03157916 \end{bmatrix}$
Rx, Ry, Rz:	$\begin{bmatrix} 0.00000000 \\ 0.01745329 \\ 0.00000000 \end{bmatrix}$	$\begin{bmatrix} 0.00000000 \\ 0.00000000 \\ 0.00000000 \end{bmatrix}$	$\begin{bmatrix} 0.00000000 \\ 0.00000000 \\ 0.00000001 \end{bmatrix}$
Iteration Tolerance: 0.001			

The values shown in this table represent the output of PRAXITELES with the three different localization algorithms described in the previous chapter. "Result" is a rating made by PRAXITELES. Furthermore, PRAXITELES returns only "RMS reported", even if not all measured points have been used. "RMS all Points" has been generated separately to evaluate PRAXITELES' results. The "Iteration Tolerance" is an input parameter of the localization.
PRAXITELES 8.0 reports the best RMS, but only, because it is ignoring three points. If they are considered, the result looks much worse. The other two algorithms deliver both the same, correct result.
The patch with the set of measured points is shown in Fig. 4-2.

Table 4.1: Localization with centered, critical data points.

PRAXITELES 8.0 reports an improvement of the RMS[2] distance by 52.8%, while PRAXITELES 9.0 reports 17.6%. This might be confusing, but gets clear by observing the results closer. The reason is, that PRAXITELES 8.0 determines the RMS only from the measured points that were actually used for the localization, and in this case only six (out of nine) have been used. If the RMS is computed from all nine points, PRAXITELES 8.0 yields a *deterioration* by -97.5%, while PRAXITELES 9.0 gained an improvement of 17.6% (see Table 4.1). The tangent plane distance algorithm also delivers the right result, and even better, it needs only 17 (vs. 39) iteration steps!

These numbers show that the RMS distances reported by PRAXITELES 8.0 after a localization can lead to wrong conclusions, which might have serious consequences.

[2]RMS = Root Mean Square

Sometimes, but certainly not always, a visual inspection of the localization results can help (Fig. 4-2 b). With PRAXITELES 9.0 these problems will be avoided in the first place. We also see, that the tangent plane distance method looks promising, although we still have to investigate the relevance of the initial position of the measured points. The results of the localizations are shown in Table 4.1.

To see the effect of the initial position of the measured points on the localization algorithm, the same localization has been performed again. But this time, all points have been shifted by $t_x = 0.4$. We want to explore how many more iterations the Euclidean distance uses (PRAXITELES 9.0), and especially the influence on the localization with the tangent plane distance.

	PRAXITELES 8.0	PRAXITELES 9.0	Tangent Plane
Points:	6 (of 9)	9 (of 9)	9 (of 9)
Iterations:	53	38	30
Results:	"Successful"	"Very good solution"	"Successful"
RMS, before:	0.24187098	0.24187098	0.24187098
RMS reported:	0.02512345 (89.6%)	0.04383439 (81.9%)	0.04865827 (79.9%)
RMS all Points:	0.10507048 (56.5%)	0.04383439 (81.9%)	0.079913972 (67.0%)
Tx, Ty, Tz:	$\begin{bmatrix} -0.25250131 \\ 0.00000000 \\ 0.04192221 \end{bmatrix}$	$\begin{bmatrix} -0.40000000 \\ 0.00000000 \\ 0.03158004 \end{bmatrix}$	$\begin{bmatrix} -0.25488617 \\ -0.03589469 \\ 0.06442748 \end{bmatrix}$
Rx, Ry, Rz:	$\begin{bmatrix} 0.00000000 \\ -0.01745329 \\ 0.00000000 \end{bmatrix}$	$\begin{bmatrix} 0.00000000 \\ 0.00000000 \\ 0.00000000 \end{bmatrix}$	$\begin{bmatrix} 0.00165219 \\ 0.01745329 \\ 0.01745329 \end{bmatrix}$
Iteration Tolerance: 0.001			

The initial position of the data points has been shifted by $t_x = 0.4$, what mainly affects the tangent plane method. The patch with the set of measured points is shown in Fig. 4-2. Note that the *reported* RMS gives the impression that PRAXITELES 8.0 works best. But by observing all points we find that only PRAXITELES 9.0 achieved an improvement.
For a description of the table entries see Table 4.1.

Table 4.2: Localization result of a set of data points with critical points

The localization results are shown in Table 4.2. PRAXITELES 9.0 seems to be stable with respect to the initial position of the data points—the number of iteration does not change significantly (39 vs. 38 iterations). The result is identical for the

unshifted and the shifted point set (RMS of 0.04383439). By contrast, the tangent plane distance algorithm does not only need more iterations with respect to that in Table 4.1 (17 vs. 30 iterations), it also delivers a result twice as bad as with PRAXITELES 9.0 (RMS of 0.04 vs. 0.08). An explanation is, that the algorithm now operates on different tangent planes in comparison to the unshifted point set. Although the starting position chosen here is unrealistically high, it shows that results delivered by this algorithm have to be verified.

4.3 Example 2: Westinghouse turbine blade

The following example is a turbine blade manufactured by Westinghouse. The blade has been measured with a laser CMM device, which delivered two rows of measured points with 81 points per row. One row is very close to the hub of the propeller. The blade is shown in Fig. 4-3.

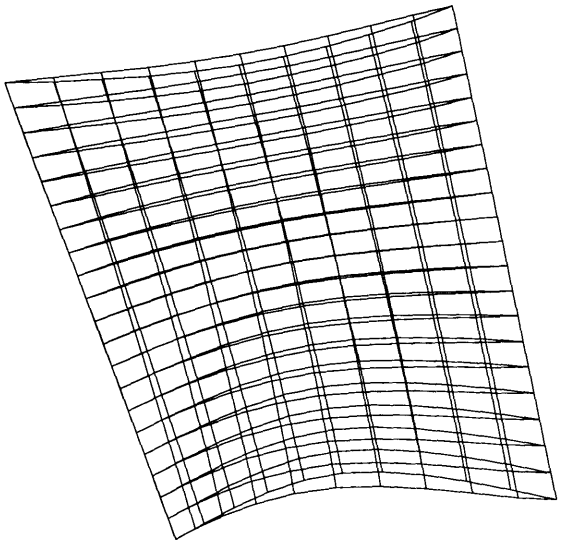
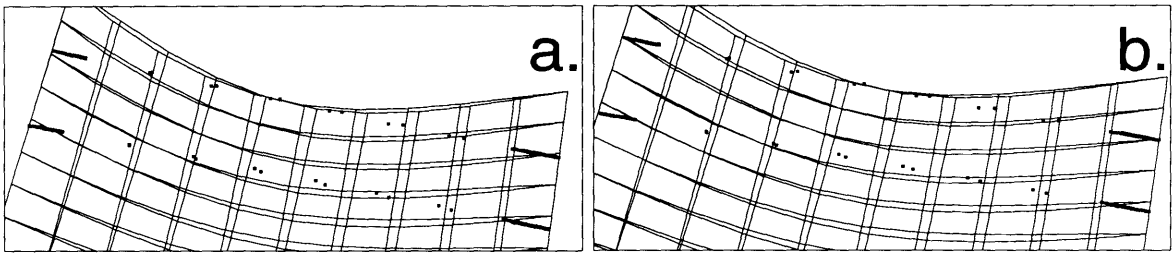


Figure 4-3: Westinghouse turbine blade

Fig. 4-4 shows the new location of the measured points on the blade, localized with PRAXITELES 8.0 and PRAXITELES 9.0. There is almost no visual difference. Table 4.3 shows the results reported by PRAXITELES 8.0, PRAXITELES 9.0 and the



a. The localization has been performed using PRAXITELES 8.0. Visual inspection gives no evidence that the localization process might not have been successful. Also, PRAXITELES 8.0 reports a "Very good solution". Still, the localization result is significantly worse than the result from PRAXITELES 9.0.
b. The localization has been performed using PRAXITELES 9.0. The position of the measured points has been improved by 28.1%, compared to only 20.0% with PRAXITELES 8.0.

Figure 4-4: Localization results on the Westinghouse turbine blade

tangent plane method.

The reported RMS distances are almost equal for the three algorithms: 28.5%, 28.1% and 28.8%. But the number of iterations differs drastically. PRAXITELES 9.0 needed 109 iterations, while PRAXITELES 8.0 and the tangent plane method performed the localization in about 30 steps. Besides that, PRAXITELES 9.0 exits with the message `"No lower point"`, which means that it was not possible to set all components of the gradient vector to zero.

If we take a closer look at the displacements and rotations, we see that PRAXITELES 9.0 localized in a significantly different direction than PRAXITELES 8.0 and the tangent plane method—still all come up with an almost identical RMS. Another look at the results tells us, that PRAXITELES 8.0 ignored two points. As we want to know the overall RMS, we recompute it, using the Euclidean distance and taking all points into account. We want to recompute the RMS for the tangent plane method as well, as the tangent plane distance is different from the Euclidean distance. In Table 4.3, the entry "RMS all Points" shows the recomputed RMS. Considering this instead of "RMS reported", we find that PRAXITELES 9.0 gives us a much better result. The RMS has been improved by 28.1%, compared to 20.0% with PRAXITELES 8.0 or 21.5% with the tangent plane method.

PRAXITELES 9.0 was not able to set the gradient to zero, so no `"success"` mes-

	PRAXITELES 8.0	PRAXITELES 9.0	Tangent Plane
Points:	160 (of 162)	162 (of 162)	162 (of 162)
Iterations:	31	109	30
Results:	"Very good solution"	"No lower point"	"Successful"
RMS, before:	0.02200943	0.02200943	0.02200943
RMS reported:	0.01574296 (28.5%)	0.01581896 (28.1%)	0.01567308 (28.8%)
RMS all Points:	0.017599042 (20.0%)	0.01581896 (28.1%)	0.017287083 (21.5%)
Tx, Ty, Tz:	$\begin{bmatrix} 0.20912135 \\ 0.15783089 \\ -0.14917195 \end{bmatrix}$	$\begin{bmatrix} 0.27480449 \\ 0.12888266 \\ -0.05523181 \end{bmatrix}$	$\begin{bmatrix} 0.20843612 \\ 0.15442764 \\ -0.14169407 \end{bmatrix}$
Rx, Ry, Rz:	$\begin{bmatrix} 0.00235642 \\ -0.00264264 \\ -0.00090982 \end{bmatrix}$	$\begin{bmatrix} 0.00178970 \\ -0.00402547 \\ 0.00000760 \end{bmatrix}$	$\begin{bmatrix} 0.00229875 \\ -0.00265736 \\ -0.00083448 \end{bmatrix}$
Iteration Tolerance: 0.001			

These are the localization results of the three different localization algorithms. Although the reported RMS values are almost identical, only PRAXITELES 9.0 reports the true RMS. PRAXITELES 8.0 and the tangent plane distance method perform worse.

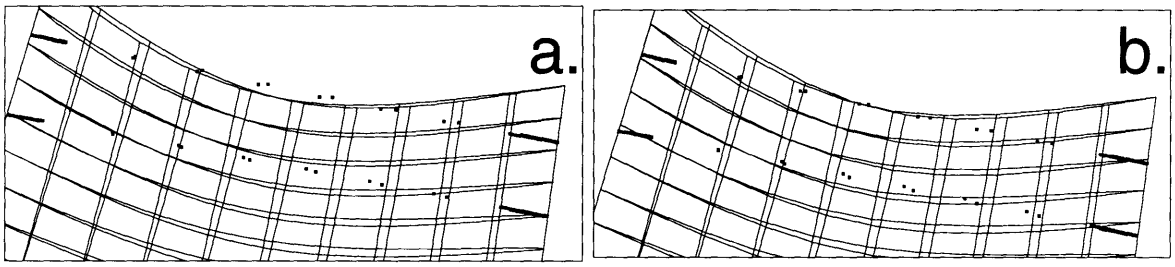
Table 4.3: Localization performed on the Westinghouse turbine blade

sage was reported, although an improvement was achieved. By ignoring two points, PRAXITELES 8.0 was able to set the gradient to zero within the iteration tolerance, so a `"Very good solution"` could be reported—although the solution is not acceptable.

One might be concerned about the drastic difference in the number of iteration steps. But this is not a problem with the localization algorithm, but with the numeric solver, which tries to set the gradient vector to zero. With this specific point set, this is not possible for PRAXITELES 9.0. The localization algorithm stopped, because the iteration limit has been reached, and not, because the solver found a solution. But the solver still minimizes the gradient, which yields a better result than the other two methods.

4.3.1 Influence of the starting position of the data points

This example has been repeated with a different starting position of the measured points. The reason for doing so was mainly to explore the effect on the tangent plane distance, as with this method the localization depends on the initial position of the footpoints and thus on the initial position of the data points.



a. Localization performed with PRAXITELES 8.0. This is the result of the localization, if the measured points are in a "bad" starting position. The bad result is caused by ignoring five data points (see Table 4.4) and the fact that the distance function operates on the wrong footpoint.
b. Localization with PRAXITELES 9.0 and the same initial position. Some of the points might not have orthogonal projections on the patch, but the algorithm still takes them into account. Although the result is usable, it is not strictly correct, as the algorithm used the "wrong" footpoint for some measured points (i.e. a closer footpoint than the selected one exists). This can be detected by the fact that "RMS reported" and "RMS all Points" differ in Table 4.4.

Figure 4-5: Westinghouse turbine blade, localization results from a "bad" starting position

But during the exploration of these cases, another problem appeared, which influenced all three localization algorithms. The point set has been shifted towards the hub of the blade. The localzation results are shwon in Fig. 4-5, and Table 4.4. The "RMS reported" is not identical with "RMS all points" for PRAXITELES 9.0 any more. But PRAXITELES 9.0 is supposed to take the Euclidean distance of all points into account, no matter whether they have an orthogonal projection or not. It turns out that this is not a problem with the localization algorithm, but with the numerical solver, which is supposed to find the minimum distance of a measured point to the surface.

Especially in the leading and trailing edge region of the blade a situation like the one shown in Fig. 4-6 might happen: a measured point $\vec{R}$ is only slightly closer to one side of the blade. The distance function uses a Newton method to find the footpoint

	PRAXITELES 8.0	PRAXITELES 9.0	Tangent Plane
Points:	157 (of 162)	162 (of 162)	162 (of 162)
Iterations:	44	107	51
Results:	"Successful"	"No lower point"	"Successful"
RMS, before:	0.17044979	0.17044979	0.17044979
RMS reported:	0.05326337 (68.8%)	0.05453509 (68.0%)	0.05282697 (69.0%)
RMS all Points:	0.088213166 (48.4%)	0.027822316 (83.7%)	0.071998054 (57.8%)
Tx, Ty, Tz:	$\begin{bmatrix} -0.40675192 \\ 0.31599731 \\ 0.32980352 \end{bmatrix}$	$\begin{bmatrix} -0.11120978 \\ 0.11077120 \\ 0.91196762 \end{bmatrix}$	$\begin{bmatrix} -0.44278615 \\ 0.25805664 \\ 0.43623420 \end{bmatrix}$
Rx, Ry, Rz:	$\begin{bmatrix} 0.00438112 \\ 0.00589277 \\ 0.00010913 \end{bmatrix}$	$\begin{bmatrix} 0.00046560 \\ -0.00113366 \\ 0.00581073 \end{bmatrix}$	$\begin{bmatrix} 0.00340569 \\ 0.00607396 \\ 0.00116466 \end{bmatrix}$
Iteration Tolerance: 0.001			

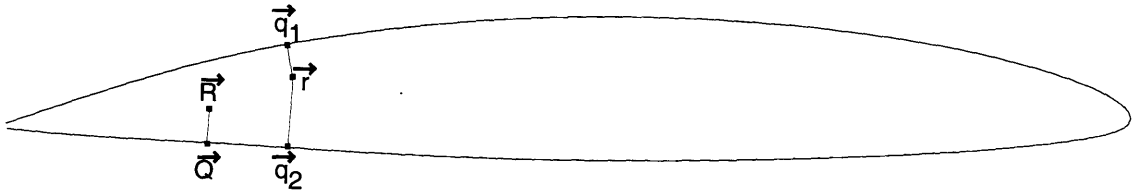
These are the localization results of the three different localization algorithms, reported by PRAXITELES. The starting position is shown in Fig. 4-4a. This example shows drastically that PRAXITELES 9.0 not only improves the result, but also stabilizes the localization process. From the reported results one would not notice that something is wrong. Only by determining the RMS for *all* points or visual inspection shows the serious mis-localization.

Table 4.4: Localization with measured points bad starting condition

$\vec{Q}$ which minimizes the distance $|\vec{Q} - \vec{R}|$. During the iteration, the footpoint of the previous iteration is used as a starting point to find the new footpoint for the new location of $\vec{r}$ (the position of $\vec{r}$ changes during the localization process, see equation (3.7)). Although $\vec{q}_1$ and $\vec{q}_2$ are close together in 3D space, they are far away in the uv parameter space of the surface. Thus, the minimum distance algorithm does not realize that $\vec{q}_1$ is closer to $\vec{r}$ than $\vec{q}_2$. Or in other words: The footpoint will hardly ever jump from one side to the other side of the blade, even if the footpoint on the other side is much closer to the measured point.

If the set of measured points has a "good" initial position before the localization starts, all points get the footpoints assigned an the side of the blade where they belong. But if the position is "bad", some points might start with footpoints on the wrong side of the blade, and this will not change during the localization process.

One way to correct this problem is to keep track of two footpoints per measured



This figure shows a cross section of a propeller blade, as it is easier to describe the issue in 2D. A measured point like the point $\vec{R}$ can easily mess up the localization process. Before the localization starts, $\vec{R}$ gets a footpoint $\vec{Q}$ assigned, which is the closest point on the surface.
During the localization process, the footpoint of the previous iteration is used as a starting point for finding the new, exact footpoint. If $\vec{R}$ moves to $\vec{r}$, the old footpoint $\vec{Q}$ is used to find the new footpoint. But in the parametric space of the surface, the closest point $\vec{q}_1$ is "far" away. The Newton algorithm finds $\vec{q}_2$ as being the closest point to $\vec{r}$, as the distance function has a local minimum.

Figure 4-6: Wrong distance measure

point during the localization. Only the closer point should be used.

4.4 Example 3: Applied Research Laboratory propeller blade

The following example has been provided by the Applied Research Laboratory (ARL) at Pennsylvania State University. They agreed to provide a fan blade that had been designed, manufactured and inspected at ARL.

The propeller design was received as a NURBS surface. The inspection data was received as x, y, z coordinates of measured points on the blade that was manufactured from the NURBS design description. These measurements were made at ARL using the Intelligent Robotic Inspection System (IRIS), which uses laser interferometry to obtain highly accurate measurements of surface coordinates. 1381 data points were received representing measurements on the pressure and suction sides of the blade. The design surface was a bicubic NURBS patch parameterized with 53 knots in the u-direction and 20 knots in the v-direction. The blade had a nominal radius of 12 inches from root to tip.

The blade with the measured points (not yet localized) is shown in Fig. 4-7. It

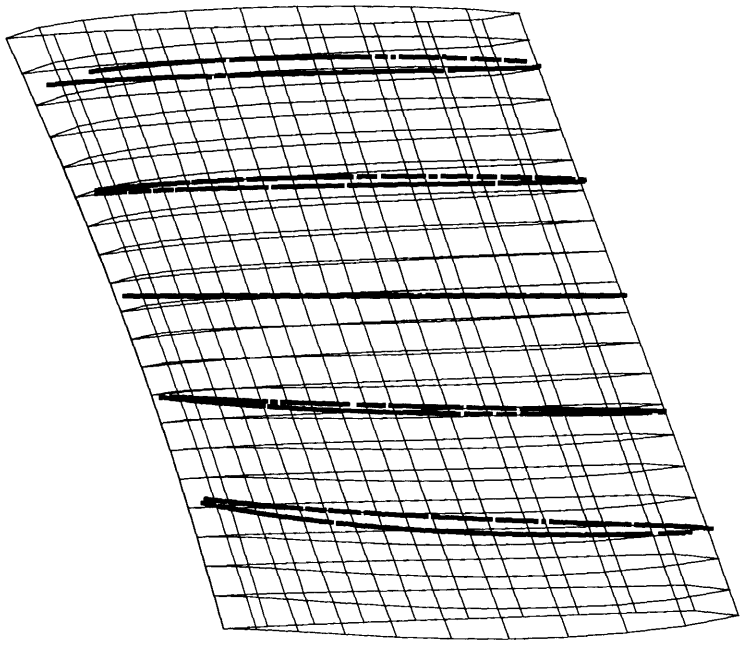


Figure 4-7: ARL fan blade

should be noted that all measured points are "far" away from the trailing edge, hub and tip. Those are the only areas, where a point could loose the property of having an orthogonal projection on the design surface. Keeping this in mind, PRAXITELES 8.0 and PRAXITELES 9.0 should deliver the same solutions. And this is the case, the results are shown in Table 4.5.

This localization process has been timed on a Silicon Graphics Onyx with four 150 MHz IP19 Processors. As it is not direct possible to determine CPU time for an interactive program, real time is used instead. The localization time was PRAXITELES 8.0: 501 sec while PRAXITELES 9.0 used 552 sec (10.2% longer). The tangent plane method performs the localization in only 60% of the time as PRAXITELES 8.0, and the RMS is worse only in the fifth digit.

	PRAXITELES 8.0	PRAXITELES 9.0	Tangent Plane
Points:	1316 (of 1316)	1316 (of 1316)	1316 (of 1316)
Iterations:	37	37	31
Results:	"Successful"	"Successful"	"Very good solution"
RMS, before:	0.03325171	0.03325171	0.03325171
RMS reported:	0.02533208 (23.8%)	0.02533208 (23.8%)	0.02528483 (24.0%)
RMS all Points:	0.02533205 (23.8%)	0.025332082 (23.8%)	0.025339513 (23.8%)
Tx, Ty, Tz:	$\begin{bmatrix} 0.12419584 \\ -0.06620158 \\ -0.03601131 \end{bmatrix}$	$\begin{bmatrix} 0.12419584 \\ -0.06620158 \\ -0.03601131 \end{bmatrix}$	$\begin{bmatrix} 0.13318697 \\ -0.07005748 \\ -0.02478989 \end{bmatrix}$
Rx, Ry, Rz:	$\begin{bmatrix} -0.00268015 \\ -0.00469398 \\ 0.00377788 \end{bmatrix}$	$\begin{bmatrix} -0.00268015 \\ -0.00469398 \\ 0.00377788 \end{bmatrix}$	$\begin{bmatrix} -0.00287191 \\ -0.00522915 \\ 0.00398723 \end{bmatrix}$
Localization Time:	501 sec	552 sec	302 sec
Iteration Tolerance: 0.001			

This example shows, that PRAXITELES 8.0 and PRAXITELES 9.0 find identical results, as there are no "critical" points. The tangent plane distance performs the localization with less iteration steps with almost the same accuracy as the PRAXITELES 8.0 and PRAXITELES 9.0.

Table 4.5: Localization performed on the ARL propeller blade

4.5 Constrained localization

The constrained localization problem for propeller blades normally differs from the unconstrained localization problem because measured points near the leading edge of a manufactured blade have greater influence on the localization than do points in other parts of the blade. The method uses the unconstrained localization as a starting point with the implicit assumption that global minimization of distances of measured points to the design surface is achieved before the start of the constrained localization.

In contrast to unconstrained localization, measured points with non-orthogonal projections *are* used and checked in the old implementation (although those points are not used in the gradient vector). That means that results from PRAXITELES 8.0 *are* reliable. Errors like the one shown in section 4.3 are impossible. Using PRAXITELES

9.0, should create the same, correct results. Effects may be found with respect to stability and number of iteration steps, but the localization result should be the same as with PRAXITELES 8.0.

4.5.1 Examples

The first example is again the patch introduced earlier (Fig. 4-2a.). To have equivalent starting conditions for both algorithms, the point set used has been localized with PRAXITELES 9.0 first. In that case both algorithms deliver exactly the same result (Table 4.6).

	PRAXITELES 8.0	PRAXITELES 9.0
Points:	8 (of 9)	9 (of 9)
Iterations:	4	4
Results:	"Successful"	"Successful"
High, Before:	0.03143148 (3)	0.03143148 (3)
High, After:	0.00700000 (3)	0.00700000 (3)
Low, Before:	0.06841996 (6)	0.06841996 (6)
Low, After:	0.07000000 (6)	0.07000000 (6)
Tx, Ty, Tz:	$\begin{bmatrix} 0.00333820 \\ 0.00004010 \\ -0.02426456 \end{bmatrix}$	$\begin{bmatrix} 0.00333820 \\ 0.00004010 \\ -0.02426456 \end{bmatrix}$
Rx, Ry, Rz:	$\begin{bmatrix} 0.00011938 \\ -0.01501282 \\ -0.00145290 \end{bmatrix}$	$\begin{bmatrix} -0.00268015 \\ -0.00469397 \\ 0.00377787 \end{bmatrix}$
Constrained area: $u = [0 \ldots 0.1]$, $v = [0 \ldots 1]$		
High constraint: 0.007, low constraint: 0.07		

Table 4.6: Constrained localization on sample patch

Applying constrained localization on the Westinghouse blade does not show anything new either. It turns out that the unconstrained localization, which is always performed before the constrained, is much more important for the success of the constrained localization. If the unconstrained localization has been performed properly, the old and the new constrained localization algorithm lead to identical results.

Finally, for the ARL blade PRAXITELES 8.0 and PRAXITELES 9.0 lead to identical results.

4.6 Conclusion

The new localization algorithm, using the Euclidean distance, works reliably, even if some measured points do not have an orthogonal projection as the nearest point throughout the localization process. Especially in unconstrained localization the improvement is clearly visible. Misleading results are prevented successfully.

We also discovered the sensitivity of the localization to the starting position of measured points. If the footpoint of a measured points is closer to the "wrong" side of the propeller blade, there is little chance that it will "flip" over once it gets closer to the right side. This phenomenon should be explored further.

The implementation of the constrained localization in connection with the Euclidean distance works, but there is no obvious improvement. As the signed distance method also uses the Euclidean distance to verify the constraint conditions, misleading results did not occur. Nevertheless, the implementation of the improved constraint function makes sense, as this gives PRAXITELES 9.0 a consistent design.

Finally, the tangent plane distance method turns out to be fast, but it usually does not find the closest position of the point set to the design surface. The use of this algorithm might be a promising approach to speed up the localization process.

Although only very few real examples were available, the above sample cases give a good impression of the applications for the new localization algorithm, as no exceptional cases can occur any longer.

Chapter 5

Camber line extraction

5.1 Introduction

This thesis also covers the impact of manufacturing errors on the change in hydrodynamically relevant features for a propeller blade. Those features include leading edge, chord length, skew and others, as described later.

It is desirable to recompute these features for comparison with the original design data. Features such as the camber surface and the blade thickness function provide a basis for idealizations that are useful in hydrodynamic and structural dynamic analyses. The availability of feature extraction techniques is also an important part of assessing the quality of a manufactured blade with respect to its ideal design description.

The first step, in extracting features from a blade design, is generating an intersection curve with a cylinder coaxial to the propeller axis (other intersection surfaces are possible and described later). The generated intersection curve can be expanded in the plane and yields a hydrofoil-type section. This thesis deals with the problem of extracting the camber line and thickness function from such a hydrofoil section.

5.2 Problem formulation

The surface of a marine propeller is, in general, a free-form (sculptured) surface of complex geometry. A surface definition in terms of a mathematical surface provides a stable framework for automated interrogation, such as calculation of numerically controlled machine tool paths and discretization for hydrodynamic or structural dynamic analysis of propeller blades. The design process of a propeller blade is as follows: Traditionally, "the blade surface is defined in terms of sections with cylindrical surfaces coaxial with the propeller axis of rotation. Two-dimensional hydrofoil sections are firstly produced using a camber line and a thickness function. These sections are subsequently translated and rotated on their plane in an appropriate fashion and, then, the underlying plane is bent to form a cylindrical surface of a given radius, thereby transforming these hydrofoil sections to three-dimensional curves. Several sections are prepared as above on coaxial cylinders having increasing radii. The shape and position of these sections, with respect to a fixed coordinate system, changes gradually as a function of the underlying cylinder radius. Finally, the propeller blade surface is produced by lofting through all sections" [26]. Some gross geometric features [3] with important hydrodynamic function, such as camber line, section thickness function, pitch, rake, skew, chord length, maximum thickness and leading edge, govern the above procedural definition of a propeller blade.

Instead of sections lying on cylindrical surfaces, the propeller blade can be described in terms of sections lying on planar surfaces or on conical surfaces coaxial with the propeller axis. But all those methods use planar hydrofoil sections which are transformed in a similar way like above to define the three-dimensional propeller surface.

In this thesis, we are interested in the *inverse problem*, that is, to recover the above important gross geometric features characterizing the propeller blade if its surface is described as a mathematical surface. "During various design steps, such as section and surface fairing, the propeller blade surface undergoes slight modifications resulting

in small changes of the value of its features" [26]. It is desirable to recompute these features and compare them with design values. But most importantly, the availability of feature extraction techniques is an important component of assessing the quality of a manufactured blade with respect to the ideal blade specified by the designer.

We assume here, that the design surface is described as a mathematical surface (usually a B-spline patch or a NURBS patch). However, by using a faceted surface generated from measured data we can derive the features of a manufactured blade. This application is even more important, as it is desirable to compute the features of the manufactured propeller blade, to compare them with the original specifications. The first step of the procedure of generating a mathematical surface from measured data is described in [2] (and partly covered by chapter 3 and 4 of this thesis):

A set of measured space points from the manufactured propeller blade is moved to the closest overall position to the design surface (localization). To achieve a continuous representation of the manufactured surface, we first compute the projections of these localized points on the design surface. We then triangulate the projected point set in the parameter space of the design surface. With each vertex of this triangulation we associate the distance error for each corresponding 3D point after localization and we produce a piecewise linear faceted offset surface. Delaunay (or Thiessen) triangulation of the points produces a triangulation that is as equiangular as possible [23]. Now the manufactured blade is represented by the following procedural offset surface:

$$\vec{M}(u,v) = \vec{P}(u,v) + \vec{N}(u,v)h(u,v) \tag{5.1}$$

where $\vec{P}(u,v)$ is the design surface, $h(u,v)$ is the height of the faceted surface found by interpolating the height values d_i, d_j and d_k associated with the vertices of the triangle facet, whose parametric coordinates enclose (u,v), and $\vec{N}(u,v)$ is the normal of the design surface at (u,v).

Either from the design surface $\vec{P}(u,v)$ or from the procedural surface (5.1) we can

generate an intersection curve with a cylinder, plane or cone. We can develop the curve on the cylinder or cone, to get a planar curve. Now we can state the problem:

Given a planar hydrofoil-shape curve $\vec{R}(u)$, and the parameter value of the leading edge u_{LE}, find the camber line $\vec{C}(t)$ and thickness function $f(t)$, which describe the curve.

5.3 Intersection curve

The intersection curve of the propeller blade surface with a cylinder (or plane, or cone), can be found approximately by computing a series of intersection points of the two surfaces. Using the approximating technique developed in [31], we obtain a highly accurate cubic B-spline approximation of this intersection curve.

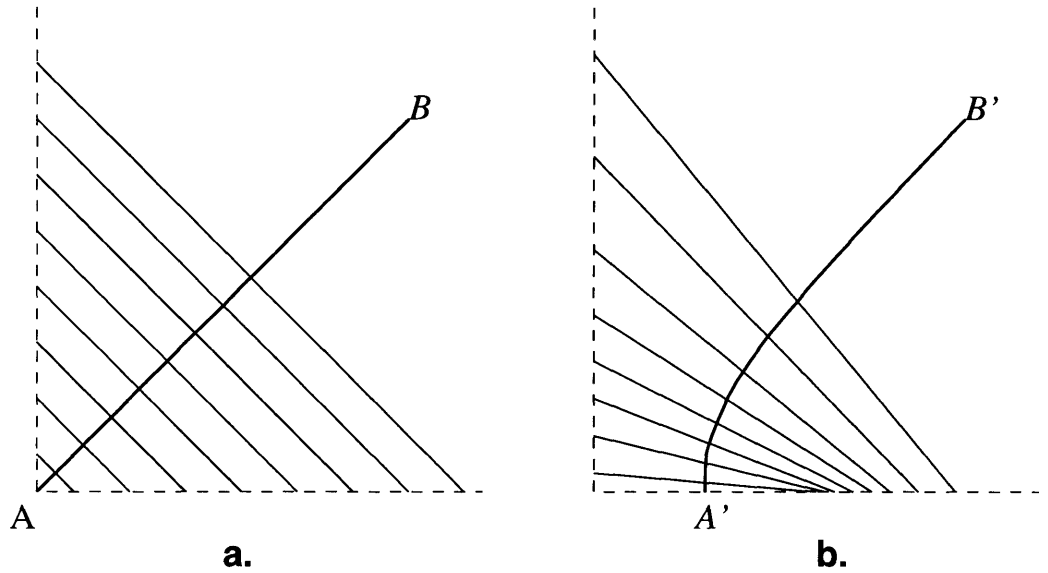
The same approximation technique can be applied on the faceted surface (5.1). Although the faceted surface has edges with C^0 continuity (on the border from one facet to the adjacent facets), the approximated intersection curve is a cubic B-spline, with C^2 continuity and given accuracy.

The details of the intersection algorithm are beyond the scope of this thesis. For the camber line extraction it is just important that we control the accuracy of the intersection curve.

5.4 Properties of Brooks ribbons

The definition of a hydrofoil section, consisting of camber line and thickness function is equivalent to the definition of a Brooks ribbon [9]. Thus, it is important to understand the characteristics of Brooks ribbons.

A Brooks ribbon is generated by a planar curve called the spine or directrix, and straight-line segments of varying length called the generators or bones, which are swept along the spine while being normal to the spine with their midpoints on the spine.



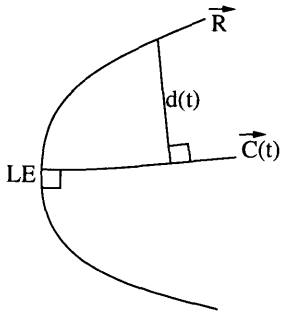
Although the generated shape is identical, the underlying Brooks ribbons are quite different. This example shows, how the starting point can significantly change the shape of the spine (and the thickness function). While A and A' are quite far away from each other, B and B' are almost identical. Within machine precision, they might even be identical. So if we pick B (or B') as a starting point, which spine do we generate? It will turn out that the algorithm developed later tends to produce a spine which is as "straight as possible".

Figure 5-1: Two Brooks ribbons, generating the same shape

5.4.1 Multiple Brooks ribbons define one shape

The shape generated by a Brooks ribbon can be identical, even if spine and thickness function are different. An example is shown in Fig. 5-1. Those ribbons are discussed in [27] in detail.

As the objective of this thesis is to generate the spine, we have to analyze the stability of the algorithm generating it. As we will see later, it works by iterating along the spine. Concerning stability, in the case of Fig. 5-1 the question would be, which spine do we generate if we use B (or B') as a starting point? Those two points might even be identical within machine precision. The preferred solution would be Fig. 5-1a. We will develop an algorithm later, which approximates the spine with straight line segments. As a straight line segment has zero curvature, the algorithm tends to find the solution with the least curvature, as in Fig. 5-1a, as desired.



The camber line C is orthogonal to the tangent of the hydrofoil curve at the leading edge (LE).

Figure 5-2: Leading edge and camber line

5.4.2 Slope of thickness function at the leading edge

As in Fig. 5-1a, the shape produced by a Brooks ribbon can have G^0 continuity at the endpoint A of the spine. In this particular example, the thickness function is a linear function, assuming an arc length parameterized spine. In our problem, we want to represent a hydrofoil section with a Brooks ribbon, where the end points are the leading and trailing edge. At least at the leading edge, the intersection curve is C^2 continuous (we approximated the intersection curve with a cubic B-spline). This is a problem for the generation of the spine and thickness function of a Brooks ribbon, particularly for the thickness function. To understand the problem, we observe the leading edge area of a hydrofoil, see Fig. 5-2. Specifically, we observe the leading edge (LE). The spine $\vec{C}(t)$ and the thickness function $d(t)$ generate the hydrofoil $\vec{R}$ by

$$\vec{R}(t) = \vec{C}(t) \pm d(t)\vec{N}(t) \tag{5.2}$$

where $\vec{N}(t)$ is the unit normal vector to $\vec{C}(t)$ at t. Considering that the thickness $d(t_{LE})$ is zero at the leading edge, we have:

$$\vec{R}'(t_{LE}) = \vec{C}'(t_{LE}) \pm d'(t_{LE})\vec{N}(t_{LE}) \tag{5.3}$$

We know that $\vec{R}$ is C^2 continuous at the LE. We observe $\vec{R}'$ at the LE. Equation

(5.3) implies two equations, one for the "upper half" of the hydrofoil curve, one for the "lower half".

$\vec{R}'$ has to be identical for the upper and lower half at LE, except for a switch in sign, which yields

$$\vec{C}'(t_{LE}) + d'(t_{LE})\vec{N}(t_{LE}) = -\vec{C}'(t_{LE}) + d'(t_{LE})\vec{N}(t_{LE}) \tag{5.4}$$

This equation can only be fulfilled either for $\vec{C}'(t_{LE}) = \vec{0}$ or $d'(t_{LE}) = \pm\infty$. We have to investigate the implications of either of those conditions.

The condition $\vec{C}'(t_{LE}) = \vec{0}$ means, that $\vec{C}(t)$ is irregular at $t = t_{LE}$.

It is however desirable that the parameter value t is approximately equal to arc length on the camber line. But

$$\vec{C}'(t) = \frac{d\vec{C}(s)}{ds} \cdot \frac{ds}{dt} \tag{5.5}$$

where s is the arc length. Therefore

$$\vec{C}'(t) = \vec{T}(t) \cdot \frac{ds}{dt} \tag{5.6}$$

where $\vec{T}(t)$ is the unit tangent vector of $\vec{C}(t)$. Therefore, it follows

$$\vec{C}'(t_{LE}) = \vec{0} \Rightarrow \frac{ds}{dt} = 0 \tag{5.7}$$

which is unacceptable and therefore we cannot fulfill $\vec{C}'(t_{LE}) = \vec{0}$.

The condition $d'(t_{LE}) = \pm\infty$ allows us to keep a uniform parameterization of $\vec{C}(t)$. Even more, it reflects the design process of a hydrofoil more realistically. Traditionally, the camber line and the hydrofoil are expressed as functions of the normalized chord. If we use a typical thickness function, like the NACA four-digit wing sections [1], we have

$$d_t(x) = \pm\frac{t_0}{0.2}(0.29690\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4) \tag{5.8}$$

where x is the parameter value equal to chord length measured from LE and t_0 the maximum thickness. If we take the derivative of (5.8):

$$\frac{\partial d_t(x)}{\partial x} = \pm\frac{t_0}{0.2}(\frac{0.14845}{\sqrt{x}} - 0.126 - 0.7032x + 0.8529x^2 - 0.4060x^3) \tag{5.9}$$

we see that

$$\lim_{x\to 0}\frac{\partial d_t(x)}{\partial x} = \pm\infty \tag{5.10}$$

An algorithm to generate the camber line has already been developed and has been implemented in PRAXITELES 8.0 [2]. This algorithm generates a camber line, parameterized according to the normalized camber line length (not to the normalized chord length). Thus, it can not fulfill $\vec{C}'(t_{LE}) = \vec{0}$. Also the thickness function is represented as a B-spline, so it cannot fulfill $d'(t_{LE}) = \pm\infty$ either. Thus, the camber line generated with PRAXITELES 8.0 is not exactly orthogonal to the intersection curve at the leading edge. From this follows, that there is a small, but measurable difference to the real camber line and/or the real thickness function. This follows also from the mechanism of that algorithm: A set of differential equations, describing the arc-length parameterized camber line, is integrated along the camber line. All the boundary conditions describing the Brooks ribbon are embedded in the differential equations. As those equations are integrated by a numerical integration scheme, ∞ for the derivative of the thickness function at the leading edge cannot be represented.

This slight difference has been discovered, because the algorithm developed later works in a different way and generates a camber line which is orthogonal to the

derivative of the intersection curve at the leading edge.

5.5 PRAXITELES

Earlier versions of PRAXITELES performed gross feature extraction from blade-like NURBS surfaces. These features have hydrodynamic importance and include: pitch, rake, skew, chord length, maximum camber, and maximum thickness [3]. Whereas previously this capability could only be applied to a *design surface*, it had been extended to extract features from a representation of the *manufactured blade* in version 8.0. The manufactured blade is represented by the procedural surface (5.1).

The hydrodynamic features are defined in relation to the camber line, the spine of a Brooks-type ribbon that is the locus of midpoints of line segments that span the intersection curve of the blade with an intersection surface, which can be a cylinder, plane or cone. The section curve is approximated by a cubic B-spline curve. PRAXITELES 8.0 uses a numerical differential equation integration scheme to find the camber for the *design surface*. This technique is no longer feasible for *measured blade surfaces*, because the integration scheme allows only one local maximum in the thickness. The intersection curve from measured design tends to oscillate slightly, and this prevents proper integration [5].

PRAXITELES 8.0 has the capability of computing the camber line for a hydrofoil generated by measured data. But it does not use the integration scheme described above. Instead, it generates a series of bisectors of matching segments from the pressure and suction side of the blade. Those straight line segments are trimmed and connected with line segments. This yields a piecewise linear camber line.

5.6 Camber line generation

The camber line of a hydrofoil section is the spine of a Brooks type ribbon, which is in accordance with the creation process of the section from camber line and thickness function [20]. The camber line, $\vec{C}(s)$, is defined as the locus of midpoints of straight line segments spanned across the section, such that the normal to the segment at its midpoint is tangent to the camber line, Fig. 5-3. The camber line is a parametric curve, parameterized with respect to its arc-length, s. If $\vec{P}\vec{R}(u)$ is one such segment in Fig. 5-4, it should be noted here that the angles formed between $\vec{P}\vec{R}(u)$ and the tangents to the section at $\vec{P}$ and $\vec{R}$ respectively, are in general not equal.

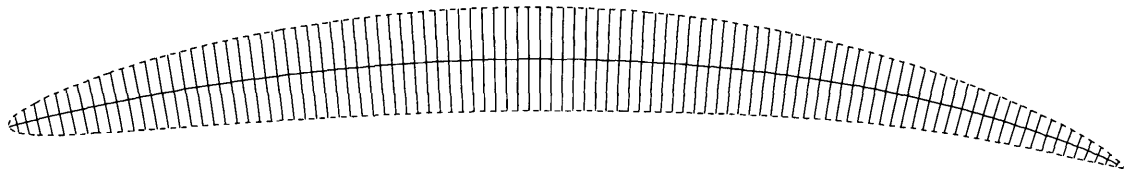
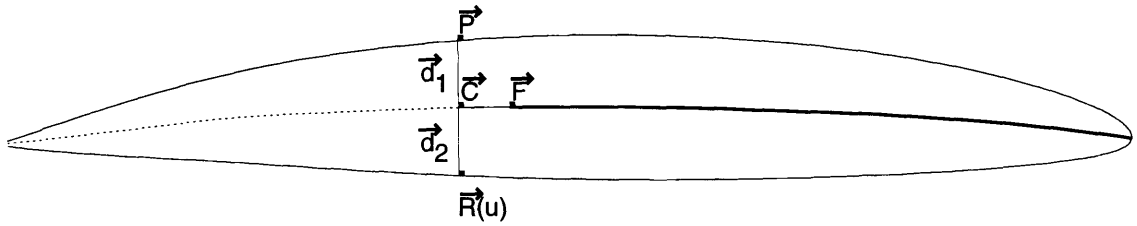


Figure 5-3: Brooks ribbon representing a hydrofoil



The camber line generation can start at any point. A useful point to start is the leading edge, as this point is exactly defined as the point with maximum curvature of the hydrofoil. Then we can step along until the camber line is complete. Assuming that $\vec{F}$ is the last generated point of the camber line, from where we want to proceed, we get the next point as follows: A point on the pressure or suction side of the blade has to be picked ($\vec{P}$), for which we look for the next point on the camber line, the point $\vec{C}$. The point $\vec{C}$ is defined by the orthogonality condition $(\vec{C}-\vec{P})\cdot(\vec{C}-\vec{F})=0$ and the condition, that $\vec{d_1}=\vec{d_2}$, which follows the definition of a Brooks ribbon. It turns out that those conditions define an equation with one unknown, the parameter value u.

Figure 5-4: Generation of the camber line

As indicated in [27], the camber line or spine of a section according to Brooks is not unique. Several spines exist depending on the choice of the initial pair of matching

points on each side of the section. Of all possible spines we are interested in the particular spine running from the leading edge to the trailing edge of the hydrofoil. Its construction is described as follows.

The camber line generation requires a starting point. Possibilities are the leading edge or the trailing edge[1]. To find the best starting point, it should be noted that the camber line extraction will be implemented in PRAXITELES 9.0. PRAXITELES represents propeller blade surfaces as open NURBS patches. The patch is open at the trailing edge (this can also be seen in Fig. 5-4, if the trailing edge is observed carefully). Thus, the hydrofoil curve is an open curve with a gap at the trailing edge. So there does not really exist a single point representing the trailing edge. The midpoint of the end points of the intersection curve is an approximation of the trailing edge, but as we do not generally have enough information about how the blade surface has been generated, we do not know whether that point really lies on the camber line. The leading edge, on the other hand, is defined as the point of maximum curvature of the planar hydrofoil curve. Picking it as the starting point, we find the camber line according to its definition.

To generate the camber line, we iterate along the pressure or suction side of the blade to the trailing edge. This is illustrated in Fig. 5-4. We interpret $\vec{P}$ as the end point of the bone of the camber line at $\vec{C}$. We also need a starting point $\vec{F}$, from where the camber line should continue (initially, that is the leading edge). During every iteration step, the starting point $\vec{F}$ is set to the last point of the camber that has been generated in the preceding step. To determine the position of $\vec{C}$, we have the following conditions:

$$(\vec{F} - \vec{C}) \cdot (\vec{P} - \vec{C}) = 0 \tag{5.11}$$

[1]It should be noted that there are other possible starting points. For example, we could also pick a point in the middle of the camber line, where the suction and pressure side of the blade are parallel. Then we could develop the camber line in both directions, to the leading edge and the trailing edge, so as to get the complete camber line.

This condition follows from the definition of a Brooks ribbon: The bone $\vec{P}\vec{R}(u)$ is orthogonal to the camber line at $\vec{C}$. As we approximate the camber line with line segments, (5.11) follows. This equation implies the unknown vector $\vec{C}$. Also:

$$\vec{P} - \vec{C} = \vec{C} - \vec{R}(u) \tag{5.12}$$

This equation follows from the fact that both segments of the bone of the camber line ($\vec{d_1}$ and $\vec{d_2}$ in Fig. 5-4) have the same length and the same orientation — they have to be identical. This equation adds another unknown u. But $\vec{C}$ is defined, as $\vec{C}$ is the midpoint of $\vec{P}$ and $\vec{R}(u)$:

$$\vec{C} = \frac{\vec{P} + \vec{R}(u)}{2} \tag{5.13}$$

So (5.11) is a function of u:

$$\left(\vec{F} - \frac{\vec{P} + \vec{R}(u)}{2}\right) \cdot \left(\vec{P} - \frac{\vec{P} + \vec{R}(u)}{2}\right) = 0 \tag{5.14}$$

This nonlinear equation with one unknown can quickly and with high accuracy be solved by the NAG routine `C05AJF`, which attempts to locate a zero of a continuous function by a continuation method using a secant iteration [24].

5.7 Accuracy analysis

With the method described, we can step from any starting point (like the leading edge of the hydrofoil curve) along the camber line. This produces a set of points on the camber line. During the camber line generation, we can also store the length of the bones to generate the thickness function at the same time.

It is desirable to process those points further, to get a smooth curve. This is possible by approximating the generated points on the camber line using a cubic B-spline. To do that, we can use the same approximation method which has been used

to generate the hydrofoil curve. We can determine the accuracy of the approximation of the cubic B-spline in relation to the given points.

But we also have to know the accuracy of those points we found on the camber line. As we iterated by assuming the camber segments as straight line segments, an error is involved, which depends primarily on the curvature of the camber line and the thickness.

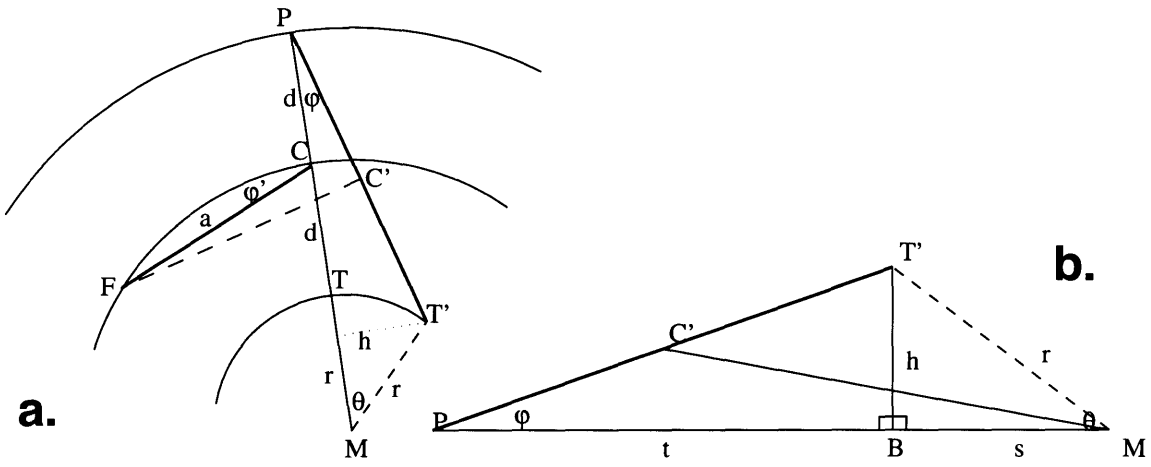
5.7.1 Error dependency

We can picture a symmetric Brooks ribbon with an appropriate starting point to get a straight spine with zero curvature. An example of such a Brooks ribbon is Fig. 5-1. Applying the algorithm described above on this shape will generate a spine with no error, no matter how the thickness function looks. The reason is, that in this case line segments represent the exact (and not approximated) spine. As the spine is exact, also the right thickness values will be reported. So it seems that the curvature of the camber line has a big impact on the accuracy of the algorithm. With curvature present in the spine, the thickness will also have an impact on the accuracy. To investigate the influence of these variables, we closely examine the following case of a spine with a constant thickness function.

5.7.2 Error in terms of spine curvature

To simplify the analysis, we observe a Brooks ribbon with a spine of constant curvature (a circular arc with radius $r + d$) and constant bone length d, see Fig. 5-5a. All points are denoted with capital letters, and the line segments with small letters. To make the figures easier to read, the vector arrows have been left out in this section. In the figure P is the fixed point, according to the algorithm described earlier. The corresponding point on the camber line would be C, but as we approximate the spine with a straight line segment a, we get C'.

We want to establish a relation between the segment length $a = \overline{FC}$, the spine



To get an accuracy estimation, we assume a Brooks ribbon with constant bone length and constant spine curvature. Using simple trigonometric relations, we can estimate the accuracy of the camber position C' and the accuracy of the bone length $\overline{PT'}$ as functions of the spine radius $r + d$, the thickness d and the stepsize a.

a. shows a circular Brooks ribbon with constant bone length. F and C are points on the spine, $\overline{PT}$ is one bone. Using the algorithm described earlier, we get $\overline{PT'}$ as the bone and C' as a point on the camber line.

b. shows the triangle $\triangle PT'M$ enlarged to show the geometric relations.

Figure 5-5: Accuracy analysis

radius $r + d$ and the thickness d. To do so, we have to establish a relation between a and the triangle $\triangle PT'M$. We approximate φ by the angle φ' formed at C by a and the tangent to the spine:

$$\varphi \approx \varphi' = \arcsin \frac{a}{2(r+d)} \tag{5.15}$$

We observe the triangle $\triangle MT'P$ closer (Fig. 5-5b.). To picture the idea, we use the angle θ as an intermediate step:

$$\tan \varphi = \frac{h}{t} \tag{5.16}$$

We can express h and t in terms of θ:

$$h = r \sin\theta \tag{5.17}$$

The segment $\overline{PM}$ is given by $r + 2d$ (see Fig. 5-5a.), so by subtracting $s = r\cos\theta$ we get the segment t:

$$t = r + 2d - r\cos\theta \tag{5.18}$$

This yields for (5.16):

$$\tan\varphi = \frac{r\sin\theta}{r + 2d - r\cos\theta}$$

We can use the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$, since θ is very small. Using those approximations, we can express θ in terms of φ:

$$\theta \approx \frac{2d\tan\varphi}{r} \tag{5.19}$$

Now we observe the triangle $\triangle BPT'$. As we know h and t from (5.17) and (5.18), we can compute the new bone length $\overline{PT'}$. Again, the trigonometric functions can be approximated as described above:

$$\begin{aligned}
\overline{PT'} &= \sqrt{t^2 + h^2} \\
&= \sqrt{(r + 2d - r\cos\theta)^2 + (r\sin\theta)^2} \\
&\approx \sqrt{(2d)^2 + (2d\tan\varphi)^2} \\
&= 2d\sqrt{1 + \tan^2\varphi} \\
&= \frac{2d}{\cos\varphi}
\end{aligned} \tag{5.20}$$

This is the bone length of the generated camber segment. The length of the real bone is $2d$, so

$$E_{bone} = |2d - \frac{2d}{\cos\varphi}| \tag{5.21}$$

is the absolute error in the thickness function.

Further on, we are looking for the error in the position of C'. The first step is finding the length $\overline{MC'}$. Again, simplifications in the trigonometric functions apply:

$$\begin{aligned} \overline{MC'} &= \sqrt{\left(\frac{h}{2}\right)^2 + \left(s + \frac{t}{2}\right)^2} \\ &= \sqrt{\left(\frac{r\sin\theta}{2}\right)^2 + \left(r\cos\theta + \frac{r + 2d - r\cos\theta}{2}\right)^2} \\ &\approx \sqrt{(d\tan\varphi)^2 + (r+d)^2} \end{aligned} \tag{5.22}$$

The correct value would be $\overline{MC} = r + d$. Thus, the sought absolute error is the difference

$$E_{\overline{MC}} = |(r+d) - \sqrt{(d\tan\varphi)^2 + (r+d)^2}| \tag{5.23}$$

We derived the errors in terms of φ. Using (5.15), we can substitute $\tan^2\varphi$. Applying basic trigonometric rules[2], we can eliminate the trigonometric functions, as

$$\tan\left(\arcsin\frac{a}{2(r+d)}\right)^2 = \frac{a^2}{4r^2 + 8rd + 4d^2 - a^2} \tag{5.24}$$

This finally yields the absolute errors for bone length and camber position as a function of a, r and d:

Error in bone thickness:

$$E_{bone} = \left|2d - 4d\sqrt{\frac{(r+d)^2}{4r^2 + 8rd + 4d^2 - a^2}}\right| \tag{5.25}$$

[2] $\tan(\arcsin(x)) = \frac{x}{\sqrt{1-x^2}}$

Error in camber:

$$E_{\overline{MC}} = \left| r + d - \sqrt{\frac{-4r^4 - 16r^3d - 24r^2d^2 + r^2a^2 - 16rd^3 + 2rda^2 - 4d^4}{-4r^2 - 8rd - 4d^2 + a^2}} \right| \quad (5.26)$$

Instead of the above absolute error, it can be desirable to find a relative measurement for the accuracy. An appropriate measurement is the chord length. The advantage of the chord length (in contrast to maximum thickness or minimum camber radius) is, that the chord length is a feature, which can be extracted at once from the intersection curve. Furthermore, the chord length is always used as a reference to describe wing sections (like the NACA series). We get the relative errors by dividing (5.25) and (5.26) by the chord length.

5.7.3 Interpretation of accuracy

In the previous section, we developed the error functions (5.25) and (5.26) to determine the error of a generated Brooks ribbon.

First we want to make sure that the approximations in the previous section are reasonable. Therefore we try to determine the range of camber curvatures and the range of maximum thickness of real hydrofoil sections.

NACA wing section properties

To determine practical values of camber curvature and thickness of hydrofoils, we consider the NACA four-digit wing sections [1, 20].

For NACA hydrofoil sections, the camber line is expressed as a function $y_c = f(x)$, where y_c is an offset to the chord line. The chord has unit length. The camber line is defined as an analytical function by:

$$y_c = \frac{m}{p^2}(2px - x^2) \text{ forward of maximum ordinate} \quad (5.27)$$

$$y_c = \frac{m}{1-p^2}\left[(1-2p)+2px-x^2\right] \text{ aft of maximum ordinate} \qquad (5.28)$$

where m is the maximum ordinate of the camber line expressed as a fraction of the chord, p is the chordwise position of the maximum ordinate, and x is measured from the leading edge along the chord of the hydrofoil. The chord is a straight line, connecting the leading edge with the trailing edge.

Observing (5.27) and (5.28) carefully, we see that those are two parabola segments, which meet at the vertices. The vertex is also the point of highest curvature of a parabola. Thus, the point of maximum curvature is $x = p$. At $x = p$ there exists a discontinuity of curvature. The "shorter" arc has the higher curvature (e.g. for $p < 0.5$ (5.27) has the higher curvature at $x = p$). As (5.27) and (5.28) are symmetric with respect to the maximum ordinate m, we observe only (5.27). The curvature of an analytic, twice differentiable explicit curve $y = f(x)$ is given by [16]:

$$\kappa = \frac{|f''(x)|}{[1+(f'(x))^2]^{3/2}}$$

We apply this on (5.27). We set $x = p$ (the point of maximum curvature), and we consider that m and p are both positive and get for the maximum curvature (after some transformations):

$$\kappa_{max} = \frac{2m}{p^2} \qquad (5.29)$$

All the information needed is encoded in the four digits of the wing section:

First digit N_1 This indicates the maximum value of the mean line or camber line ordinate in per cent of the chord. Therefore we have the relation $N_1 = 100m$.

Second digit N_2 This indicates the distance from the leading edge to the location of the maximum camber in tenths of the chord, $N_2 = 10p$. If the digit is greater than five, we use $10 - N_2$ (instead of using (5.28)) to get the right value for the

curvature.

Third and fourth digits N_{34} The last two digits indicate the section thickness in per cent of the chord, which yields $N_{34} = d$ in Fig. 5-5a.

E. g., the `NACA 2415` wing section has 2% camber at 0.4 of the chord from the leading edge and is 15% thick.

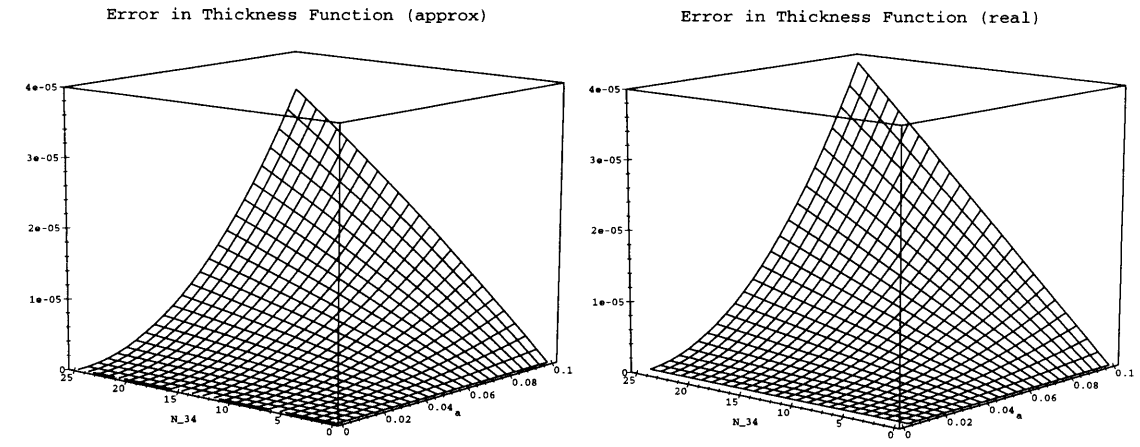
We investigate the error function on the `NACA 24xx` wing section. We modify the thickness from zero to 25% of the chord length (25% is the upper limit used in applications). The camber radius and thickness in terms of a unit chord are:

$$r_{max} = \frac{p^2}{2m} = \frac{N_2^2}{2N_1} \tag{5.30}$$

$$d_{max} = N_{34} \tag{5.31}$$

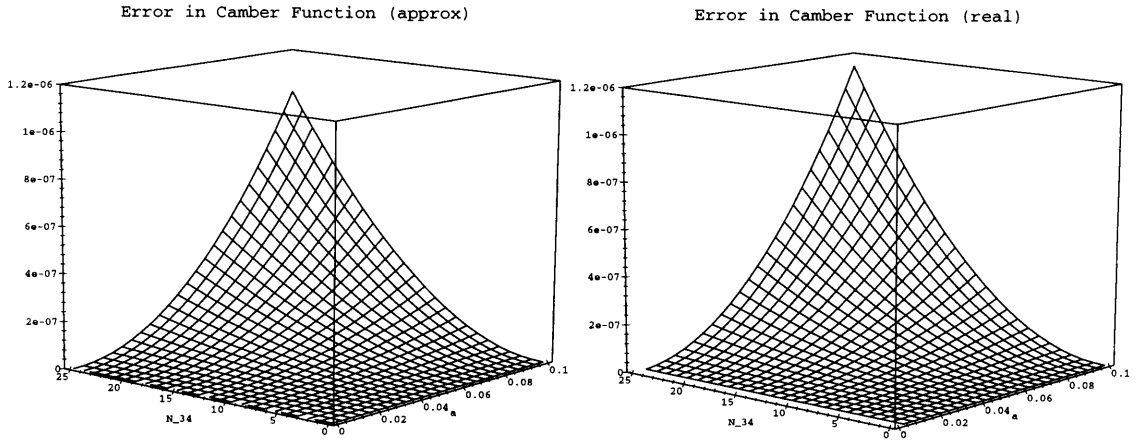
When we developed the error function E_{bone} (5.25) and $E_{\overline{MC}}$ (5.26), we frequently used the approximations $\cos\theta \approx 1$ and $\sin\theta \approx \theta$. So first we will investigate whether these approximations were appropriate. Using the software package Maple [10], it is possible also to visualize the exact error functions, *without* the approximations $\cos\theta \approx 1$ and $\sin\theta \approx \theta$. We call them "real" error function and "approximated" error function. However, the approximation $\varphi \approx \varphi'$ cannot be eliminated, as it is part of the analysis.

Fig. 5-6 and Fig. 5-7 show the error functions E_{bone} and $E_{\overline{MC}}$. The independent variables are N_{34} (the maximum thickness) and the stepsize a. N_1 and N_2 are kept constant. If the two graphs in Fig. 5-6 and Fig. 5-7 respectively would be identical, the approximation described would be perfect. But this is not the case. The larger the values of the independent variables are, the more the "approximated" error function differs from the "real" error function. It should be noted, that the approximated error function reports a better accuracy than actually achieved.



The graphs show the error in the thickness function of a NACA 24xx wing section, calculated with the error function approximation and also with the real error function, where the only approximation is $\varphi \approx \varphi'$.

Figure 5-6: Error in thickness function for NACA 24xx



The graphs show the error in the camber position of a NACA 24xx wing section, calculated with the error function approximation and also with the real error function, where the only approximation is $\varphi \approx \varphi'$.

Figure 5-7: Error in camber position for NACA 24xx

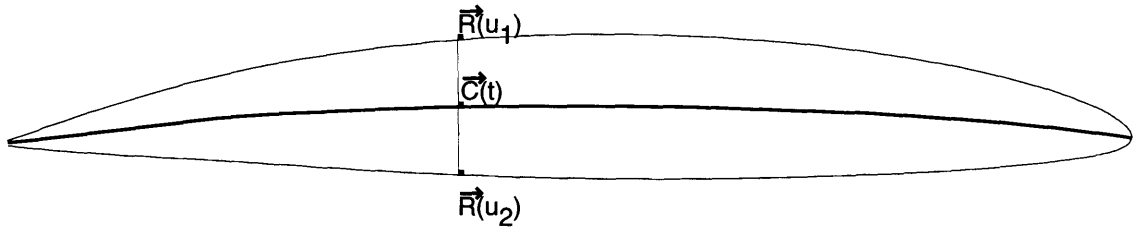
Error for an average wing section

In Fig. 5-6 and 5-7, a value of $a = 0.1$ means ten steps along the camber line, as a is the fraction of the normalized chord length. Although the results are relatively accurate,

we must not forget that the curvature is *not* constant, as assumed. Furthermore, experiments showed, that the stepsize must not be selected too small, to prevent $\vec{C}$ and $\vec{F}$ from being practically coincident (see Fig. 5-4).

5.8 Camber line refinement

The algorithm described above generates a number of points which are located approximately on the camber line. Still, numerical experiments showed, that a camber line, approximated using those points, is not yet accurate enough for practical use. Thus, an investigation was made to refine these points.



Once an approximation of the camber line is available, we can generate a number of refined points on the camber line. Picking any point on the camber $\vec{C}(t)$, we can find the corresponding points $\vec{R}(u_1)$ and $\vec{R}(u_2)$ using the orthogonality property. The midpoint of $\vec{R}(u_1)$ and $\vec{R}(u_2)$ is the refined point on the camber line.

Figure 5-8: Refinement of the camber line

Using the generated points, we can approximate the camber line with a cubic B-spline $\vec{C}(t)$ (see section 5.9 for details).

Once an approximation of the camber line is available, we can generate a new set of more accurate points. Given a parameter value t, to refine the point $\vec{C}(t)$, we find the corresponding points $\vec{R}(u_1)$ and $\vec{R}(u_2)$ on the hydrofoil curve (see Fig. 5-8). Those points are the intersection of the extended bone at $\vec{C}(t)$ with the hydrofoil curve $\vec{R}(u)$. We can generate the derivative of the spine $\vec{C}'(t)$ (as $\vec{C}(t)$ is a cubic B-spline, the derivative is available), which has to be orthogonal to $\vec{R}(u_1) - \vec{C}(t)$ and $\vec{R}(u_2) - \vec{C}(t)$ respectively, as the bone has to be orthogonal to the spine. In mathematical terms, we have to solve the following equation:

$$\vec{C}'(t) \cdot (\vec{R}(u) - \vec{C}(t)) = 0 \tag{5.32}$$

This equation has two solutions, u_1 and u_2. To solve this equation with one unknown, we again use the NAG routine `C05AJF`, which attempts to locate a zero of a continuous function by a continuation method using a secant iteration [24]. The refined point on the camber line is the midpoint of $\vec{R}(u_1)$ and $\vec{R}(u_2)$. If requested, at this time also the refined thickness may be reported, which is

$$\left| \frac{\vec{R}(u_1) + \vec{R}(u_2)}{2} \right| \tag{5.33}$$

Using this method, an arbitrary large set of points can be produced, which can be used to compute a new, better approximation of the camber line.

It turned out that this refinement achieved significant improvement, compared to the algorithm described earlier. The accuracy can be measured by recomputing the hydrofoil curve using camber line and thickness function. By comparing the original and recomputed hydrofoil, we find the error involved. We will use this method in the next chapter to check the accuracy of some examples.

5.9 Approximation

Both camber line and thickness functions are finally approximated by a cubic B-spline curve through the point set. We use the `NAG` routine `E02BAF`, which approximates by a least square method [24].

An important question was, which ratio of control points to data points should be used. If the ratio is one, we have an interpolation. But an interpolation tends to oscillate. On the other hand, an approximation generally produced an error. Still we have control over the approximation error. If the desired accuracy can not be achieved with a specific number of data points and control points, with the algorithm

described in the previous section more data points can be generated, and more control points might be used for the approximation without increasing the ratio.

In the implementation in PRAXITELES 9.0, a small ratio is picked. If the desired accuracy can not be achieved, the ratio is raised to a maximum of 0.6. If at this stage the accuracy is still not reached, the process is repeated with a higher number of data points. Experiments showed, that a maximum ratio of 0.6 safely prevents an oscillation of the approximation.

The approximations for the camber line and thickness function respectively, both follow this algorithm. Still, there are also differences.

5.9.1 Approximating the camber line

For the approximation of the camber line, the `NAG` routine `E02BAF` is not called directly, but the function `approx_cubic_curv` is used instead. This function has been developed formerly and has the big advantage, that it creates a knot vector (in this case, arc length parameterized), and it returns the maximum error.

5.9.2 Approximating the thickness function

The approximation for the camber line delivers highly accurate results. Still, for the thickness function we have a problem with the accuracy. The thickness function is a scalar function $d(t)$. The parameter t describes the position on the camber line, where the thickness is $d(t)$. The thickness function is arc length parameterized, according to the camber line.

As described in section 5.4.2, the derivative of the thickness function at the leading edge is $\pm\infty$ (under the assumption, that the derivative of the camber line equation is not zero at the leading edge). We represent the thickness function as a cubic B-spline, so the derivative of the thickness function is a polynomial. But a polynomial can never have a slope of $\pm\infty$. This means, that we can never exactly represent the thickness function correctly at the leading edge.

One way to bypass this problem would be to change the representation of the thickness function. For example, we could represent the thickness function parametrically as $d(u), t(u)$. This representation would allow to assign several thickness values d to one point on the camber line with parameter value t (e.g., $t(0.00)$ and $t(0.01)$ might both be 0, but may have the thicknesses $d(0.00) = 0$ and $d(0.01) = 0.1$. But then we do not have an explicit function $d(t)$ any more. Assuming the parametric representation, if we wanted to have the thickness at a parameter value t, we first would have to compute u (numerically), then we could compute $d(u)$.

Finally it was decided not to choose a parametric representation. The main reason for this decision was that in practice the thickness function is not processed any further. For the task of extracting gross geometric features (as listed earlier), the thickness function is not used at all, but only the camber line. For visual inspection the explicit thickness function is accurate enough, and it should be considered that the inaccuracy occurs only in the leading edge region (experiments showed, that 3%–7% of the leading edge region are out of tolerance, see next chapter). And last, the explicit representation is very convenient, as there is no need for a numerical solver, to compute the thickness at a given parameter value.

5.10 Conclusion

Two algorithms have been introduced in this chapter: An algorithm, which generates a camber line approximation with limited accuracy by approximating it with straight line segments. And a refining algorithm, which can be repeatedly applied, until the desired accuracy of the camber line is achieved. Both algorithms are highly stable and allow feature extraction from design and measured propeller blade surfaces.

Having a set of points on the camber line, we can create a B-spline approximation of the camber line. This approximation is highly accurate and the maximum error of the approximation is known.

The thickness function can also be approximated, but as a result of the representation as a scalar B-spline function, it is not very accurate in the leading edge region. It should be emphasized again that this inaccuracy affects only the thickness function, and not the camber line.

Chapter 6

Examples of camber line extraction

6.1 Introduction

The camber line generation algorithm described in the previous chapter is meant to substitute two existing algorithms implemented in PRAXITELES 8.0: The integration method [26], which is used to compute the camber line of a design hydrofoil, and the bisector method [2], to compute the camber line from measured data.

In section 5.4.2 we showed, that the camber line generated by the integration method is not accurate at the leading edge, as the camber line is not orthogonal to the tangent of the leading edge. Thus, if we want to judge the quality of camber line, it does not make sense to compare it with a camber line generated by the integration method. More extreme is the situation with the camber line generated from measured data. As it can be seen in the following, the results from the bisector methods have an accuracy which is far too low to be used as a reference.

Instead, we keep in mind that the objective is to represent a hydrofoil with a spine and a thickness function. Thus, we can regenerate a hydrofoil by using the camber line and thickness function, and compare this hydrofoil with the hydrofoil curve we

started with. Specifically, we generate a set of points on the hydrofoil by

$$\vec{P} = \vec{C}(t) \pm d(t)\vec{N}(t) \tag{6.1}$$

where $\vec{C}(t)$ is the camber line, $d(t)$ the thickness function and $\vec{N}(t)$ the unit normal of $\vec{C}(t)$ at t. Using the capabilities of PRAXITELES to find the minimum distance of a point to a curve, we can check the accuracy of the regenerated points with respect to the starting hydrofoil curve.

It should also be kept in mind that even if the camber line is highly accurate, the thickness function can not be very accurate near the leading edge, because its representation does not allow it (see section 5.4.2).

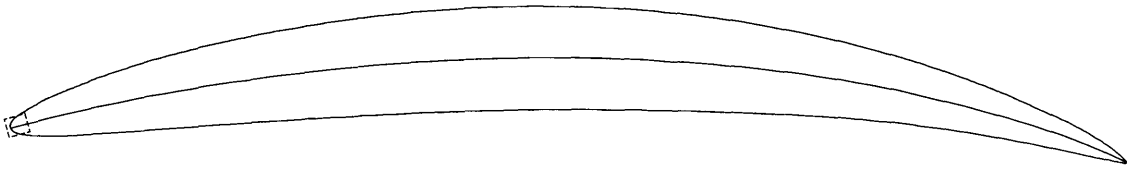
Concerning the selected accuracy of the camber line, it turns out that it does not make sense to choose an accuracy which is higher than the accuracy used to generate the hydrofoil curve by the intersection method. This occurs, because the intersection curve itself is oscillating within tolerance, so if we try to force a higher accuracy on the camber line, the camber line tries to follow this oscillation. In the following examples, the highest possible accuracy has been chosen to generate the intersection curve, and the same accuracy has been used to compute the camber line.

6.2 ARL fan blade

The following example uses the same blade described in section 4.4 and shown in Fig. 4-7. As for this blade a faceted surface is available, it is useful to show all aspects of the old and the new algorithm for computing the camber line, from design and measured data.

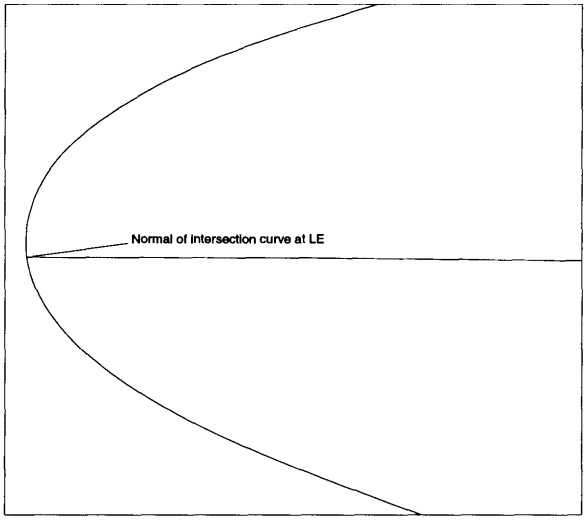
6.2.1 PRAXITELES 8.0 (from design)

Fig. 6-1 shows the intersection curve and camber line generated from design data with PRAXITELES 8.0. The picture alone does not indicate too much. But an amplification



This camber line has been generated from the ARL fan blade at a radius of 10.0 by using the integration method implemented in PRAXITELES 8.0. The selected accuracy was 10^{-4}. The camber line is not orthogonal to the hydrofoil at the leading edge. The dashed rectangle is amplified in Fig. 6-2. Thus, the camber line has a "big" error at the leading edge, which decreases towards the trailing edge.

Figure 6-1: Camber line from design generated by PRAXITELES 8.0



In this amplification of the leading edge it can be seen, that the starting direction of the camber line is visibly not orthogonal to the intersection curve in the leading edge region.

Figure 6-2: Camber line from design generated by PRAXITELES 8.0

of the leading edge region, together with the normal of the intersection curve at the leading edge, shows, that the camber line is not normal to the intersection curve (see Fig. 6-2). The error produced by the "wrong starting direction" decreases towards the trailing edge.

6.2.2 Bisector method

The old implementation of camber line extraction from measured data worked with a bisection method and delivered results, which were usable only to a very limited degree. Fig. 6-3 shows such a camber line. The hydrofoil shown here has been

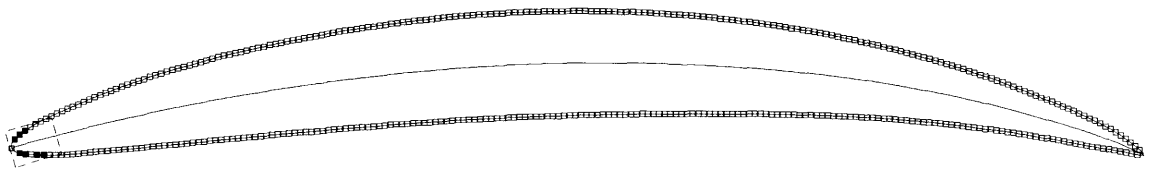
extracted from the ARL fan blade at a radius of 10.0.

This camber line has been generated from measured data with PRAXITELES 8.0. The hydrofoil has been extracted from the ARL fan blade at a radius of 10.0. The bisection method has been used at an accuracy of 10^{-4}.

Figure 6-3: Camber line generated by PRAXITELES 8.0

6.2.3 PRAXITELES 9.0 (From design surface)

The following example uses a hydrofoil section generated from the ARL fan blade, by intersection with a cylinder of radius 10.0 and at an accuracy of 10^{-4}. The intersection curve could not be generated with higher accuracy (for more information about the intersection algorithm, see [26]).



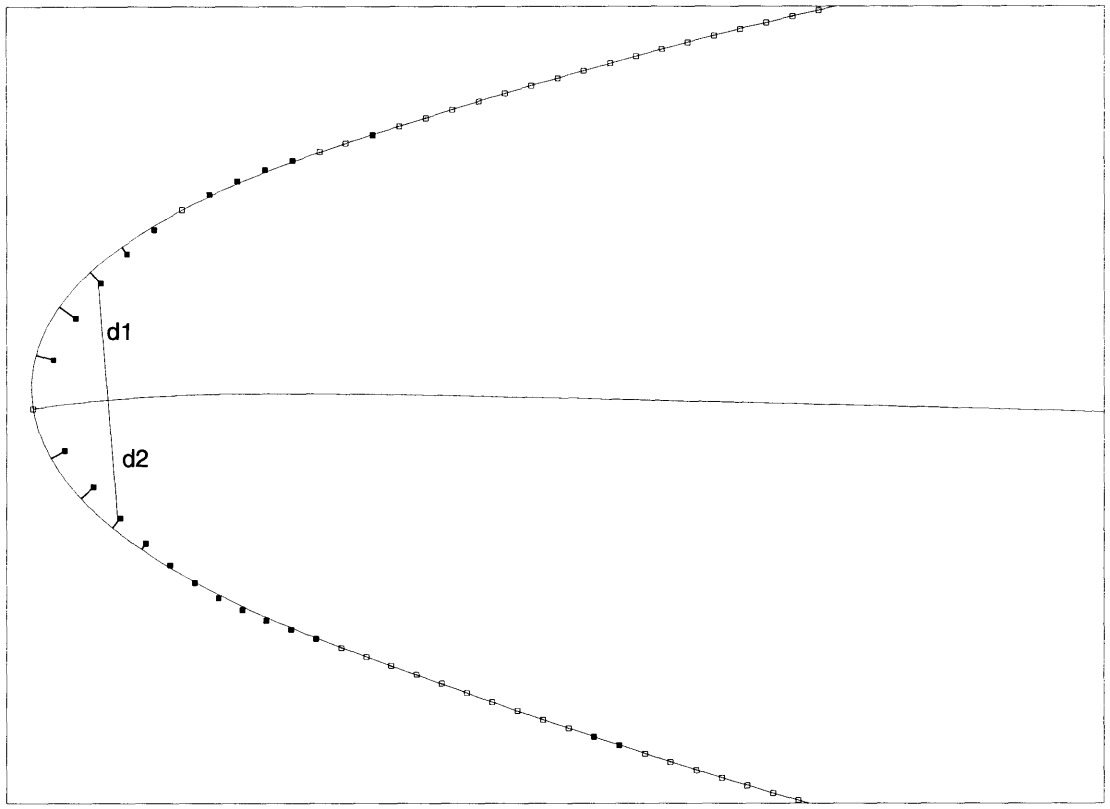
Camber line, extracted from the ARL fan blade at $r = 10.0$ and an accuracy of 10^{-4}. Hollow points represent points within tolerance, solid points without tolerance. The leading edge region (as indicated with a dashed rectangle) is amplified in Fig. 6-5.

Figure 6-4: Camber line from ARL fan blade (from design surface)

Fig. 6-4 shows the camber line itself and regenerated points on the intersection curve. The points have been generated using camber line and thickness function:

$$\vec{P} = \vec{C}(t) \pm \vec{N}(t)d(t) \tag{6.2}$$

If camber line and thickness function would be "perfect", for all $t = [0..1]$ $\vec{P}(t)$ should be located on the intersection curve. PRAXITELES has the capability to de-

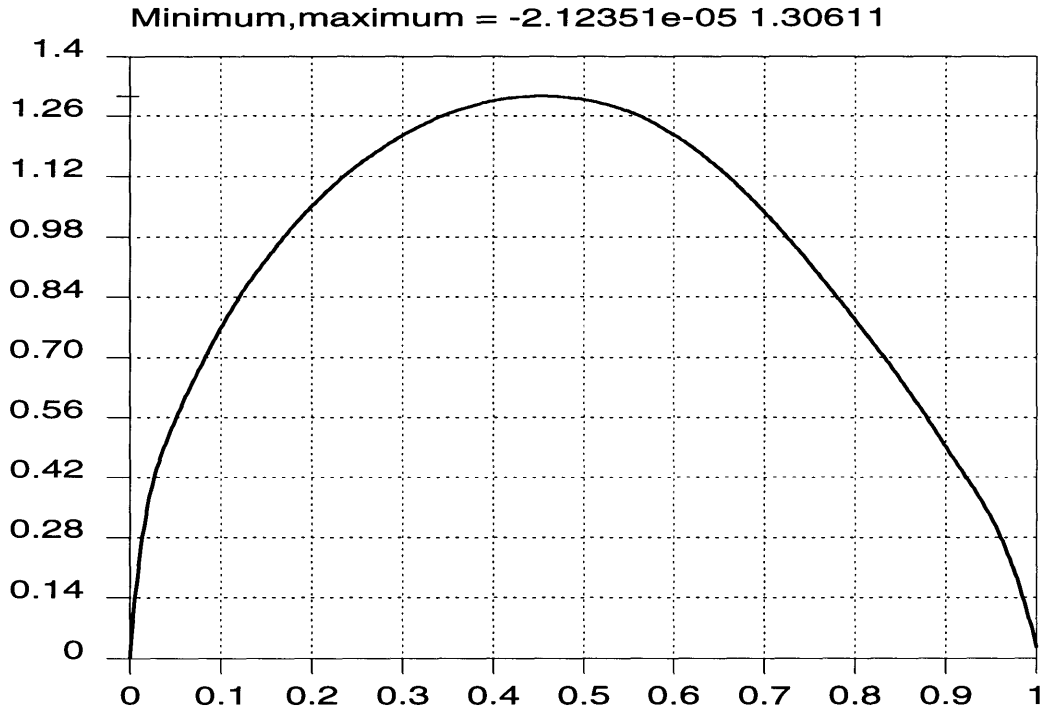


The regenerated camber line is not accurate in the leading edge region. But the error is symmetric with respect to the camber line (e.g., $|d_1 - d_2| < tolerance$). Thus, the thickness function is inaccurate, but not the camber line.

Figure 6-5: Leading edge region of ARL fan blade

termine the minimum distance of a point to a curve. In Fig. 6-4, hollow points are within tolerance, filled points are out of tolerance. Only at the leading edge points are out of tolerance. The inaccuracy at the leading edge is produced by the inaccurate thickness function. If we observe the leading edge magnified (Fig. 6-5), we notice that the error is symmetric to the camber line (e.g. two mating points of the same bone on the suction and pressure side are both either inside or outside the hydrofoil). It seems that the points are oscillating around the hydrofoil, while the amplitude of the oscillation is decreasing towards the trailing edge. The maximum error involved is 10^{-2}.

To confirm, that the camber line is accurate in the leading edge region, while the thickness produces the error, we compare the two branches of the bone (see Fig. 5-8):



Thickness function of ARL fan blade at $r = 10$. The thickness function generated from design or measured data cannot be distinguished visually.

Figure 6-6: ARL blade thickness function

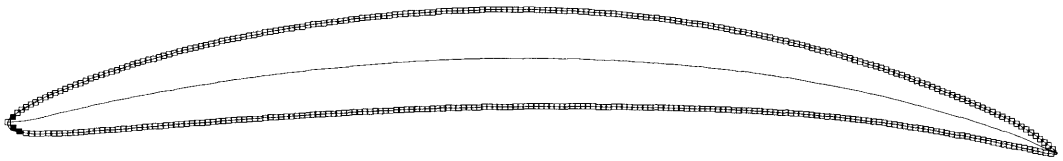
$$error = |\vec{T}(t_1) + \vec{T}(t_2) - 2\vec{C}(t)| \tag{6.3}$$

This error refers to the difference of length of the bone segments, rather than to the bone length itself. The difference stays within the given tolerance for all parameter values t, including the leading edge region. Thus, the inaccuracy is produced by the thickness function, and not by the camber line.

The thickness function alone is shown in Fig. 6-6. As the intersection curve is open at the trailing edge, the thickness is not exactly zero at the trailing edge $t = 1$. It is also obvious that the slope at the leading edge, although large, is not ∞.

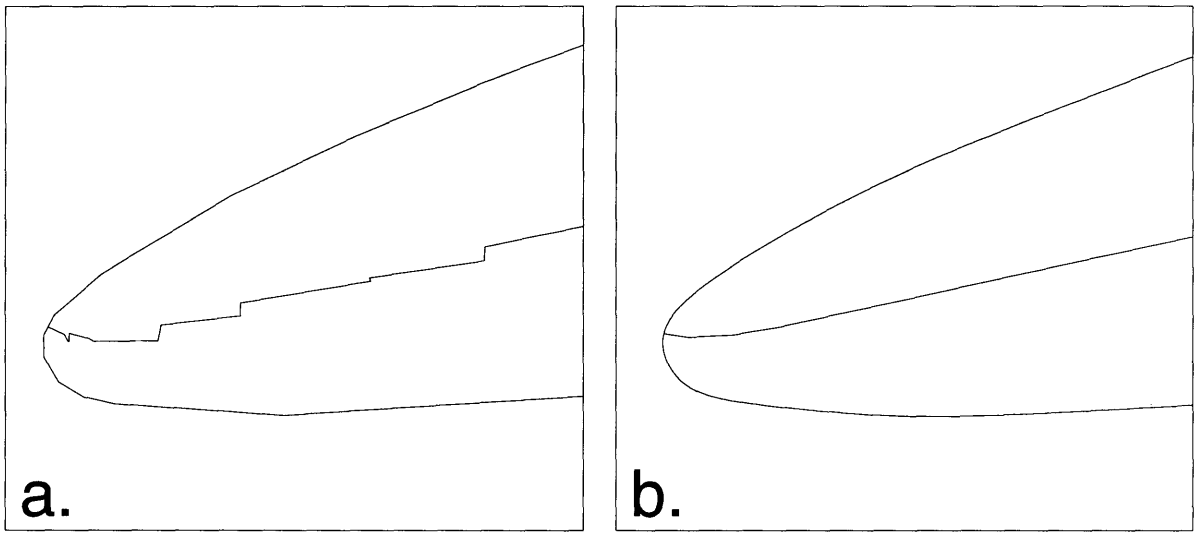
6.2.4 From measured surface

For the ARL fan blade, there is a faceted surface available. The intersection algorithm generated an intersection curve at $r = 10$ with an accuracy of $5 \cdot 10^{-3}$.



Camber line, extracted from the ARL fan blade at $r = 10.0$ and an accuracy of $5 \cdot 10^{-3}$. Hollow points represent points within tolerance, solid points out of tolerance. Note the new position of the leading edge.

Figure 6-7: Camber line from ARL fan blade, using PRAXITELES 9.0 (from measured surface)



a. Close-up of Fig. 6-3, generated with PRAXITELES 8.0. Especially in the leading edge region, the bisector method can only give a rough idea of the camber line.
b. Close-up of Fig. 6-7 from PRAXITELES 9.0. The camber line is within tolerance. The leading edge moved, as the point of maximum curvature of the intersection curve moved.

Figure 6-8: Camber lines at leading edge

The result is shown in Fig. 6-7. Again, we can observe the problem at the leading edge. It is interesting to notice that the position of the leading edge has moved, as the leading edge is defined as the point of the highest curvature of the intersection

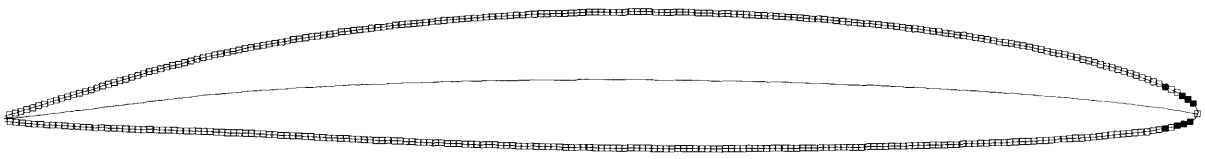
curve. This can be seen even better in the close-up, see Fig. 6-8b. The definition of the leading edge used here is only one out of several possible definitions. Another definition defines the leading and trailing edge as the points with the largest distance from each other [20]. Others are available. This example demonstrates that the used definition is probably not the best one. In this particular example, the camber line has a high curvature at the leading edge, which decreases, if we march along the camber line. But this is not the shape of a typical camber line. I.e. a camber line, as defined by the `NACA series` (see section 5.7.3) is described by a parabola with the vertex at the point of maximum camber. Its curvature is continuously increasing from the leading edge to the point of maximum camber.

Finding a better definition for the position of the leading edge might be a subject of future research.

The thickness function alone is not shown here, as it cannot be distinguished visually from Fig. 6-6. An amplification would show slight differences.

6.3 Praxiteles logo blade

For the following blade (Fig. 6-9), there were no measured points available. It is just another example of camber line generation. The intersection curve has been generated with an accuracy of 10^{-5}. Again, except a few points in the leading edge region, the approximation is accurate.

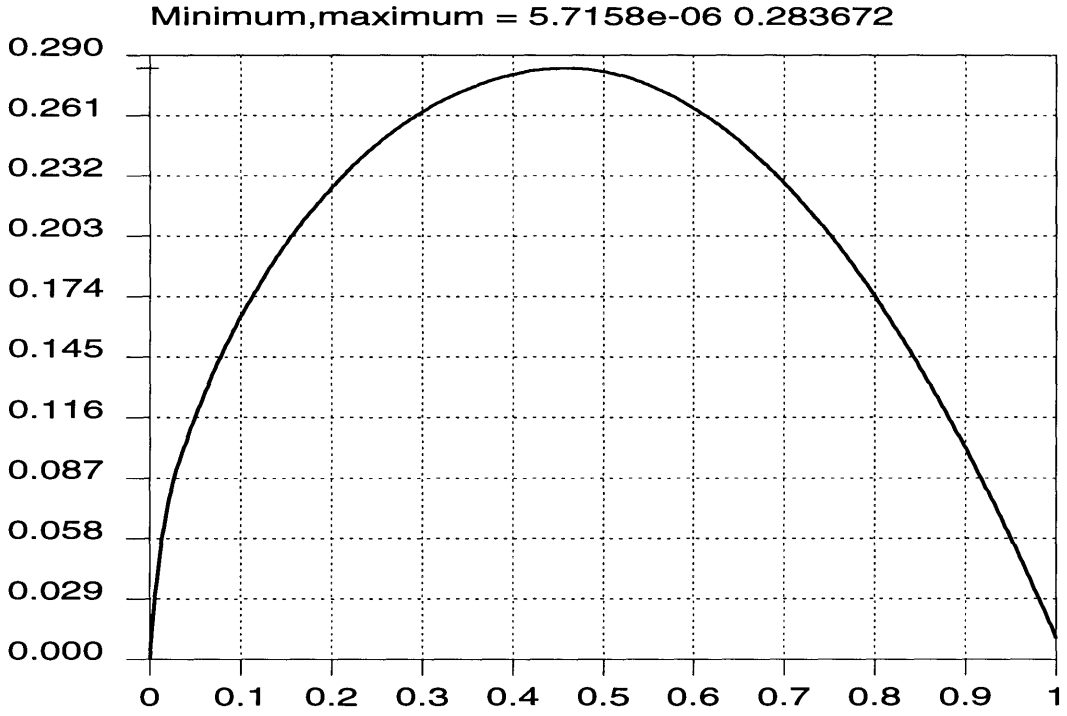


Camber line, extracted from the PRAXITELES logo blade at $r = 3.0$ and an accuracy of 10^{-5}. Hollow points represent points within tolerance, solid points are out of tolerance.

Figure 6-9: Camber line from PRAXITELES logo blade (from design)

The thickness function is shown in Fig. 6-10. A characteristic of the representation

used in PRAXITELES are a non-infinite slope at $t = 0$ and a nonzero value at $t = 1$, as the intersection curve is open at the trailing edge.



Thickness function of PRAXITELES logo blade at $r = 3$.

Figure 6-10: PRAXITELES logo blade thickness function

The algorithm has been applied on the blades shown above at different radii as well as on other blades. The algorithm worked properly and without any problems.

6.4 Conclusion

The examples show that the algorithm developed here delivers highly accurate and reliable approximations of the camber line. It also shows that the representation of the thickness function is convenient, but not accurate close to the leading edge. But as long as the thickness function is not processed any further (except for visual inspection), the current solution is sufficient. It should be noted again, that the problem is not finding an accurate value for the thickness, but approximating the points representing the thickness function with an explicit B-spline.

Another subject of future research might be an exploration of other definitions of the leading edge.

Appendix A

Additional tables

```
# Order and Number of Control Points
3 3 4 4

# Knot Vector in u and v direction
0.0 0.0 0.0 1.0 2.0 2.0 2.0
0.0 0.0 0.0 1.0 2.0 2.0 2.0

# The Control Points
0.0 0.0 0.0 1.0
0.0 1.0 0.0 1.0
0.0 2.0 0.0 1.0
0.0 3.0 0.0 1.0

1.0 0.0 0.0 1.0
1.0 1.0 0.5 1.0
1.0 2.0 0.5 1.0
1.0 3.0 0.0 1.0

2.0 0.0 0.0 1.0
2.0 1.0 0.5 1.0
2.0 2.0 0.5 1.0
2.0 3.0 0.0 1.0

3.0 0.0 0.0 1.0
3.0 1.0 0.0 1.0
3.0 2.0 0.0 1.0
3.0 3.0 0.0 1.0
```

The patch generated by this input file for PRAXITELES 8.0 and 9.0 is shown in Fig. 4-1. This simple patch is ideal to verify the localization algorithm.

Table A.1: The input file of the sample patch

0.2 0.2 0.0	1.5 0.2 0.0	2.8 0.2 0.0
0.2 1.5 0.0	1.5 1.5 0.4	2.8 1.5 0.0
0.2 2.8 0.0	1.5 2.8 0.0	2.8 2.8 0.0

The location of these data points is shown in Fig. 4-1b.

Table A.2: A set of data points

PRAXITELES 8.0
Points: 9 (of 9) from 9
Iterations: 24
Results: "Very good solution"
RMS, before: 0.06858704
after: 0.03768735 45.1%
Tx, Ty, Tz: 0.00000000 0.00000000 0.06007877
Rx, Ry, Rz: 0.00000000 0.00000000 0.00000000 Rad
PRAXITELES 9.0
Points: 9 (of 9) from 9
Iterations: 24
Results: "Very good solution"
RMS, before: 0.06858704
after: 0.03768735 45.1%
Tx, Ty, Tz: 0.00000000 0.00000000 0.06007877
Rx, Ry, Rz: 0.00000000 0.00000000 0.00000000 Rad
Iteration tolerance: 0.001

The patch is shown in Fig. 4-1. The set of measured point is shown in Table A.2. The results of PRAXITELES 8.0 and PRAXITELES 9.0 are identical. These results show that the new localization algorithm has the same capabilities as the earlier algorithm.

Table A.3: Localization result of a set of measured points with orthogonal projections on the patch

0.0 0.2 0.0	1.5 0.2 0.0	3.0 0.2 0.0
0.0 1.5 0.0	1.5 1.5 0.4	3.0 1.5 0.0
0.0 2.8 0.0	1.5 2.8 0.0	3.0 2.8 0.0

The location of these data points is shown in Fig. 4-2

Table A.4: A set of data points

Bibliography

[1] I. H. Abbott, A. E. von Doenhoff, and L. S. Stivers, Jr. Summary of airfoil data. Technical report 824, National Advisory Committee for Aeronautics, Washington DC, 1945.

[2] S. L. Abrams, L. Bardis, C. Chryssostomidis, N. M. Patrikalakis, S. T. Tuohy, F.-E. Wolter, and J. Zhou. The geometric modeling and interrogation system Praxiteles. *Journal of Ship Production*, 11(2):116–131, May 1995.

[3] D. W. Allen, A. F. Harsch, and J. D. Machin. Computer-aided marine propeller inspection data analysis. *Naval Engineer's Journal*, 107(2):33–40, March 1995.

[4] L. Bardis, R. A. Jinkerson, and N. M. Patrikalakis. Localization for automated inspection of curved surfaces. *International Journal of Offshore and Polar Engineering*, 1(3):228–234, September 1991. Errata, 2(2):160, June, 1992.

[5] L. Bardis and M. Vafiadou. Ship-hull geometry representation with b-spline surface patches. *Computer Aided Design*, 24(4):217–222, April 1992.

[6] H. Blum. Biological shape and visual science (part I). *Journal of Theoretical Biology*, 38:205–287, 1973.

[7] P. Bourdet and A. Clement. A study of optimal-criteria identification based on the small-displacement screw model. *Annals of the CIRP*, 37(1):503–506, January 1988.

[8] J. M. Brady and H. Asada. Smooth local symmetries and their implementation. *International Journal of Robotics Research*, 3(3):36–61, 1984.

[9] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[10] W. C. Bruce, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *First Leaves: A Tutorial Introduction to Maple V.* Springer-Verlag, 1992.

[11] W. Choi and T. R. Kurfess. Data localization algorithms for automated inspection. In B. J. Gilmore, D. A. Hoeltzel, S. Azarm, and H. A. Eschenhauer, editors, *Proceedings of the 19th ASME Design Automation Conference, Advances in Design Automation, Albuquerque, NM, September, 1993*, volume 2, pages 1–6. New York: ASME, 1993.

[12] P. E. Gill, S. J. Hammarling, M. A. Saunders, and M. H. Wright. User's guide for LSSOL (Version 1.0). Technical Report SOL 86-6R, Department of Operations Research, Stanford University, Palo Alto, CA, 1986.

[13] G. Goch and H. J. Renker. Efficient multi-purpose algorithm for approximation and alignment problems in coordinate measurement techniques. *Annals of the CIRP*, 39(1):553–556, 1990.

[14] K. T. Gunnarsson. *Optimal Part Localization by Data Base Matching with Sparse and Dense Data.* PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, 1987.

[15] K. T. Gunnarsson and F. B. Prinz. CAD model-based localization of parts in manufacturing. *Computer, Journal of the Computer Society of the IEEE*, 20:66–74, August 1987.

[16] F. B. Hildebrand. *Advanced Calculus for Applications.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[17] G. R. Hottel, S. T. Tuohy, P. G. Alourdas, and N. M. Patrikalakis. Praxiteles: A geometric modeling and interrogation system. In *Marine Computers '91: Proceedings of the Second Symposium on Computer Applications in the Marine Industry*, Burlington, MA, September 1991. SNAME, New England Section. Paper CC5.

[18] R. A. Jinkerson. Unconstrained and constrained localization for automated inspection of marine propellers. Engineer's thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, Cambridge, Massachusetts, 1991.

[19] R. A. Jinkerson, S. L. Abrams, L. Bardis, C. Chryssostomidis, A. Clément, N. M. Patrikalakis, and F.-E. Wolter. Inspection and feature extraction of marine propellers. *Journal of Ship Production*, 9(2):88–106, May 1993.

[20] J. E. Kerwin. *13.04 Lecture Notes: Hydrofoils and Propellers.* Massachusetts Institute of Technology, Cambridge, MA, 1990.

[21] G. A. Kriezis. *Algorithms for Rational Spline Surface Intersections.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, March 1990.

[22] G. A. Kriezis, N. M. Patrikalakis, and F.-E. Wolter. Topological and differential equation methods for surface intersections. *Computer Aided Design*, 24(1):41–55, January 1992.

[23] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

[24] Numerical Algorithms Group, Oxford, England. *NAG Fortran Library Manual, Volumes 1-8*, Mark 14 edition, 1990.

[25] N. M. Patrikalakis and L. Bardis. Localization of rational B-spline surfaces. *Engineering with Computers*, 7(4):237–252, 1991.

[26] N. M. Patrikalakis and L. Bardis. Feature extraction from B-spline marine propeller representations. *Journal of Ship Research*, 36(3):233–247, September 1992.

[27] A. Rosenfeld. Axial representations of shape. *Computer Vision, Graphics and Image Processing*, 33:156–173, 1986.

[28] K. C. Sahoo and C.-H. Meq. Localization of 3-D objects having complex sculptured surfaces using tactile sensing and surface description. *Journal of Engineering for Industry, Transactions of the ASME*, 113:85–92, February 1991.

[29] H. F. Thorne, F. B. Prinz, and H. O. K. Kirchner. Robotic inspection by database matching. Technical Report CMU-RI-TR-85-4, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1985.

[30] S. T. Tuohy and N. M. Patrikalakis. Geometric representation of marine propulsors. In *Marine Computers '91: Proceedings of the Second Symposium on Computer Applications in the Marine Industry*, Burlington, MA, September 1991. SNAME, New England Section. Paper CC4.

[31] F. E. Wolter and S. T. Tuohy. Approximation of high degree and procedural curves. *Engineering with Computers*, 8(2):61–80, 1992.