CHALMERS

Estimating wetted area of a model-hull from a set of camera images, using NURBS curves and surfaces.

M.Sc.Thesis in the Biomedical Engineering Programme

JOSE MARIA PEREZ-MACIAS MARTIN

Department of Signals and Systems
Division of Biomedical Engineering
Image Analysis Group
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2009

M.Sc.Thesis EX051/2009

Estimating wetted area of a model-hull from a set of camera images using NURBS curves and surfaces

M.Sc.Thesis EX051/2009

Department of Signals and Systems
Division of Biomedical Engineering
Image Analysis Group
Chalmers University of Technology
SE-41296 Göteborg
Sweden
Tel. +46-(0)31 772 1000

Reproservice / Department of Signals and Systems
Göteborg, Sweden 2009

Estimating wetted area of a model-hull from a set of camera images using NURBS curves and surfaces

JOSE MARIA PEREZ-MACIAS MARTIN
Department of Signals and Systems
Division of Biomedical Engineering
Image Analysis Group
Chalmers University of Technology

A mi padre, mi madre
y
mis hermanas

Abstract

This thesis proposes a semi-automatic system for estimation of the wetted surface of a hull given camera images and a digital CAD representation of the hull surface. The development of the system can be divided in four main problem areas:
The first problem concerns the waterline delineation from each of the camera images. Here I proposed an algorithm based on snakes requiring limited operator intervention.
The second problem concerns the export of CAD data from Rhinoceros into MATLAB. The method is completely implemented in MATLAB, reading NURBS (Non-Uniform Rational B-Spline) geometry from an IGES interface file.
The third problem considered is the registration of the camera images with digital representation of the model hull. This problem concerns the need to find a correspondence between the camera images and the digital hull.
The fourth and main problem concerns to the manipulation of the geometrical data. A fast algorithm is presented to back-project the waterline onto the digital hull, suitable for a real-time preview, together with a set of geometric algorithms to divide the set of NURBS surfaces and an algorithm to compute the area of the wetted surface.
Results presented using synthetic data were extremely accurate and results using field experimental data were repeatable to 2-3%. They differ from manual estimates by more, but the manual estimates are not necessarily more accurate.

Key words: NURBS, wetted surface estimation, CAD

Sammanfattning

Denna uppsats presenterar ett halvautomatiskt system för uppskattning av den våta ytan av ett fartygsskrov utifrån fotografier och en digital CAD-representation av skrovets yta.
Det första problemet har att göra med vattenlinjens kontur från varje fotografi. Här föreslås en algoritm baserad på ”snakes” som kräver ett visst mått av interaktion av operatör.
Det andra problemet har att göra med import av CAD-data till MATLAB. Metoden är helt implementerad i MATLAB och består av inläsning av NURBS- (Non-Uniform Rational B-Spline) geometri från IGES format.
Det tredje problemet som behandlas har att göra med registrering av fotografier med den digitala representationen av modellens skrov. Detta problem har att göra med behovet att hitta en överensstämmelse mellan fotografierna och CAD-modellen av skrovet.
Det fjärde och största problemet har att göra med manipuleringen av geometrisk data. En snabb algoritm presenteras för att återprojicera vattenlinjen på en 3d modell av skrovet. Detta är lämpligt för att skapa en vy i realtid. Ett antal geometriska algoritmer används för att uppdela NURBS ytorna i CAD-modellen och beräkna arean av den våt ytan.
Resultaten presenteras med hjälp av syntetisk data som är av hög precision och resultaten som använde sig av data från fältexperimentet kunde replikeras till 2- 3%. Resultaten skiljer sig mer från manuella uppskattningar, men det betyder inte nödvändigtvis att de manuella uppskattningarna är mer exakta.

Nyckelord: NURBS, uppskattning av våt yta, CAD.

Preface

This research is a contribution to the European Hydrodynamic Testing Alliance (HTA) Project. HTA is an EU FP6 (EU Framework Programme 6) Network of Excellence (EU Contract Number TNE5-CT-2006-031316). It consists of 9 different Joint Research Projects (JRP), including JPR8, "Wetted surface estimation".
Partners involved in JPR8 are MARIN (Maritime Research Institute Netherlands) from the Netherlands, MARINTEK (Norsk Marinteknisk Forkningsinstitutt AB) from Norway and SSPA Sweden AB who lead the JRP8, and the Department of Signals and Systems at Chalmers University of Technology. This research focuses on some tasks within JRP8 and has been carried out at the Department of Signals and Systems. The objective of JRP8 is to determine the wetted area of a model ship's hull from test tank camera images using computer assistance [1,2].

Contents

Abbreviations & Notation

Abbreviation / Notation	Description
WDA	Waterline Detection Algorithm
NURBS	Non-Uniform Rational B-Splines
PCM	Pin-hole camera model
CAD	Computer Aided Design
RGB	Red, Green, Blue content of an image
IGES	Initial Graphics Exchange Specification
CAM	Computer aided modelling
CAE	Computer Aided Editing
JRP	Joint Research Program

Acknowledgements

Gracias mamá, Noemí, Olga y Eduardo. Especialmente muchas gracias a mi padre, a quien dedico esta tesis, es un orgullo ser tu hijo y te tengo siempre muy dentro de mí. Gracias por darme tanto, por vuestra paciencia, dedicación, amor y todo aquello que sigo descubriendo, por estar cuando os he necesitado, por ser vosotros.

Me mustaría agradecer al Profesor Tomas Gustavsson por su confianza en mí. Durante este tiempo. he tenido el privilegio de trabajar con una de las personas más interesantes que jamás he conocido, Geoffrey Shippey. Gracias por tu paciencia, soporte, enseñanzas, soporte y dedicación a la hora de finalizar esta Tesis. Gracias a ti, he descubierto el mundo de la imagen y de la investigación. Gracias también a Artur Chodorowski, por tus comentarios y por tu tiempo, que han contribuido sin duda a la finalización de este documento. Un especial agradecimiento a Per Bergström, por tu ayuda, por recibirnos en Luleå y por todo el tiempo que has dedicado para responder a mis preguntas.

También he de agradecer a Matz Brown por su tiempo, soporte y confianza. Dr. Jan Tukker, por tus comentarios, preguntas que sin duda han contribuido al desarrollo de esta tesis. Gracias a Björn Allenström y SSPA por todo el material y por su tiempo.

Gracias a mis amigos del Erasmus y a todo el pasillo Viktor Rydberg. Mis amigos de la Universidad, he disfrutado unos de mis mejores momentos con vosotros.

English

Thanks Mom, Noemi, Olga and Eduardo. Specially Dad whom I dedicate this thesis. Thank you for giving so much, for your patience, your dedication, love, all of you that is in me yet to discover, for being there always, for being you.

I would like to thank Professor Tomas Gustavsson for your confidence in me. I had the privilege to meet one of the most interesting persons I have ever met, Geoffrey Shippey. Thanks for your patience, support, teaching and dedication to finalize this thesis. Through this thesis I found the amazing world of imaging and research. Thanks Artur Chodorowski for your valuable comments and all the time you spend with me. I must thank Chalmers for sending me to Gdańsk and Luleå. Special thanks to Per Bergström, for receiving us in Luleå, and for your time you spend answering my questions.

Thank you Mr. Matz Brown for your time, support and confidence. Dr. Jan Tukker, for your good comments, questions and confidence in my work; they undoubtedly contributed to this thesis. Thank you Björn Allenström and SSPA for all the given material and your time.

Thanks to my Erasmus friends, my corridor in Viktor Rydberg, friends from my University, I spent the best times ever with you.

Jose M. Pérez-Macías Martín
Göteborg, June 8th, 2009

Chapter 1

Introduction

1.1. Motivation

The purpose of this research was to develop a method to measure 3D surface parameters, given 2D camera images and a 3D digital description of the model ship vessel (from now on: ship's hull). It also includes image analysis techniques to trace the waterline in the camera images.
I chose this subject due to the increasing significance of NURBS (Non Uniform Rational B-Splines) [3] in the imaging field. NURBS has been used for a long time in several industries, including ship test and design, however its use has been restricted to free surface design. In recent years, its applications have become the state of the art in imaging.

1.2. Problem description

To carry out this research, a model hull was specifically designed for JRP8. The model was designed with Rhinoceros®[1] (RHINO) and subsequently built to scale. The scaling factor depends mainly on the true size of the vessel. The hull is towed in a test tank. The setting can be described as follows: the hull suspended from the mobile platform and towed through the water at different speeds. Three cameras are mounted on the platform (therefore moving with the ship) while the fourth is placed underwater at a fixed length along the tank. A set of pictures is taken for each towing speed tested (Figure 1.1). The underwater camera is the most important for waterline delineation, since it should have a complete view. However, two problems may arise. First, in some boats and at certain speeds, parts of the waterline are obscured by bubbles under the surface [2,1]. Secondly, if the shape of the vessel at a certain points is vertical or very steep, it is almost impossible to determine the precise height of the waterline; one pixel deviation might result in a large deviation of the waterline on the hull surface in these conditions. The platform-mounted cameras become important in such scenarios, so the operator can modify the waterline from each view. To take into account all scenarios, more than one camera is needed, generating a need of calibration and position calculation of the boat. Recent papers [4,5,6] deal with these issues.

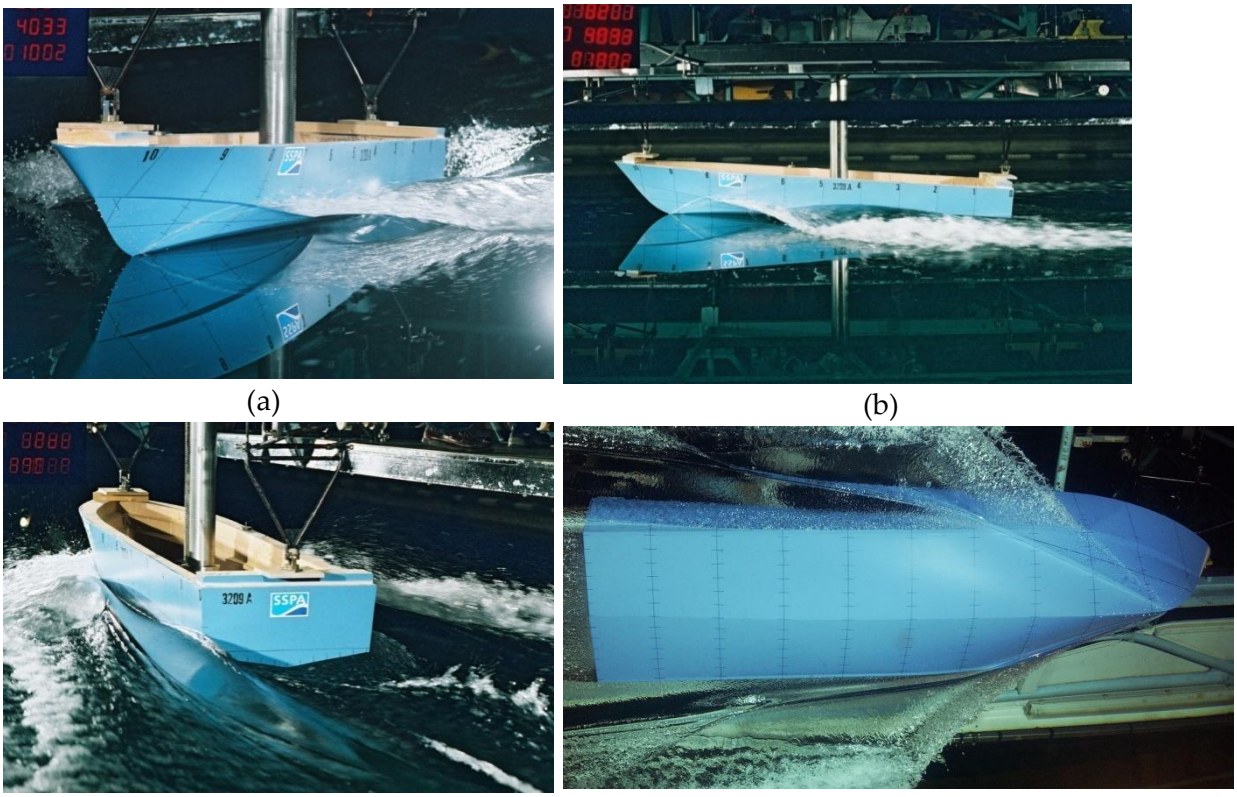
[1] CAD design software within the ship industry

The problem is to determine the wetted area of a hull using images and its digital representation. This area is necessary for calculating the resistance and propulsion data of the hull [7]. Nowadays, for certain types of ships (sailing boats, fast ships and special ship types), area calculations are made manually in two steps. First, the model hull is painted with a grid or checkerboard pattern. Then the wetted area is estimated manually using one or more camera images, in conjunction with other existing measurements. Accurate painting of equal area grid patterns on the hull is an expensive process, so it is desirable to reduce the number of markings and hence the time required to paint. Manual measurement of the pictures is also a time consuming task compared with computerized methods. It was hoped originally that a totally automatic process, using image analysis techniques to trace the waterline would save almost all the human effort. After a first study of the problem, it was realised that such a goal was unrealistic. My aim in the research was to develop an accurate semi-automatic method requiring a minimum of human effort.

My research has evolved from previous work undertaken under JPR8, summarised in section 2.2. A detailed evaluation and review is described in Chapter 4: Materials and Methods. The current work is a substantial evolution, based on the "IGES Toolbox" written in Luleå University of Technology by Per Bergström [8][9].

I propose a semi-interactive determination of the wetted surface, using any available camera images. An image analysis method has been developed to track the waterline using a snake [10,11]; a fast back-projection algorithm (BPA) to back-project the waterline onto the 3D hull suitable for real-time preview; a set of geometric algorithms to divide the surfaces by the waterline and an algorithm to compute the area of the wetted surface.

The whole method is based on NURBS using MATLAB. The digital model is designed in the Rhinoceros CAD system, exported into IGES 144 using standard Rhinoceros facilities and then imported into MATLAB. MATLAB was the chosen software for this research, due to the wide range of functions available for edge detection.



(a) (b)

(c) (d)

Figure 1.1. Views from the four different cameras at 30 knots: (a) Bow. (b) Side. (c) Stern. (d) Underwater.

1.3. Thesis Organization

Chapters 2 and 3 give a relevant background to the reader, main concepts about NURBS and the IGES file specification, computer vision and image analysis. The reader is referred to the Appendix and the cited references for further insight. Chapter 4 details the materials used in the research and gives a detailed description of the methods and models used to estimate the wetted area of the boat. Chapter 5 several tests are discussed and results presented and analyzed. In Chapter 6, I discuss the advantages and disadvantages of the choices made during the execution of research and suggest future lines of research. In Chapter 7, I summarize the work carried out and present the main conclusions.

Chapter 2

Scientific background

2.1. Introduction

This section provides general introduction of CAD, and the role of NURBS in this thesis. I start with a short review of previous work, followed by key definitions from the CAD background used in the document. I continue with some overview of meshing (a tool to discretize the surface to compute its area and plot it with the computer) and finish with an introduction to snakes.

2.2. Previous work under JRP8

This research has evolved from previous work, a waterline detection algorithm using snakes [12] and a projection method presented at the JRP8 meeting in Holland, 2007 [13].

Geometric method

A method of projecting perspective images back on to the hull surface was developed [13] using MATLAB as the tool for defining the model hull. The main disadvantage of this approach was that no member of the consortium, or probably elsewhere, uses MATLAB for ship design. Nevertheless it was hoped that the surface description of the hull could readily be exported into MATLAB. This proved not to be the case, so I began my research project with the task of reconciling RHINO software for hull design, with MATLAB software for analysis of camera images. What I took from the previous research was the idea of projecting back from the 2D camera image onto the 3D hull surface, by interpolation in parameter space.

Waterline detection algorithm

The waterline-detection algorithm [12] employed a snake energy functional, operating on the control points of a Spline. To generate the edges, the image was processed in MATLAB by a Canny edge-detection filter. The longer edges were then more heavily weighted, which discriminated successfully between the waterline edges and the bubbles. The important edge is operator-selected by choosing control points near the desired edge. What I took from this work was the idea of weighting longer edges, and also the method of moving the control points by considering the image areas above and below the Spline.

2.3. Rhinoceros

Rhinoceros [14] is a CAD software package used in ship and car body design etc. It is maintained by Robert McNeel & Associates and its native file format is called OpenNURBS. It is one of the CAD packages, but not the only one, used by HTA Consortium.

2.4. NURBS Curves and Surfaces

NURBS is the abbreviation for "Non-Uniform Rational B-Splines". It is a mathematical model for generating and representing curves and surfaces. NURBS are a generalization of both Bézier and B-Spline forms, a unified representation of all piecewise polynomial geometry. NURBS curves include both piecewise curves with any desired degree of continuity at the boundary, and also standard conic sections. The specification and use of NURBS curves and surfaces is described in textbooks such as [3,15,16]
Since 1981 when Boeing proposed NURBS as an IGES standard, it has become the de facto industry standard for the representation, design, and data exchange of geometric information processed by computer. Many national and international standards, e.g., IGES, STEP, and PHIGS, recognize NURBS as powerful tools for geometric design [3]. At the moment NURBS has evolved to T-NURBS and D-NURBS for different applications.

2.5. OpenNURBS

OpenNURBS is the native file format in Rhinoceros, used in many software applications like *Catia V5*, *SolidWorks*, *SolidEdge*, *Parasolids*. McNeel & Associates provide several tools, documentation, libraries, etc. The OpenNURBS initiative provides CAD, CAM, CAE and computer graphics software developers with the tools to transfer 3-D geometry between applications.

2.6. Export formats

A file format is a particular way to encode information for storage in a computer. Such formats simply specify how to store certain information regardless of the underlying mathematical representation e.g. Bézier surfaces, NURBS surfaces, B-Spline curves, Hermine Spline curves, etc.
The most widely used formats are: StL (Stereo-Lithography) specifies triangle meshes, DXF (Autodesk), DWF (Design Web Format from Autodesk), 3DS (3D Studio R1-R4, Lib3DS, VRML (Virtual Reality Markup Languge), IGES (Initial Graphics Exchange Specification ANSI Standard), STEP (Standard for the Exchange of Product Data ISO 10303-21), VDA-FS (Verband der Automobilindustrie - Flächenschnittstelle developed by BMW, Porsche and Volkswagen), X (Open file format owned by Microsoft related to DirectX, COLADA

(COLLAborative Design Activity for establishing an open standard Digital Asset schema for interactive 3D applications -latest release 2005) and 3DM (OpenNURBS).

2.7. IGES – Initial Graphics Exchange Specification

Initial Graphics Exchange Specification (IGES) is an independent file format, first published in January 1980, by the National Bureau of Standards at NBSIR (National Bureau of Standards Information Report) 80-1978. After the initial release of STEP (STandard for the Exchange of Product model data) in 1994, interest in further development of IGES declined and version 5.3 (1996) was the last published standard. A pre-ANSI submission review, IGES 6.0 Volume II, was published in April 2003. However it was not accepted, and there will no new IGES versions in the future, as announced by US-PRO on their web page [17]. Nevertheless IGES is still a widely used standard. Most CAD and 3D software packages can export to IGES.

2.8. IGES Toolbox

I investigated a number of ways for converting Rhinoceros data structures into MATLAB. Finally I came upon the *"IGES Toolbox"*, which provides the necessary bridge via IGES. The same toolbox would also provide a bridge from other CAD systems that can export an IGES file. This toolbox was created by Per Bergström research [18] towards his PhD at Luleå University of Technology, and is posted in *Mathworks* [8]. It uses a non-official Toolbox for MATLAB called the "*NURBS ToolBox"* [9], based on C routines for B-Spline evaluation. Per Bergström' research is concerned with accurate measurement of car body shapes using an interferometric laser system. Surface estimation requires back-projection of the laser image on to the nominal surface, so his problem has features in common with this one.

2.9. Perspective Projection and other transformations

Images are converted from 3D to 2D space through a camera. Using a pin-hole camera model (PCM) the formation of the image follows a perspective projection in which points are projected onto a common plane by following straight lines which intersect at a common point known as centre of projection. The image obtained in reality requires a more complex analysis, taking into account the distance of the object from the camera, position, rotations, clipping, distortions of the medium (air or water), aberrations of the lens, axis misalignment of the optical axis with the LCD sensor, the effect of sampling the image by sensors and inequality along x,y axes. These transformations involve a non-linear system of equations, and the computation depends on the accuracy required. However, point correspondences from the 2D image to the 3D model, gives an estimate of the required transform which is adequate for many practical situations.

References [19] and [20] give a good overview of the problem. Concerning accuracy, reference [4] gives an overview of the existing methods. Same authors propose a new camera model including other type of distortion and calculation algorithms [5,6].

2.10. Snakes (Active Contours Models)

A snake is defined as a Spline whose energy depends on its shape and location within the image. An energy-minimizing method changes the shape of the snake in order to fit to a desired image property. The energy has two components. One component is based on the distance of the curve from edges detected by a previous edge-detection algorithm. The other is based on the internal energy,i.e.the integrated torsional energy of the curve. Hence the resulting curve should be a smooth curve passing close to already detected edge sections. The important quality of the snake is that it fills in sections where no good edge has been detected [10].

Chapter 3

Geometry

3.1. Introduction

This section provides general introduction of mathematical background NURBS; the coordinate system used in the document, homogeneous coordinates, trimmed surfaces and its representation in IGES and finish with an introduction to meshing.

3.2. Curve and Surface representation

There are many ways to represent curves and surfaces. These can be grouped into closed forms and other forms.

Closed forms divide into the explicit form e.g. $y = f(x)$ and the collection $\{(x, f(x))\}$ and the implicit equation of a curve has the following form $f(x, y) = 0$ or $f(x, y, z) = 0$ for surfaces, e.g. $f(x, y) = x^2 + y^2 - 1$. There is an implicit relationship between the x and y coordinates of the points lying on the curve.

In parametric form, each of the coordinates of a point on the curve is represented separately as an explicit function of an independent parameter

$$C(u) = \big(x(u), y(u)\big) \;\; \forall \;\; a \leq u \leq b \tag{3.1}$$

Where $C(u)$ is a vector-valued function of the independent variable, u. Examples of this type of representation are the power basis form of a curve, Bézier Curves, Rational Bézier Curves, Hermite Spline curves, Tensor Product Surfaces, B-Spline Curves and Surfaces, **N**on **U**niform **R**ational **B**-**Spline**s (NURBS), etc. [16,3,21]. Some parametric curves and surfaces are defined by data points called control points. The curves pass through or are influenced by the control points. Continuity conditions are used to ensure smooth transitions between curve segments.

Another type of representation is the dataset representation, e.g. the STL format, which represents curves and surfaces as a collection of points. These are simple but computationally expensive.

The digital format of the model boat hull provided by SSPA was a Rhinoceros file, defined in its native format OpenNURBS (3dm), using curves and surfaces defined in NURBS geometry.

3.3. Coordinate systems

Coordinate system used:

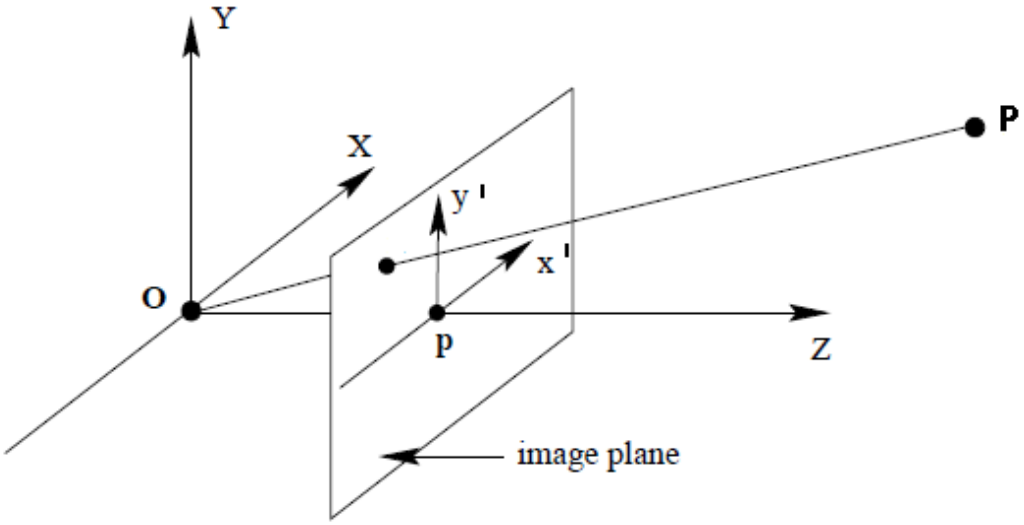


Figure 3.1 Camera coordinate system[2].

The NURBS surfaces that form the hull are represented in different domain or spaces.

- **Parameter space:** where $(u, v) \in \mathcal{R}^2$ refers to the domain of the definition of the NURBS surfaces. In the case of curves we refer with t or u always in the interval [0,1].
- **Model space**:
  - **Homogeneous coordinates:** (XW, YW, ZW, W)
  - **Non-homogenous coordinates:** $(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$
- **Image space**:
  - **Homogeneous coordinates**: (x, y, w). It is the result of the following transformation, where H is obtained in Appendix A.

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = H \begin{pmatrix} XW \\ YW \\ YW \\ W \end{pmatrix} \tag{3.2}$$

[2] Except for section 3.4. subsection "Homogeneous coordinates in NURBS" where notation was chosen to follow [3]

- **Non-homogenous coordinates:** $(x', y') = \left(\frac{x}{w}, \frac{y}{w}\right)$

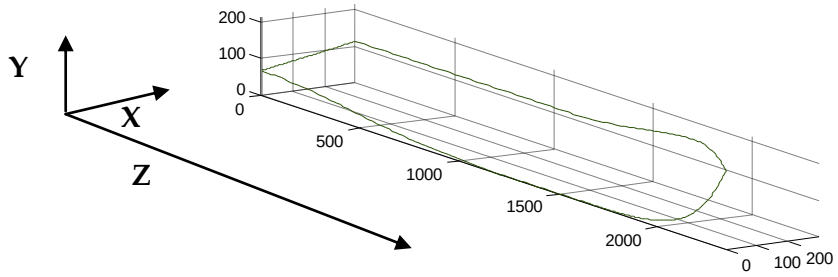


Figure 3.2 Model space.

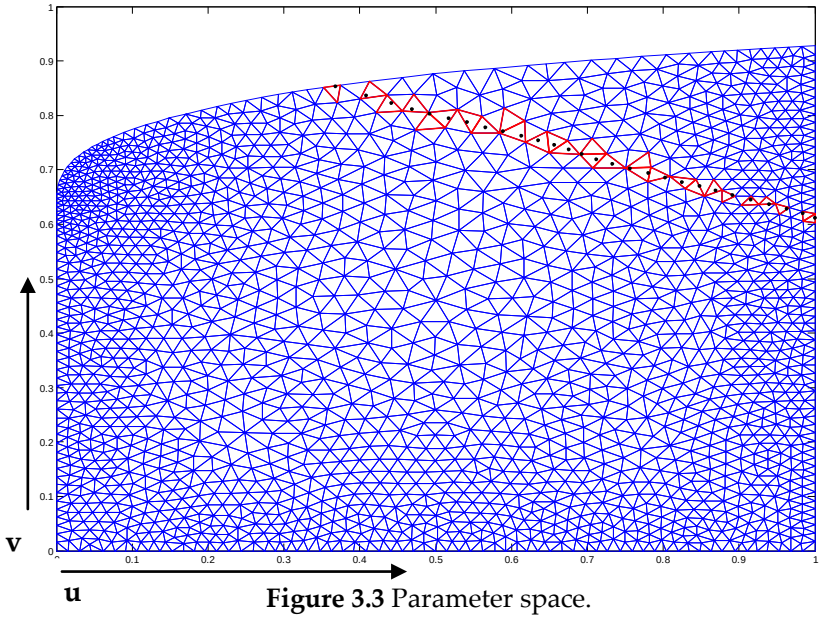


Figure 3.3 Parameter space.

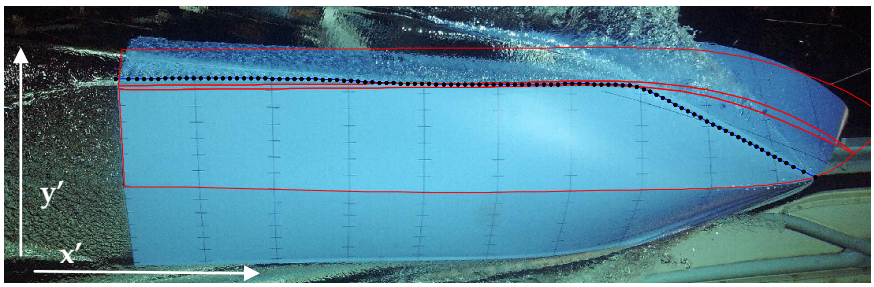


Figure 3.4 Image space.

3.3.1. Non-Uniform Rational B-Splines (NURBS)

B-Spline basis functions

NURBS uses B-Spline basis functions for curve and surface description. There are several ways to define the B-Spline basis functions. Here I will use the recurrence formula.

First, let $U = \{u_0, \dots, u_m\}$ be a non-decreasing sequence of real numbers, i.e. $u_i \leq u_{i+1}, i = 0, \dots, m-1$, where u_i are called knots, and U the knot vector. Then the *ith* B-Spline basis function of p-degree (order p+1), denoted by $N_{i,p}$, is defined as

$$N_{i,0}(u) = \begin{cases} 1, & if\ u_i \leq u < u_{i+1} \\ 0, & otherwise \end{cases} \quad \textbf{(3.3)}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \quad \textbf{(3.4)}$$

The term breakpoint requires $u_i < u_{i+1}\ \forall\ i$ while knots assumes $u_i \leq u_{i+1}\ \forall\ i$. Breakpoints correspond to the set of distinct knot values. Knot spans of nonzero length define the individual polynomial segments. Therefore, knot is used in two different ways: a distinct value (breakpoint) in the set U, and an element of the set U.

Basis functions $N_{i,p}$ are defined by the knot vector. Their purpose is to blend the control points, so they are also called blending functions. Knot vectors consist of a non-decreasing series of numbers. A position on a NURBS curve is influenced by the closest control points. Between each pair of adjacent control points there is a rational polynomial defined by intervals in the knot vector. Control points and their associated weights can be used to modify the local shape of NURBS surfaces.

3.4. NURBS curves

A p*th-degree* NURBS curve is defined by

$$\boldsymbol{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \boldsymbol{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}\ a \leq u \leq b \quad \textbf{(3.5)}$$

where the $\{\boldsymbol{P}_i\}$ are the control points, $\{w_i\}$ their weights (assumed non-negative), and $\{N_{i,p}(u)\}$ the p*th*-degree B-Spline basis functions defined on the non-periodic (and non-uniform) knot vector.

$$U = \left\{ \underbrace{0, \dots, 0}_{p+1}, u_{p+1}, \dots, u_{r-p-1}, 1, \dots, 1 \right\} \quad \textbf{(3.6)}$$

where $a = 0, b = 1$, and $w_i > 0\ \forall i$.

NURBS surfaces

A pth-degree NURBS surface is a vector-valued function of two parameters, u and v and represents a mapping of the uv plane in $\Re^2$ into Euclidean three-dimensional space $\Re^3: S(u,v) = \big(x(u,v), y(u,v), z(u,v)\big), (u,v) \in \Re^2$. There are many schemes for representing surfaces; we will follow the tensor product scheme. The basis functions are bivariate functions of u and v, constructed as products of univariate basis functions. The geometric coefficients are arranged (topologically) in a bidirectional, $n \times m$ net.

A NURBS surface of degree p in the u direction and degree q in the v direction is a bivariate vector-valued piecewise rational function of the form

$$\boldsymbol{S}(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} \boldsymbol{P}_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}}\ 0 \leq u, v \leq 1 \quad \textbf{(3.7)}$$

The $\{P_{i,j}\}$ form a birectional control net, where the $\{w_{i,j}\}$ are the weights, and the $\{N_{i,p}\}$ and $\{N_{j,q}\}$ are the nonrational B-Spline basis functions defined on the knot vectors

$$U = \left\{ \underbrace{0, \dots, 0}_{p+1}, u_{p+1}, \dots, u_{r-p-1}, 1, \dots, 1 \right\} \quad \textbf{(3.8)}$$

$$V = \left\{ \underbrace{0, \dots, 0}_{q+1}, v_{q+1}, \dots, v_{s-q-1}, 1, \dots, 1 \right\} \quad \textbf{(3.9)}$$

where p,q is the degree, r+1 and s+1 are the number of knots, n, m the number of control points. They follow the following relationship $r = n + p + 1$ and $s = m + q + 1$. A NURBS surface can be modified locally both by moving control points and by changing the associated weights. A higher weight attracts the surface more to the control point. The denominator is a normalization factor.

Properties of NURBS

There are many advantages associated with the NURBS representation, for example:

- It provides a unified mathematical basis for representing analytics shapes, such as conic sections and quadric surfaces, as well as free-form entities.
- NURBS algorithms are fast and numerically stable.
- NURBS curves and surfaces are invariant under common geometric transformations, such as translation, rotation, parallel and perspective projections.

Homogeneous coordinates in NURBS

Rational curves use homogeneous coordinates that are projected back into Euclidean space for the purpose of display. They are very important because they allow exact representation of conics such as circles and ellipsoids. Bézier and Spline curves can also be rational in form. Rational functions are defined as the ratio of two polynomials[3]

$$x(u) = \frac{X(u)}{W(u)} \text{ and } y(u) = \frac{Y(u)}{W(u)} \quad \textbf{(3.10)}$$

where $X(u), Y(u), and\ W(u)$ are polynomials. In this way, rational curves have an elegant geometric interpretation using homogeneous coordinates to represent a rational curve in n-dimensional space as a polynomial curve in (n+1) dimensional space. Therefore, a point in three-dimensional Euclidean space, $P = (x, y, z)$ is written as $P^w = (wx, wy, wz, w) = (X, Y, Z, W)$ in the four-dimensional space $\forall w \neq 0$. P is obtained from P^wdividing all coordinates by the fourth coordinate, W.

$$P = H\{P^w\} = H\{(X, Y, Z, W)\} = \begin{Bmatrix} \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}\right) \ if\ W \neq 0 \\ direction\ (X, Y, Z)\ if\ W = 0 \end{Bmatrix} \quad \textbf{(3.11)}$$

where H is a perspective transformation. Notice that for arbitrary x, y, z, w_1, w_2, where $w_1 \neq w_2$ (perspective projection invariance)

$$\begin{aligned} H\{P^{w_1}\} &= H\{(w_1 x, w_1 y, w_1 z, w_1)\} = (x, y, z) \\ &= H\{(w_2 x, w_2 y, w_2 z, w_2)\} = H\{P^{w_2}\} \end{aligned} \quad \textbf{(3.12)}$$

[3] Note that the coordinates in this section is different following Piegl notation, where P=(X,Y,Z,W) in homogeneous coordinates and P=(X/W,Y/W,Z/W) in non-homogeneous coordinates.

For a given set of control points, weights, weighted control points are constructed, $P_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$. Define the non-rational (polynomial) B-Spline curve in 4-dimensional space

$$C^w(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i^w \tag{3.13}$$

Then applying the perspective mapping, H, to $C^w(u)$ yields the corresponding rational B-Spline curve.

$$X(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i x_i, \;\; x(u) = \frac{X(u)}{W(u)} \tag{3.14}$$

The same reasoning is applied to $Y(u), Z(u)$, leads to $C(u) = (x(u), y(u), z(u))$
NURBS surfaces are defined in homogeneous coordinates as it follows.

$$S^w(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j}^w \tag{3.15}$$

where $P_{i,j}^w = (w_{i,j} x_{i,j}, w_{i,j} y_{i,j}, w_{i,j} z_{i,j}, w_{i,j})$. Then $S(u,v) = H\{S^w(u,v)\}$.

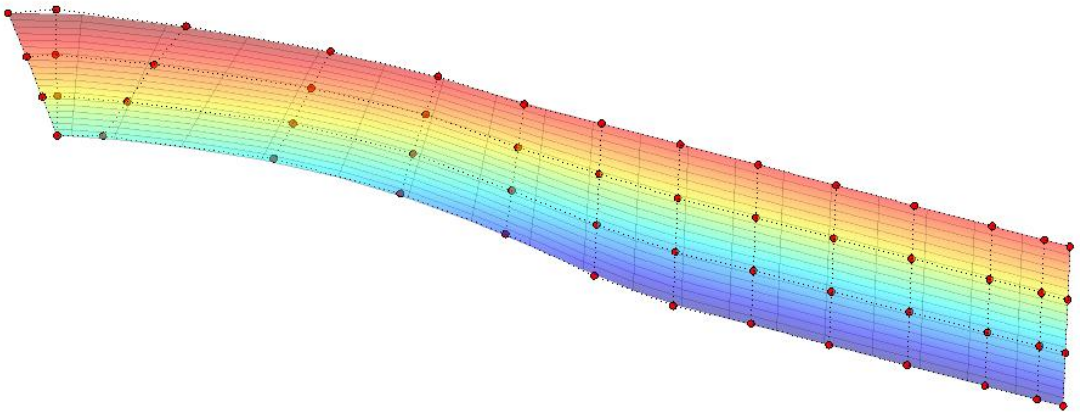


Figure 3.5 NURBS surface from the hull. In red dots, the control points of the surface.

3.5. Trimmed surfaces and their description in IGES

Most surfaces in ship design are "trimmed surfaces", i.e. surfaces bounded by NURBS curves lying in the NURBS surface, rather than by parameter limits: (u, v) where $u_{min} \leq u \leq u_{\max}$ and $v_{\min} \leq v \leq v_{\max}$. For further information read [17,8,22,9,23].

Mathematical representation

Call $C(t)$ the trimming curve resulting from the union of several trimming curves

$$C(t) = \big(x_k(t), y_k(t)\big) = \sum_{i=0}^{n} P_i N_{i,l}(t), k = 1, 2, \dots, K_1 \tag{3.16}$$

The trimmed-surface boundaries are then obtained by mapping the 2D trimming curves onto the surface [24]

$$S\big(x_k(t), y_k(t)\big), k = 1,2, \dots, N \tag{3.17}$$

After expansion, the resulting trimmed surface becomes

$$S^w(x_k(t), y_k(t)) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j}^w \tag{3.18}$$

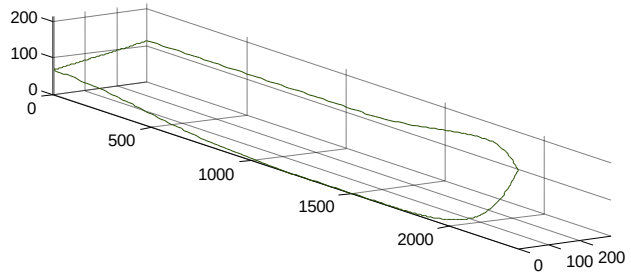


Figure 3.6. NURBS trimming curve evaluated on the surface. Here the surface is trimmed by four NURBS curves.

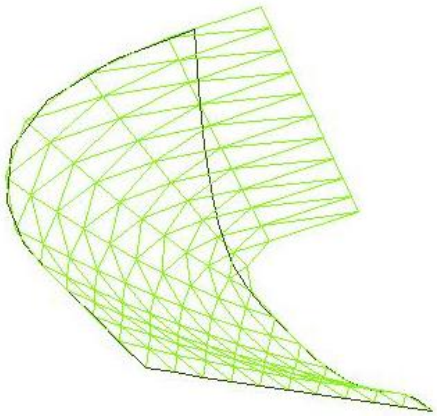


Figure 3.7. In green, the NURBS surface. In black, the NURBS curves evaluated on the surface.

Trimmed surfaces in IGES

Trimming curves can be represented in different ways. The IGES 144 convention represents these curves using NURBS, using the entities listed in Table 3.1. The IGES data structure consists of fundamental units called entities, categorized as either geometric or non-geometric. Geometric entities define the shape of objects in terms of points, curves, surfaces, solids, and collections of similarly structured entities. Non-geometric entities specify annotation, definition, and structure. They also specify attributes of entities such as colour and status, associations among entities, and flexible grouping structure that allows instancing of entity groups contained either within the file of in an external definition file.

Trimmed surfaces are obtained by mapping the 2D trimming curves onto the surface as shown in Equation (3.18). In IGES, a trimmed surface (entity type number 144) consists of a rational B-Spline surface (entity type number 128), together with the set of curves that trim the surface (entity type number 142) where each curve is a composite curve (entity type number 102) consisting of a sequence of rational B-Spline curves (entity type 126) and/or lines (entity type 110). The hierarchy of IGES is represented in Figure 3.8.

These curves are defined in IGES in two alternative ways, a NURBS curve defined in the 3D space and a NURBS curve defined in the parametric space of the surface, ensuring that the curve lies in the surface.

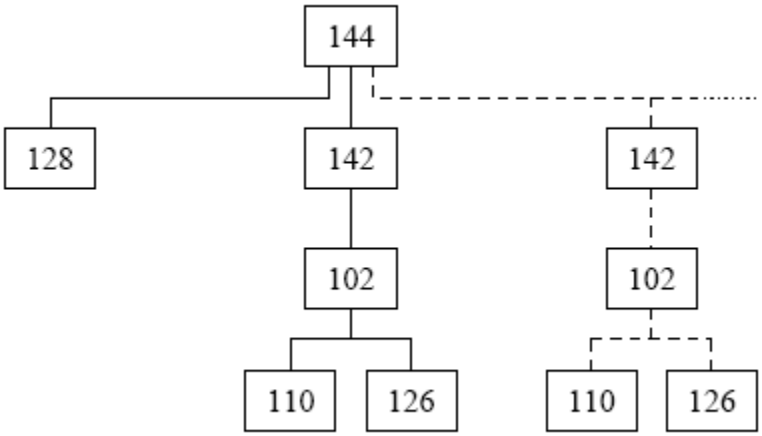


Figure 3.8. IGES entity hierarchy **[18]** The dotted line represents the NURBS trimming curves defined in the 3D space. The continuous line represents the NURBS trimming curves defined in the parametric space of the surface.

Our model hull is defined by RHINO as a set of parametric surfaces, each of which is a parametric function $S = [X(u,v), Y(u,v), Z(u,v)]$ where S takes its values in the three-dimensional Euclidean space. S is defined in $\mathfrak{D}$, whose untrimmed domain is a rectangle consisting of those the points (u,v) where a $\leq u \leq b$, c $\leq v \leq d$ for given constants a, b, c and d where $a < b$ and $c < d$. However, different surfaces do not share parameter space.
Two types of curves define the domain of the surface. The outer boundary lies in $\mathfrak{D}$, and can be the boundary curve of $\mathfrak{D}$. There can be any number of inner boundaries, including zero. I simplified the research problem by only considering outer boundaries, a simplification which was satisfactory for the model hull supplied by SSPA. Further details of the IGES specification can be found in [22].
Some conventions in the definition of trimmed surfaces [25] become important for manipulation purposes

- The rotation number of all trimming curves is $+1$, i.e. they have the same orientation.
- Each curve must be at least C^0 continuous but can contain tangent discontinuities.
- A trimming curve must not intersect another trimming curve and must not self-intersect.
- There is one trimming cure assumed to be C^0, enclosing the region that contains all the other trimming curves.
- If not explicitly defined, then the boundary of the parameter space is C^0.

TABLE 3.1. IGES 144 ENTITIES

Entity number	Entity Name	Entity number	Entity Name
102	Composite Curve entity	110	Line Entity
126	Rational B-Spline Curve Entity	116	Point entity
128	Rational B-Spline Surface Entity	314	Colour definition entity
142	Curve on a parametric Surface entity	406	Property entity
144	Trimmed (Parametric Surface entity)		

3.6. Triangulation / tessellation / meshing / reparameterisation

These are terms of different generality used in the literature for closely related operations. Smooth parametric surfaces are represented in MATLAB and other graphical software by surface rendering, either shading or colouring the surface in a way that depends on the local orientation of the surface. Surface representation, surface plotting, and surface measurements are all simplified if the surface is approximated by a mesh of small triangles or polygons. Meshing is widely used in FEM analysis and in computer graphics.
NURBS allow us to generate an infinite number of points lying on the defined surface. However points are much more useful if they are vertices of non-overlapping polygons and every point is used.
Meshing can be classified in terms of dimension:

- **2D meshing:** the mesh is generated in a two -dimensional region. e.g. a circle
- **3D meshing:** the mesh is generated in a three-dimensional volume, e.g. a sphere
- **Adaptive meshing in 2D**: this is the special case of parametric curves and surfaces. Here a 2D mesh is generated, taking into account the 3D coordinates of the nodes of the mesh (evaluation).

Meshing can be seen from different perspectives:

- We can think just of inserting nodes inside a domain defined by closed curves. Insertion of nodes depends on the size of the corresponding triangles, considering size, length of sides or the error from the real surface. This is the method of *"Triangle"* [26] and *"Mesh2D"* [27].
- An alternative way is to evaluate the NURBS surface to obtain a set of nodes that are equally spaced in 3D. This is called re-parametrization or parametrization. The mesh of triangles or polygons can be generated using the Delaunay, restrained Delaunay or Voronoi algorithms as in GGTK [28] and OpenSG [29,30].

Either way, the underlying problem is to find the parameter space values. One method solves the problem by focusing on some characteristics of the polygon in the 2D domain while the other concentrates on the 3D characteristics.

Figure 3.9 shows the nodes of the mesh for a NURBS surface. Figure 3.10 shows the plot of the triangles as a result of the triangulation process.

Meshing and Delaunay Triangulation

Suppose then nodes R^{UV} have been found, where $R^{UV} = \{u_l, v_l\}$ are L points in the parameter space defining the trimmed surface, where L depends on the density of the grid. Once an appropriate set of node points has been obtained, the next step is to form them into polygons or triangles. A well-known meshing technique is Delaunay triangulation. The Delaunay triangulation of R^{UV} has the property that for each triangle T_i, its circumcentre circle contains no other point of R^{UV} than those of this triangle. This ensures that most triangles are well shaped, avoiding obtuse angles. MATLAB provides one Delaunay triangulation. The triangulation matrix T_i, is an $i \times 3$ matrix where each row i contains the index vertices referred to R^{UV}.

Meshing error

In this research, triangles are used to back-project points from a camera image onto the model hull surface, and also to calculate surface areas. There are many existing triangulation and tessellation algorithms [31]. It is important to have a good triangulation, because it affects the accuracy of area estimation.

The surface triangulation problem is to find a set of triangles $\{T_i\}$ such that any triangle T_i satisfies $\sup\|T_i(u,v) - S(u,v)\| < \varepsilon$ where ε is the tolerance. In this research I did not mesh in terms of ε but instead used a non-adaptive meshing algorithm based on the maximum length of the side for any triangle. However, this error is possible to measure.

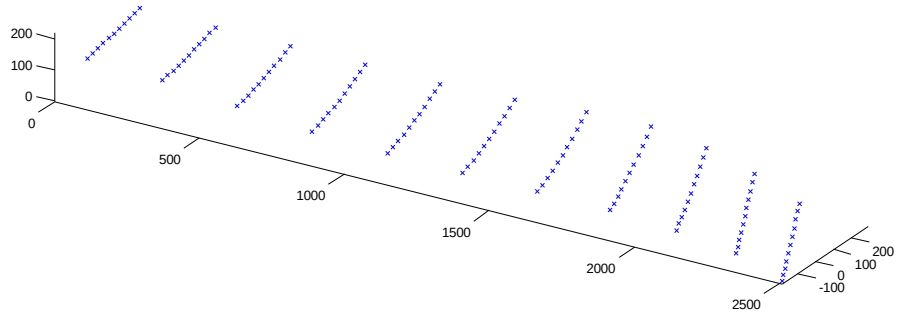
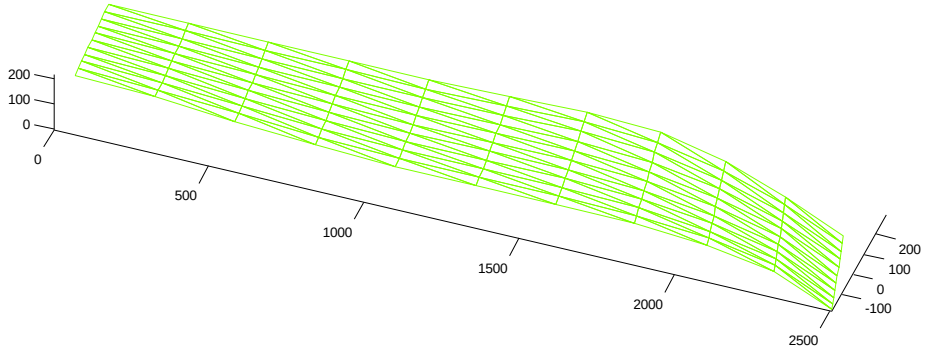


Figure 3.9. Nodes points used for the meshing of an untrimmed surface.



.

Figure 3.10. Plot of the surface using a meshing using nodes shown in Figure 3.9

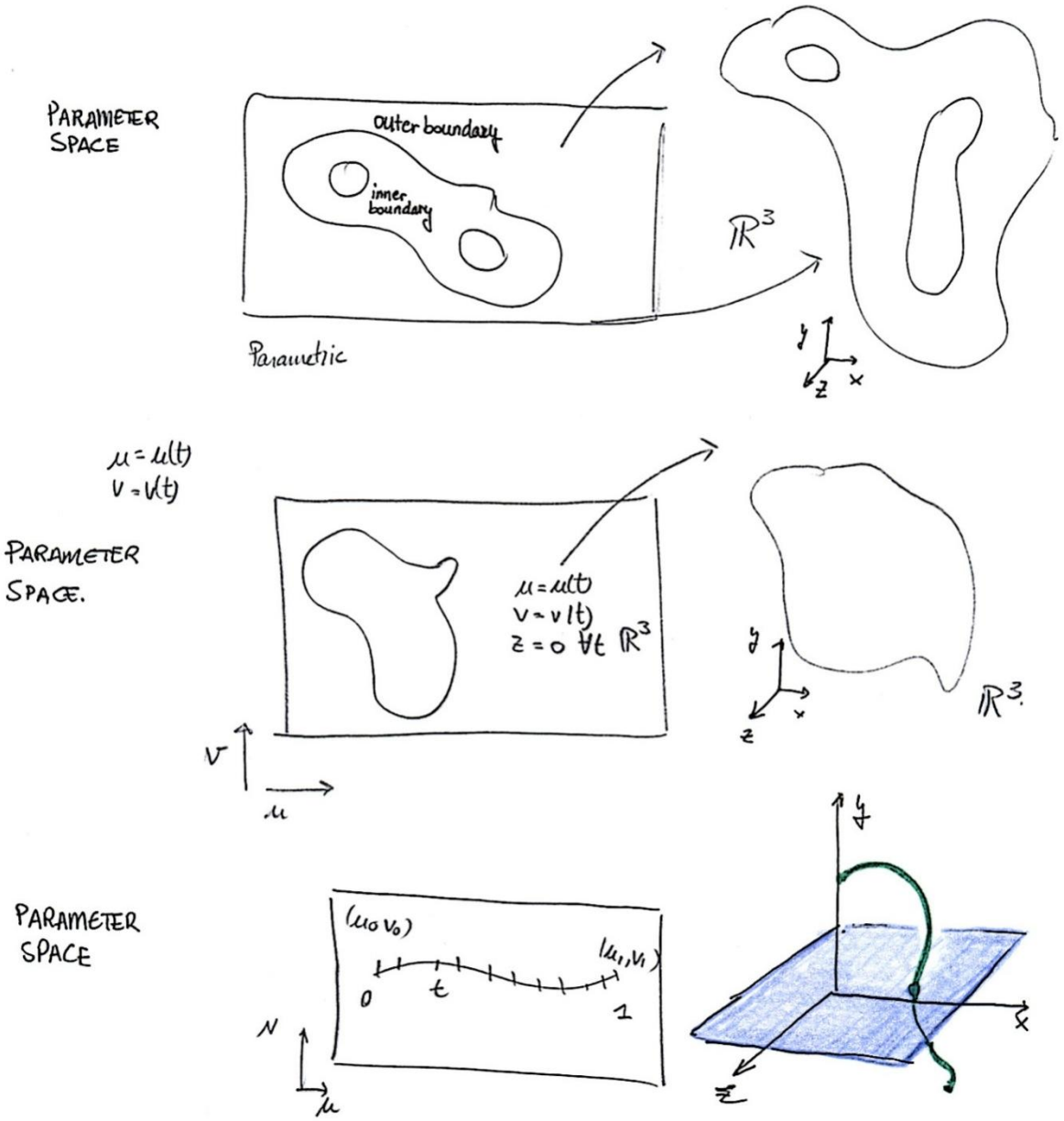


Figure 3.11. Transformation between Parameter Space and Model Space. NURBS representation of surfaces. Dual representation parameter space and model space.

Chapter 4

Materials and Methods

4.1. Introduction

This chapter describes the main research problems in some detail. Wetted surface estimation can be divided in three main tasks:

i) The image analysis task of waterline delineation in one or more camera images.
ii) Registration of camera images with the digital description of the hull surface.
iii) Back-projection of the waterline onto the hull surface, and wetted area estimation

Figure 4.9 is a flowchart showing the relationship between these tasks. The blue text indicates the inputs to the wetted surface estimate algorithm. There are three main tasks represented in three bold boxes: image analysis, image registration and geometry.

4.2. Materials and experimental arrangement

The CAD specification of the SSPA model hull is shown in Figure 4.1. The model length, width, and depth are 2.5m, 0.65m, and 0.38m respectively. At a scale of 1:7.5, this corresponds to ship dimensions of 18.75m, 4.9m, and 2.9m. The hull is defined by four NURBS surfaces, including the flat stern. One of these surfaces is very narrow, which caused significant problems with surface splitting (Section 4.8). In my computation, as in the manual computation, only half of the hull is considered, relying on a symmetric hull shape, and zero roll angle.

The experimental arrangement is as follows: The hull is towed along a test tank, suspended from the mobile platform, which can be driven at different speeds. The tank is 300 meters long by 10 meters wide and 5 meters deep. Three cameras are mounted on the platform (Figure 4.2) (therefore moving with the hull) while the fourth is placed underwater at a fixed length along the tank. It is mounted on an arm structure that can be lifted in the water (see Figure 4.3).

Test runs were carried out at four different towing speeds, providing four sets of pictures from each camera (Figure 1.1, Table 4.1). However, all pictures are not taken at the same time. The underwater camera is triggered when the platform crosses a certain trigger point along the track, which changes for different speeds. This time difference in shooting the

picture is not a problem since the tested hydrodynamic condition is static and the wave pattern seen by the platform cameras should not change during the run. However triggering errors lead to some uncertainty in the hull position seen by the underwater camera.

Table 4.1 TOWING SPEED AT WHICH THE PICTURES WERE TAKEN

knots	*m/s*
0	0
20.00	10.289
25.00	12.860
30.00	15.433

All pictures were taken with colour-sensitive cameras. The underwater camera images used here have a resolution of 3008x1547 pixels. The above-water camera images have 1840x1632 pixels resolution. Although the approximate positions of each above-water camera were given, their precise directions of view were not known. Detailed information of the cameras in shown in Appendix B.

Execution times for the algorithms were tested using an Intel Core™ 2 Duo E4600 2.2Ghz with 2Gb RAM at 667 Mhz. For this application, tests with other machines suggest that memory speed is more important than processing power. This project was developed with MATLAB 2008a and Rhinoceros 4.0.

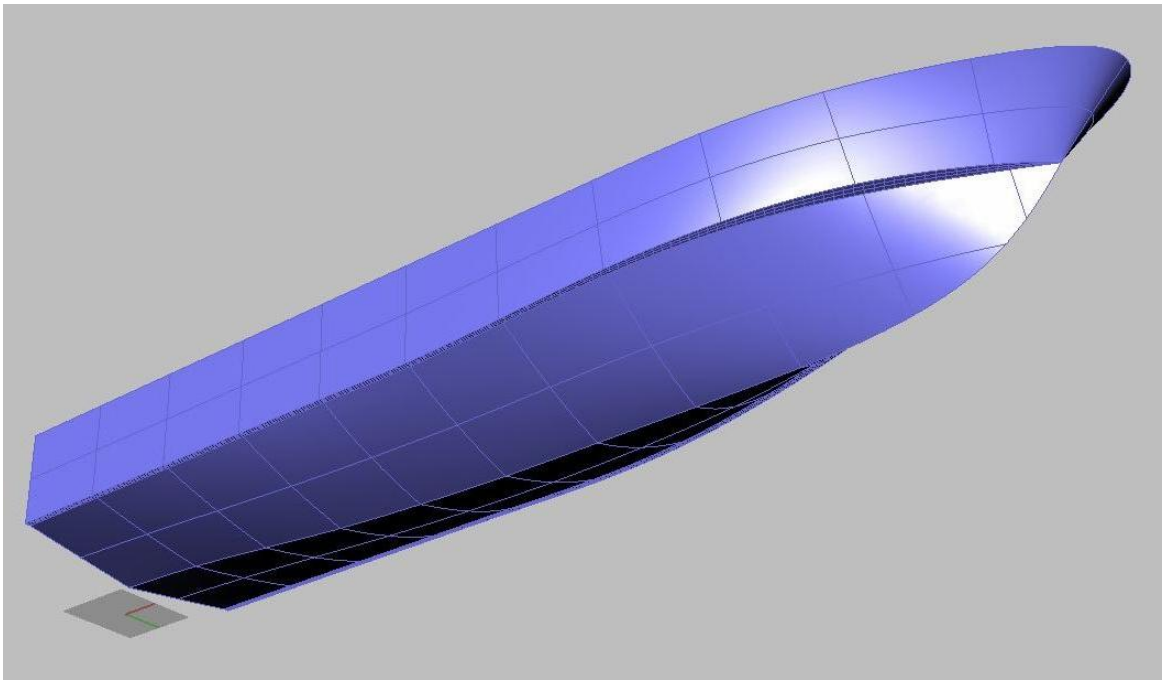


Figure 4.1 Rhinoceros specification of the model hull.

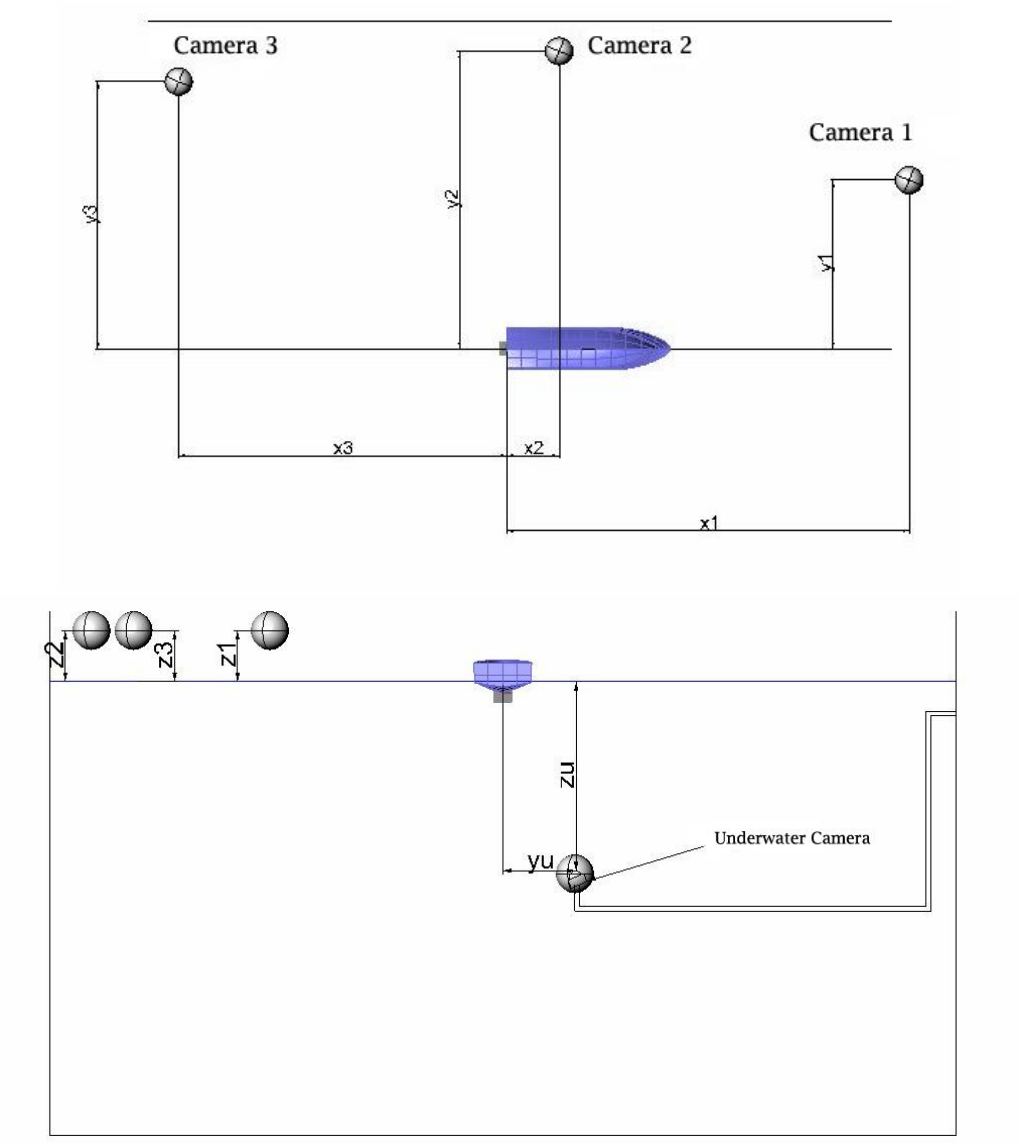


Figure 4.2. Plan and elevation of the setting

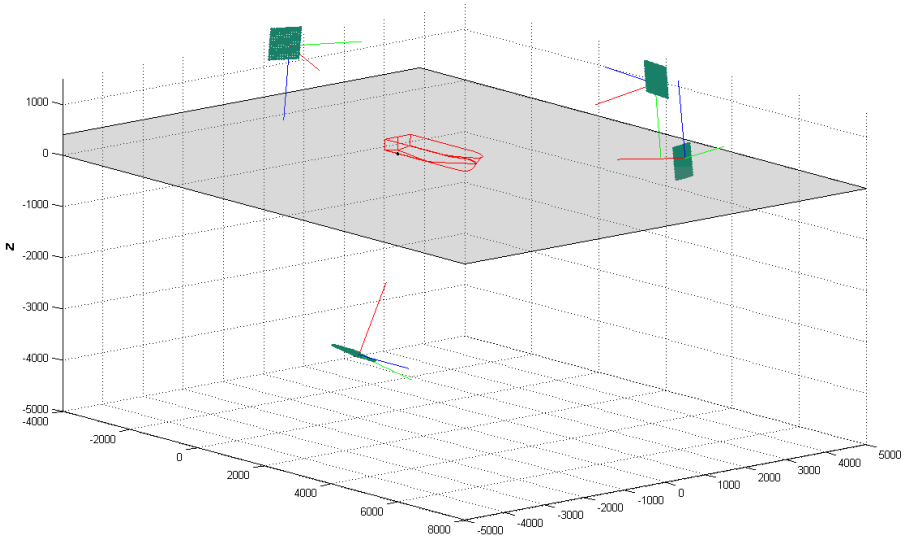


Figure 4.3. Drawing of the setting (underwater camera flipped).

4.3. Waterline delineation

***Overview*:** An image analysis stage traces the waterline with the intervention of an operator, using an edge detection algorithm to find the waterline edge where it is sufficiently clear, and a snake algorithm to fill in the gaps between these edges. The operator's intervention is essential, since there can be many edges in the image which are not the waterline. The operator marks some points where the waterline is clearly visible. My snake implementation constructs a smooth NURBS curve (apart from designated corner points) passing close to the points marked by the operator. This traced waterline can be back-projected back onto the digital description of the hull, and thence forward projected onto any other camera view.

Image representation

The acquired image is a MxNx3x8-bit unsigned-int RGB image stored in a JPG format. This is transformed into gray-scale values by forming a weighted sum of R,G, and B components.

$$I = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad \textbf{4.1}$$

Image pre-processing

As a routine step, the image was enhanced using histogram equalization, and some local smoothing applied, although the importance of these steps has not been evaluated (Figure 4.4).

Image processing

In order to prepare the image for edge detection, the image was directionally smoothed [32] (Figure 4.5) taking the local direction from the operator input. Direction filtering does two things. Firstly it produces a more uniform waterline edge, leading to better edge detection. Secondly it tends to smear out bubbles present in the image. However, here again, the importance of this step has not been evaluated quantitatively. Many more images would be required for this purpose.

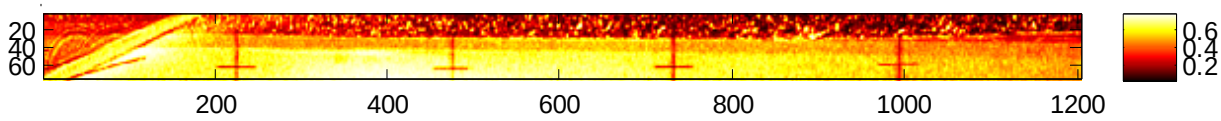


Figure 4.4 Pre-processed image.

Figure 4.5 Directionally smoothed image.

Image analysis

Edge detection

Canny edge detection was carried out on the enhanced image in a standard MATLAB operation, generating a binary image (Figure 4.6). The resulting edges were post processed. Neighbouring line elements are joined by standard morphological operations (Figure 4.7).

Higher weights are given to longer lines. The weights of lines farthest from a Spline curve through the operator input are attenuated using a mask derived from a sequence of dilation operations. This weighting converts the binary image derived from edge detection into a real-valued image (Figure 4.8).

Figure 4.6 Canny edge detection.

Figure 4.7 Line connection.

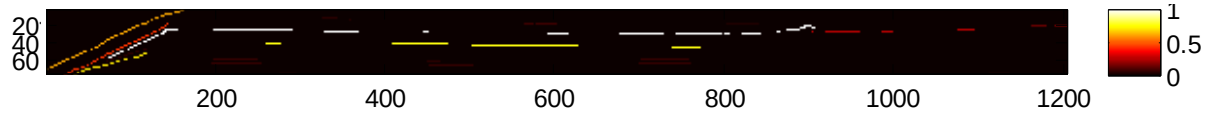


Figure 4.8 Image derived from the edge detection. White lines are strong (corresponding to long lines and closer to the operators input).

Snake

A number of different snake implementations were investigated. Finally the variational method given in [33,11] was chosen as the most robust. The pixel points returned by this algorithm are then used as a sequence of Spline control points. Discontinuities in the Spline direction (eg crossing surface boundaries) can be implemented by noting a sharp changes of direction at an operator's input point.

The result is a NURBS curve waterline, where $\bar{P}_i^w = [x_i, y_i, 0,1]$; $\overline{P_i} = [x'_i, y'_i]$ are the control points

$$C^w(u) = \sum_{i=0}^{n} N_{i,p}(u)\overline{P_i^w} \qquad \textbf{(4.2)}$$

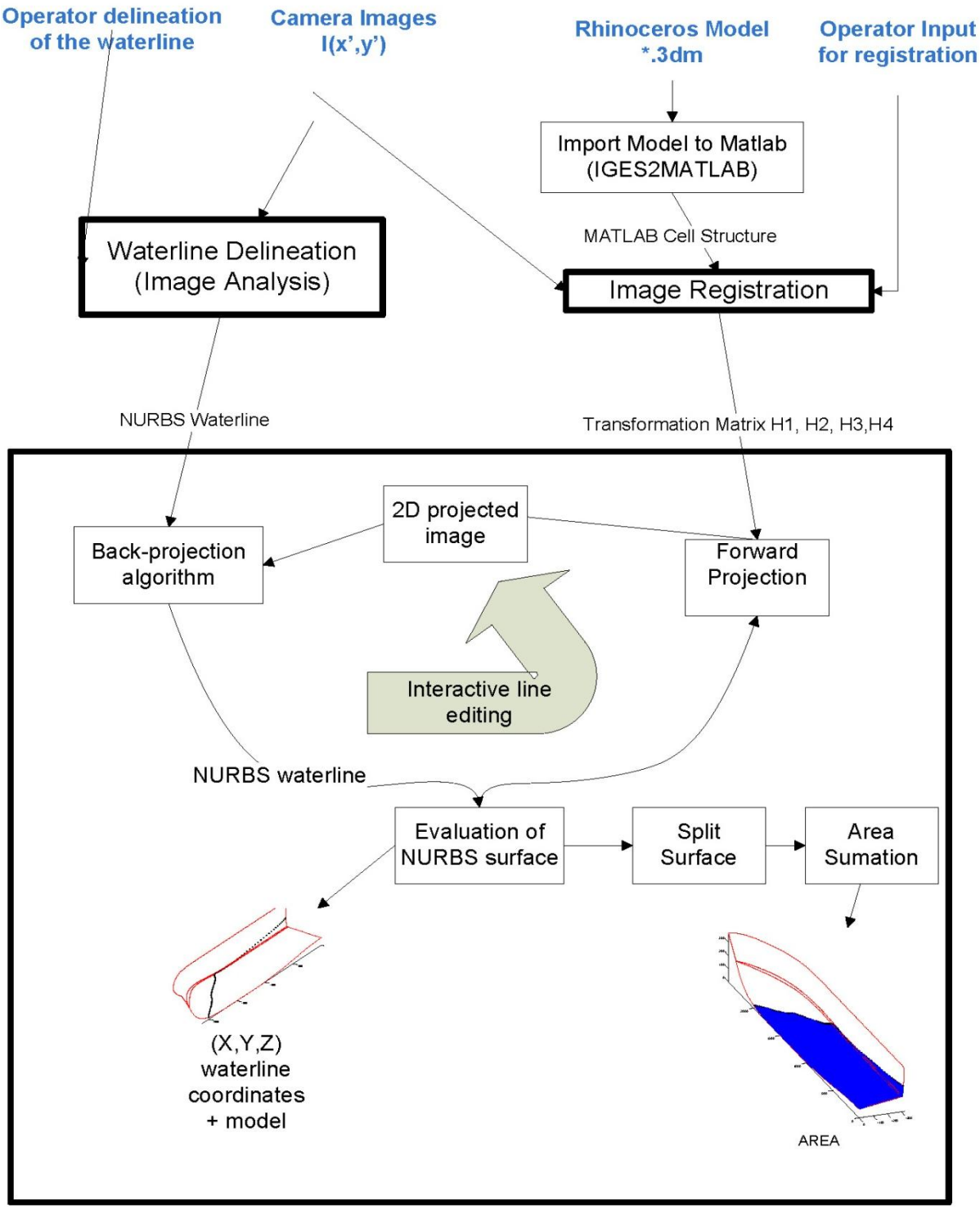


Figure 4.9. Flow overview. In blue characters, the operators input. Waterline Delineation, Image Registration and Geometry are enclosed in bold boxes.

4.4. Image registration

Image registration refers to the process of finding the correspondence between the camera images with the digital description of the hull [34]. Image registration can be carried out with each camera individually, or using a sufficient number of registration points to obtain the best collective fit for boat position, camera directions, etc. [20]. It will be shown that satisfactory registration was achieved here using the underwater camera.

4.4.1. Estimation of the perspective matrix

As can be seen in Figure 1.1, a regular set of lines is marked on the hull by SSPA to assist in manual estimation of the wetted area. Any of the small crosses in these lines can be used as registration points, since their location is given in the RHINO hull definition. A subset of these points was used to compute the perspective transformation matrix H, as described in [35] and [36]. The 3x4 matrix H defines a homogeneous Affine transformation, which is more general than a perspective transformation, requiring a minimum of 6 registration points instead of 3. However it is much easier to invert. Corresponding points in the RHINO definition were matched with points on the image manually.

4.5. Geometry

4.5.1. IGES Toolbox

This consists of three programs:

a) Convert IGES to MATLAB cell structure. The IGES2MATLAB script reads an IGES 144 file exported from Rhino and generates a MATLAB cell structure called ParameterData. This cell contains the mathematical geometric definitions representing the model hull in MATLAB. IGES2MATLAB was designed to import IGES files from the I-DEAS 3D IGES Data Translator from CATIA [18]. It also uses the unofficial "*NURBS ToolBox".*

b) **Plot script**. This evaluates NURBS functions using the NURBS Toolbox to obtain a $\Re^3$ representation and plots the result. This script included a meshing technique in order to plot surfaces.

c) **Back-projection script.** This uses a Newton-Raphson [37] iterative method, and is much too slow for our application. I propose a faster method in this chapter.

Not only do the first two programs solve many of the interface problems, but the cell structure in which IGES2MATLAB stores the geometrical data of the ship provides a flexible way to represent the complex shapes which may constitute the hull geometry. The plot script is also very useful, by meshing the surfaces using triangles. It also allows points to be tracked, checking which triangle they belong to, with simple algorithms. This proved the key to back-projection of the waterline control points.

4.6. Surface Triangulation (meshing) algorithms

The *IGES Toolbox* Plot Program [8] includes a Delaunay Triangulation to construct triangles between the inner and outer boundaries of a trimmed surface. Since the orientation of the outer curve and the inner trimming curves is predefined, it checks which triangles are out or inside the trimmed-surface domain. After sorting out which triangles are inside and which ones outside, they are subdivided in smaller triangles. Unfortunately this triangulation proved unsatisfactory when carried out on the SSPA model hull, generating long thin triangles, with almost all node points on the boundary curve.
Then I used and tested two meshing techniques "*mesh2dv23"* [27] and "*triangle"* [26], within my program. I chose *mesh2dv23* for the meshing. Other authors [29] used an adaptive meshing technique within the *OpenSG* framework using the OpenGL library [30].

4.7. Forward-projection and back-projection

4.7.1. Overview

The digital representation of the model hull can be projected onto any available camera image transforming the control points of the NURBS curves and surfaces followed by its evaluation. This is certainly necessary for image registration where the registration points need to be projected onto each camera image for comparison with the registration points seen by the operator. However, forward-projection is also used in viewing the waterline traced in one image from another viewpoint.
The operator does not see the most important aspect of forward-projection. The triangulation mesh defining the surface is projected onto each camera image, so that the NURBS control points defining the waterline can be back-projected onto the hull surface. Back-projection of the computer-generated waterline is achieved by back-projecting the NURBS curve onto the surface of the hull.

These concepts are presented more rigorously below:

4.7.2. Forward and Back-projection

Forward and back-projection algorithms proceed from a number of assumptions:

1. Given a NURBS-waterline (Eq 4.2).
2. Given H, the transformation matrix from the registration stage.
3. Given the hull, defined by a collection of K mathematical surfaces $S_k, k = 1 \dots K$

$$S_k^w(u_l, v_l) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u_l) N_{j,q}(v_l) P_{i,j}^w \qquad \textbf{(4.3)}$$

If the surface is trimmed then $C_n(t),\ n = 1 \dots N$, the resulting trimming curve is the union of a sequence of N trimming curves represented in NURBS:

$$C_n(t) = \big(u_n(t), v_n(t)\big) = \sum_{i=0}^{p} P_i^k N_{i,l}(t), n = 1, 2, \dots, N \quad \textbf{(4.4)}$$

4. Given a NURBS surfaces $S_k(u, v)$, with the trimming boundaries $C_n(t)$ the meshing algorithm generates a list of node points, R_k^{UV}, and a triangulation matrix, T^k, specifying which nodes belong to each triangle in the mesh.

4.7.3. Forward projection

Overview: Here, forward projection means the perspective projection of 3D curves and surfaces of the model hull onto a 2D camera image.

Description: NURBS surfaces are evaluated to obtain a 3D hull boundary in $\Re^3$. Given the set of surfaces $\boldsymbol{S_k}(u, v)$ or inside the resultant domain from the intersection of trimming curve and the surface, each surface is meshed (Section 4.6) to obtain a triangulation matrix T^k and the nodes matrix of the mesh for each surface $R_k^{UV} = (u_l, v_l)$ (in parameter space) for each surface.

A 3D NURBS surface is projected into a 2D image by projecting the 3D control points into 2D image points, multiplying the control points $P_{i,j}^w$ by H [3x4], to obtain $\bar{P}_{i,j}^w$ (4.5). Points on the projected surface can then be obtained by evaluating the surface in parameter space. In this implementation, only the boundaries of the trimming curves and meshing node points need to be projected.

$$\bar{P}_{i,j}^w = H \cdot P_{i,j}^w \quad \textbf{(4.5)}$$

$$S^w(u_l, v_l) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) \bar{P}_{i,j}^w \ ^4 \quad \textbf{(4.6)}$$

$$\overline{R^w} = (x, y, 0, w); \overline{R} = \left(\frac{x}{w}, \frac{y}{w}\right) = (x', y') \quad \textbf{(4.7)}$$

4.7.4. Back-projection

Overview*:** The basic task is to back-project points from a given camera image onto the computer-generated 3D hull surface. This is called "point inversion" in the literature [38]. This extends to the task of back-projecting a line defined by a sequence of points, for example the waterline, onto the hull surface. The essence of the method given here is to take each surface entity and project the triangulation mesh onto the camera image (unseen) using the known projection matrix, H. using the MATLAB function, ***tsearch. Then the point is interpolated within the triangle in parameter space by finding its homogeneous coordinates within the image triangle, and assuming that these are preserved in parameter space. Having located the point in parameter space, the 3D location in the surface can be evaluated directly from the NURBS surface definition. The point in the camera image to be inverted is first located within one of the meshing triangles, assuming that the projected surface entity contains the chosen point. This is achieved

[4] Note that indexes i,j do not correspond to the index i of the T-matrix.

Description: To back-project the traced waterline, the NURBS control points could be back-projected into parameter space and then evaluated in parameter space to generate a sufficient number of boundary points. However a problem can arise, and it did arise with this particular model, where the waterline crosses a narrow surface entity. A minimum of two points within the surface is needed for the surface splitting algorithm (Section 4.9) to work. Hence it is necessary to evaluate the NURBS curve before back-projection to obtain a sufficiently dense sequence of points.
In retrospect, it is clear that the operations of splitting and waterline curve fitting should be refined. The waterline found by the snake algorithm (or otherwise) could be intersected with the perspective projections of the surface boundaries on the image to give all pixels where the waterline crosses a surface boundary. Then these pixels and their immediate neighbors on the waterline could be included in the list of waterline points to be back-projected.
When back-projecting an individual point, there are three alternatives, the point is contained in precisely one triangle, the point is not contained in any triangle (which can happen at edges, or outside the surface), or the point is contained in more than one triangle. This means that there are two surfaces or a very bent surface - unusual since we normally work with half of the ship. However if the situation does occur, the nearest surface is chosen by considering the fourth value in the homogeneous coordinates.

Triangle interpolation
Overview: I find the "attraction" of each point in image space to the vertices of the enclosing triangle and apply them to the same triangle in parameter space. The attraction is represented by weights $w = [w_1, w_2, w_3]$ where $w_1 + w_2 + w_3 = 1$

Description: The $T = [0_1, 0_2, 0_3] = [x'_1, y'_1], [x'_2, y'_2], [x'_3, y'_3]$ define the triangle that encloses a point in image space. Consider a point $Q_r = [x'_r, y'_r]$, within the triangle. The problem is to find the corresponding point Q_r^{UV} in parameter space using weights $w = [w_1, w_2, w_3]$ for the same triangle $T = (R_1^{UV}, R_2^{UV}, R_3^{UV})$ defined by $R^{UV}{}_1 = [u_1, v_1], R^{UV}{}_2 = [u_2, v_2], R^{UV}{}_3 = [u_3, v_3]$.
$\overline{Q_0}$ can be represented as the weighted sum of the triangle corner points. Then

$$\begin{aligned} x'_0 &= w_1 x'_1 + w_2 x'_2 + w_3 x'_3 \\ y'_0 &= w_1 y'_1 + w_2 y'_2 + w_3 y'_3 \end{aligned} \quad \textbf{(4.8)}$$

Together with the condition

$$w_1 + w_2 + w_3 = 1 \quad \textbf{(4.9)}$$

This gives the matrix equation for the weights, $w = [w_1\ w_2\ w_3]$

$$\begin{pmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{pmatrix} w^T = \bar{Q}_0^T \quad \textbf{(4.10)}$$

where $Q_0 = [x_0\ y_0\ 1]$ which in image coordinates is represented as $\overline{Q_0} = [x'_0\ y'_0]$

Having found w, we find $Q_0^{uv} = [u_0\ v_0\ 1]$ from

$$\begin{pmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{pmatrix} w^T = [Q_0^{UV}]^T \quad \textbf{(4.11)}$$

The result is a set of points in parameter space Q^{UV} which can be evaluated within the surface to obtain their coordinates in model space.

Algorithm 4.1. Back-projection algorithm for single point

Description: given a point $\bar{Q}_1 = [x'_1, y'_1]$, the perspective projection matrix, H and a surface S_k find the corresponding value in parameter space Q_1^{UV} with the triangles (T_i^k) and evaluate obtaining $[X_1, Y_1, Z_1]$,

Require: $\bar{Q}_1$, H, S_k,T_i^k

1. Forward project the mesh using H
2. Find and store the triangle $i - index$ so that $T_i \supset \overline{Q_1} = (x'_1, y'_1)$. If more than two triangles are found, choose the closest one to the camera
3. Triangle interpolation
4. Store the interpolation weights
5. Apply the weights and calculate the new point in parameter space Q_1^{UV}

Return: $\boldsymbol{Q_1^{UV}}$

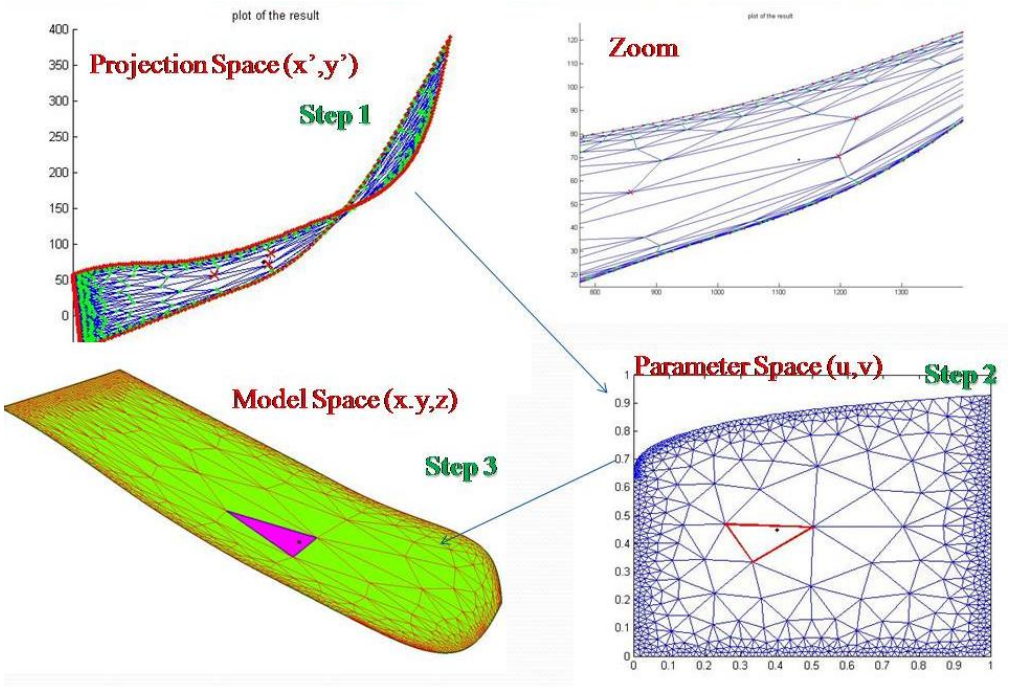


Figure 4.10. Graphical back-projection schema. **Step 1:** represents the image space. In black, a waterline point. Red crosses, are the vertices of the $\boldsymbol{i}$ −triangle enclosing the waterline point. In red, the boundary of the surface. In green, the nodes of the meshing. In blue, the triangles of the mesh. **Step 2:** represents the parameter space. The $\boldsymbol{i}$ −triangle enclosing the point is plotted in red. The weighted waterline point is plotted in black, inside the triangle. **Step 3**: Represents the three-dimensional space. In magenta, the $\boldsymbol{i}$ −triangle. In black, the evaluation of the waterline point onto the surface. In red, the triangle mesh of the surface.

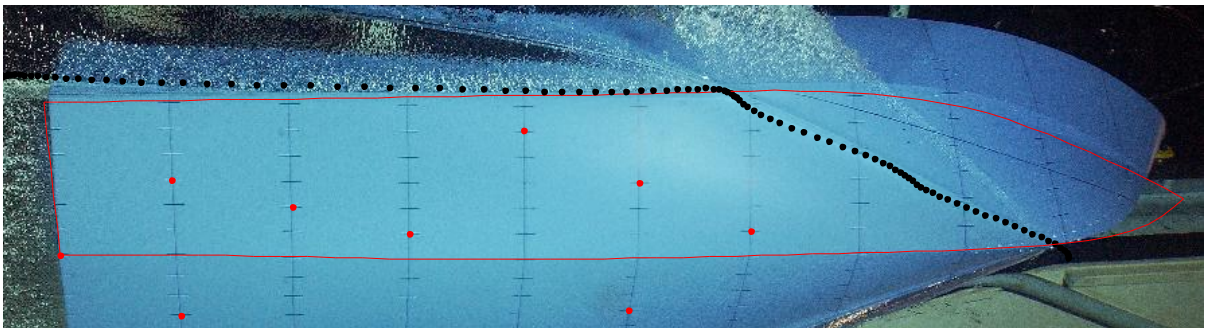
The next step is to back-project the waterline. In order to calculate the wetted area of each individual hull surface, it is convenient to represent the waterline as a trimming curve dividing the surface. Then the other processing is already available, including displaying the waterline to the operator. Trimming curves are represented as NURBS curves in the IGES2MATLAB data structure, so the task is to back-project the waterline and save as a NURBS curve.

Algorithm 4.2. Back-projection of the waterline

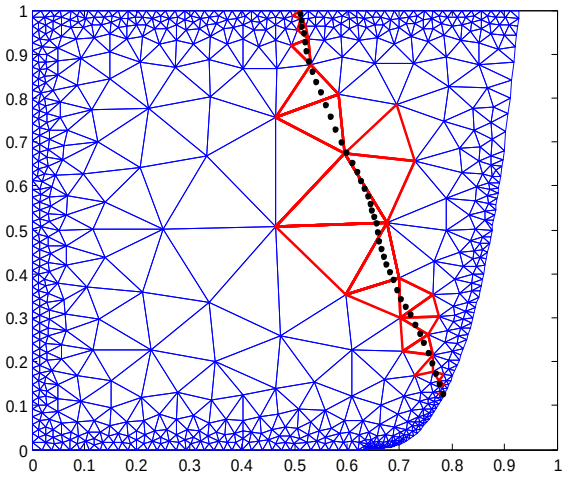
Description: given the NURBS waterline, the projection matrix of camera, H, find the corresponding nodes Q_r^{UV} with the triangles and evaluate obtaining $[X_r, Y_r, Z_r]$

1. Forward project the mesh using H
2. Sample the NURBS curve with control points and obtain $\overline{Q_r} = (x'_r, y'_r), r = 1 \dots R$
3. Find $i - index$ so that $T_i^k \supset \overline{Q_r} = (x'_r, y'_r)$
4. Triangle interpolation algorithm requires $(R_k^{UV}, i, T_i, \overline{Q_r})$ and returns Q_r^{UV}
5. Create the NURBS curve in the parameter space domain of the surface

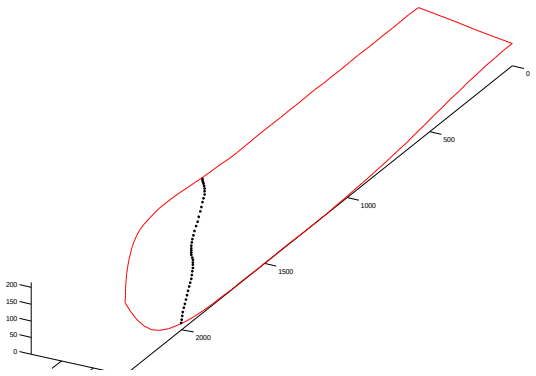
Return: $\boldsymbol{Q_r^{UV}}, Q_r$



(a)



(b)



(c)

Figure 4.11 (a) In red, the forward projection of the boundaries of the surface. In black dotted points the waterline from the detection algorithm**. (b)** In blue, the parameter space representation of the surface. In red, the triangles $\boldsymbol{T_i}$ containing the waterline points. In black, the back-projected waterline points into parameter space**. (c)** Three-dimensional plot: in red, the boundary of the surface. Black dotted points, the evaluated waterline points onto the surface.

4.8. Surface Splitting

The problem here is to find the intersection between the waterline and the individual NURBS curves trimming the each of the surfaces comprising the hull (Figure 4.12). I considered the waterline to divide the affected surface in two. Then two new surfaces are formed with a new sequence of trimming curves, the waterline being one of them.

If any waterline point is back-projected onto a surface S_k, this is considered to be a "wetted surface entity" (Figure 4.13). I check if the waterline crosses the edge of S_k an even number of times. If this happens, the splitting algorithm is executed; otherwise the surface is ignored. Intersections with a surface boundary are determined as follows. The trimming curves of the surface area are evaluated to obtain a vector of points forming a closed polygon. The function ***inpolygon*** in the MATLAB *Mapping Toolbox* returns all waterline points inside the trimming curve. This set of points is then extended by a few points in the directions of the first derivative at the leftmost and rightmost point (in case the waterline terminates just before the boundary). Then the MATLAB function ***polyxpoly*** is used to find the intersection of the waterline and the trimming curve. It is now straightforward to construct two new boundary polygons defining the original boundary polygon split by the waterline. This gives a new set of trimming curves $C_n(t)$ for each of the resulting split surfaces. A whole picture of this process is represented in Figure 4.14. In order to know which of the divided surfaces is wet, the mid-point of the waterline is selected inside the surface in the image space and moved a small step downwards. I back-project this point to parameter space and use ***inpolygon*** again to determine which of the two new surfaces the point belongs to. This surface is then re-meshed to obtain a new set of nodes and triangles. This polygon formulation is sufficient for area estimation

At this stage the new trimming curve is defined as the polygon given by a vector of boundary points. In order to plot the waterline on different camera views using the IGES2MATLAB plot package, the waterline vector must be converted to a NURBS curve. This is achieved by treating each vector point as a NURBS control point. An approximation is involved here, since NURBS curves do not usually pass through their control points. However I use a dense vector of points so the error should be negligible. Although any surface that is not split makes no contribution to the wetted area, all back-projected waterline points are plotted in the 3D model shown here in this thesis and can be displayed to the operator.

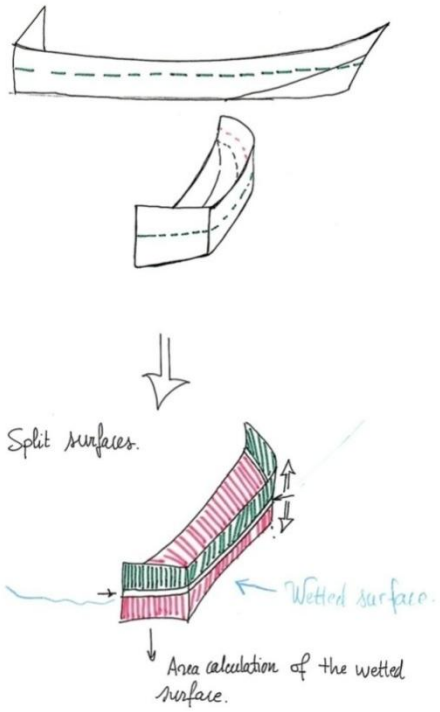


Figure 4.12. Surface Splitting. The waterline treated as boundary of trimmed surface (for plotting and area measurement).

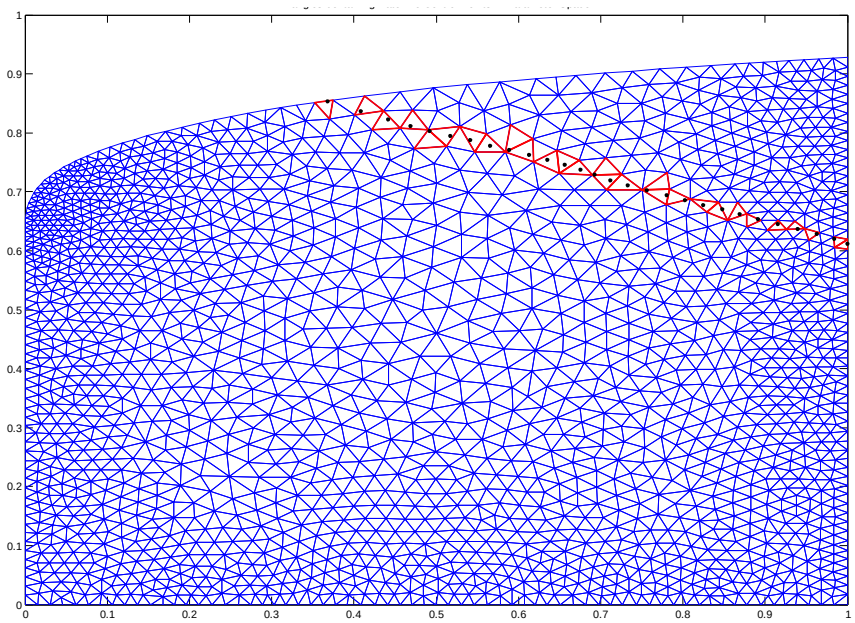


Figure 4.13 Illustration of the surface before the splitting algorithm. Parameter space plot of the surface and back-projected waterline. In red the triangles enclosing waterline points. In black, the waterline points. In blue the mesh of the trimmed surface.

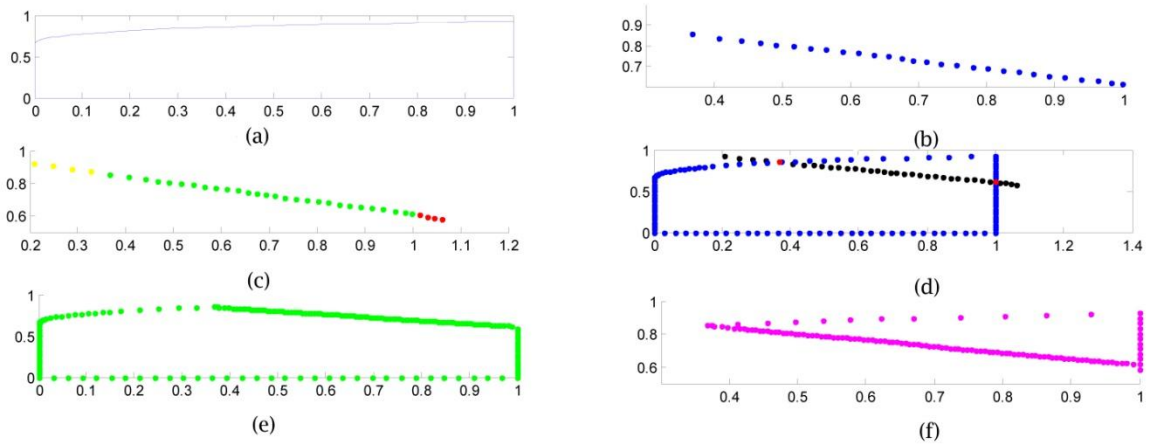


Figure 4.14 Illustration of the splitting algorithm. **(a)** surface boundary in parameter space. **(b)** back-projected waterline points within the boundary **(c)** waterline points extended to surface boundary **(d)** intersection of boundary with waterline points **(e)** Polygon A: boundary of first split surface **(f)** Polygon B: boundary of second split surface

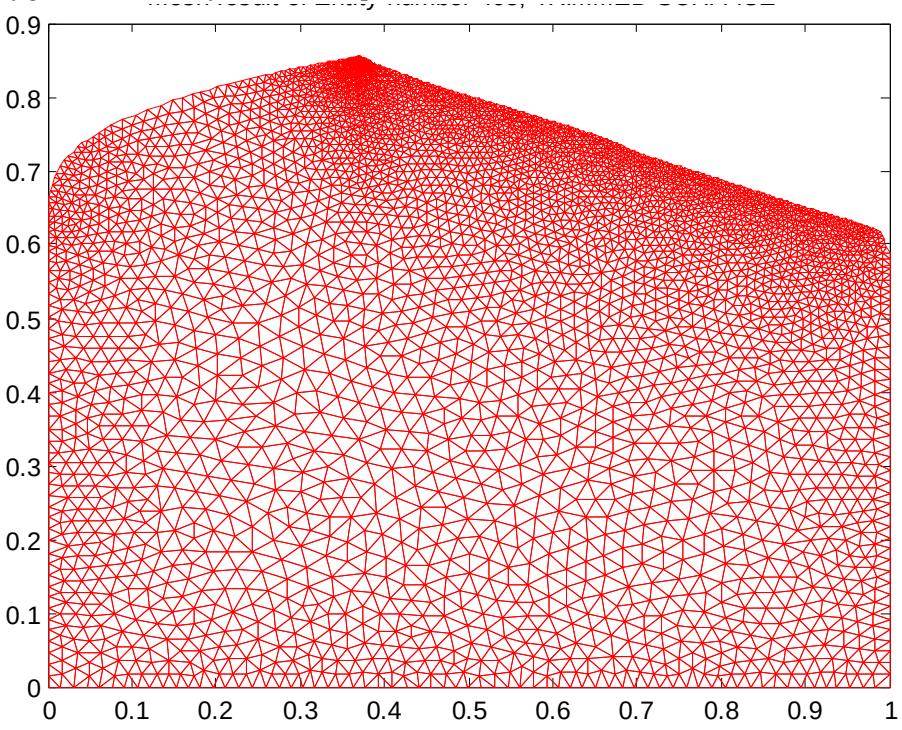


Figure 4.15 Mesh result of one of the split surfaces.

Algorithm 4.3. Split surface algorithm

Description: Given the selected wet surface S_k including its trimming boundaries C_n, its triangulation matrix T, the waterline points in parameter space Q_n^{UV}, a selected wetted point Q_{wet}^{UV}. The idea is to divide the surface in two, select the wetted half S_{k1}trimmed by a new set of curves C_n'., and re-mesh it, obtaining a new set of mesh nodes R and a new triangulation matrix T_j^{k1}

Require: $i, S_k, Q_{wet}^{UV} Q_n^{UV}, T$

1. Check if it is a trimmed or untrimmed surface
2. Read T, C_n
3. Extend the line
4. Construct a new trimming boundaries C_n' for the wetted surface *
5. Check in which surface is the triangle $T_i \subset Q_{wet}^{UV}$ and store $k1$
6. Re-mesh obtaining a new set $T_j^{k_1}, R_{k1}^{UV}$ where k_1 is the wetted surface

Return: T_j^k, R_k^{UV}

(*)**Note**: note that C_n has a predefined orientation defined in the IGES standard

4.9. Area calculation

Given a wetted surface, $S(u, v)$, and its corresponding triangle mesh, R, area estimation consists of adding the areas of the constituent triangles. The area of each triangle is given by the vector cross-product of two of the sides, $A = \frac{1}{2}|V_1 \times V_2|$ or alternatively by the orthogonal-triangular decomposition [39].

The total area of each surface is the sum of the individual triangle areas, while the total wetted area is obtained by adding the areas of the individual wetted surfaces.

Chapter 5

Test examples – Results

5.1. Introduction

A variety of tests for different aspects of the research are considered here: registration accuracy, fidelity of waterline delineation, and precision of the back-projection and splitting algorithm. Finally, complete system test results are given for the experimental model hull together with the test camera images. Here the wetted surface area estimates obtained using my proposed methods are compared with the manual estimates made by an SSPA expert. I also give some timing estimates for different operations, although it should be stressed that no systematic optimisation has been carried out.

5.2. Image registration from the underwater camera

***Description*:** Image registration was done using correspondences between selected points in the camera image and the 3D model. Point matching was carried out manually. Registration of the underwater camera image was evaluated in two ways. Firstly, by perspective projection of the RHINO registration points onto the underwater camera image. The coordinates of these registration points were supplied by SSPA. This gives the registration error in pixels. Secondly, by back-projection of the image registration points onto the RHINO hull. This gives the registration error in millimetres.
In Figure 5.2, registration points marked by the operator are circled in white, while the projected points using the estimated transformation matrix estimate are marked with red dots. Boundaries of the boat are projected into the image. The error in pixels and on the model surface for 10 registration points is given in Figure 5.3. The rms errors are 1.8 pixels corresponding to 2.4 mm on the surface. Distortion of the hull boundaries above the waterline is due to refraction.

***Comment*:** Results were very satisfactory. The rms deviation of only 1.8 pixels was very low. The standard deviation is lower that the mean at 0.8 pixels, suggesting a random error distribution. Errors in mm are more variable, suggesting that other factors than operator and painting errors, such as camera aberrations could be contributing.

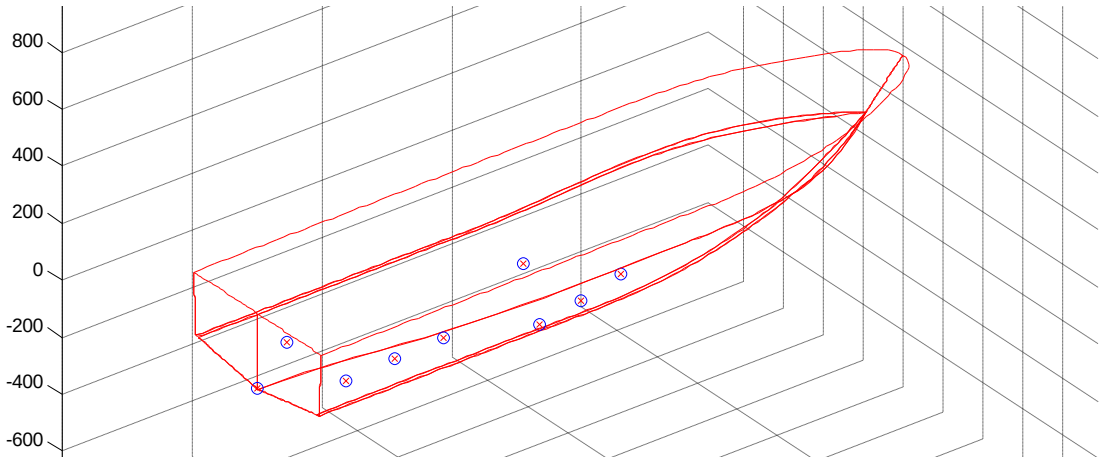


Figure 5.1. Registration points represented in the 3D model. In blue-redish points are the registration points used for the estimation of H.

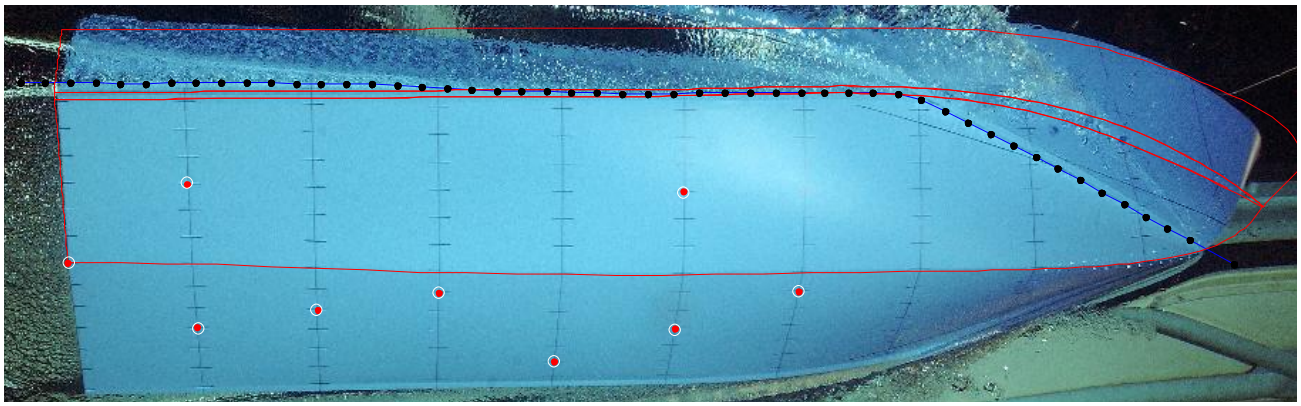


Figure 5.2. Registration results of the underwater camera. White circles are the operator's input. Red points are the projected registration points using the estimated H. Red lines are the boundaries of some of the surfaces. The black points and the blue waterline are not relevant for this section.

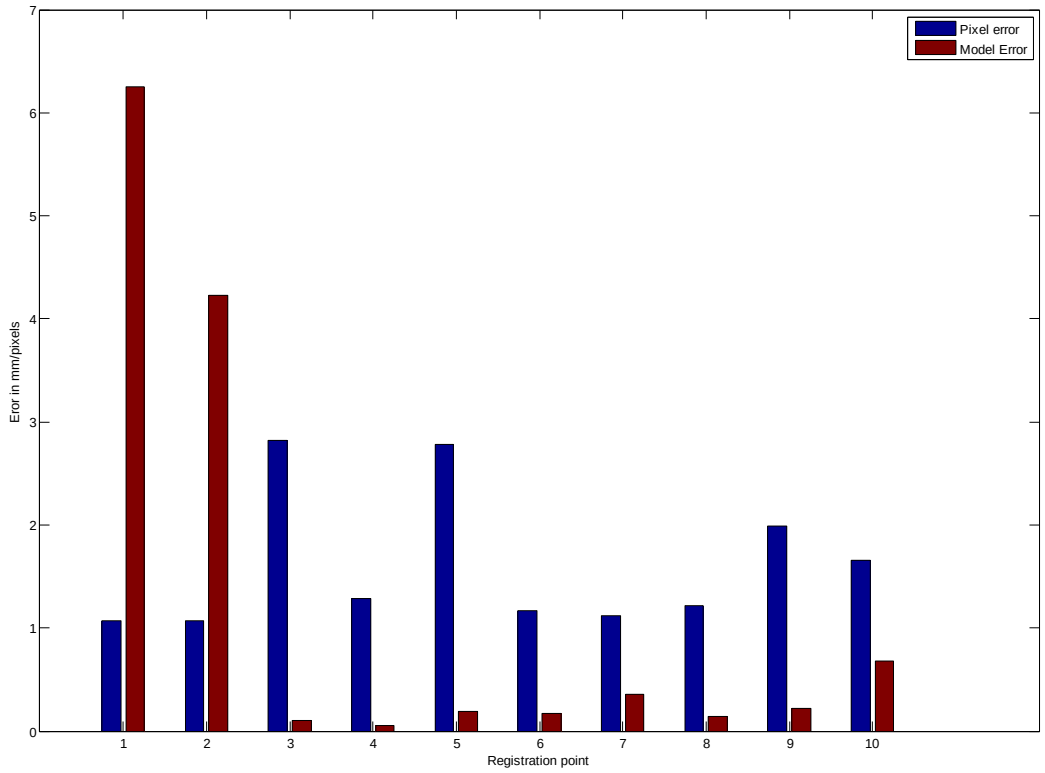


Figure 5.3 Pixel error and its corresponding error in the model in millimetres.

5.3. Image registration from the stern camera

***Description*:** The registration points were chosen to be as spread as possible in the model (Figure 5.4). Figure 5.5 shows the results. Figure 5.6 shows the projected boundaries of the hull onto the picture (red), as well as the waterline detected from the underwater camera (blue). The boat is severely foreshortened by perspective errors in the transformation matrix. I then tried another set of registration points (Figure 5.7), which gave much better results (Figure 5.8). Here the rms error is 2 pixels. The errors were too large for back-projection to yield points lying on the surface, so the rms error in mm is not available.

Comment: Why is registration using the stern camera so much less reliable than registration using the underwater camera? There are different possible explanations. Firstly, stern camera registration is very sensitive to small errors in registration points along the side of the hull, i.e. at surfaces almost parallel to the camera direction. Secondly, this set of registration points was not provided by SSPA, so I attempted to determine them myself with RHINO, leading to possible errors in the 3D specification, could be definition. Finally, it should be noted that determination of the waterline end point appears to be correct, even if features further from the camera are distorted. However this experience suggests that multiple-view registration needs detailed investigation.

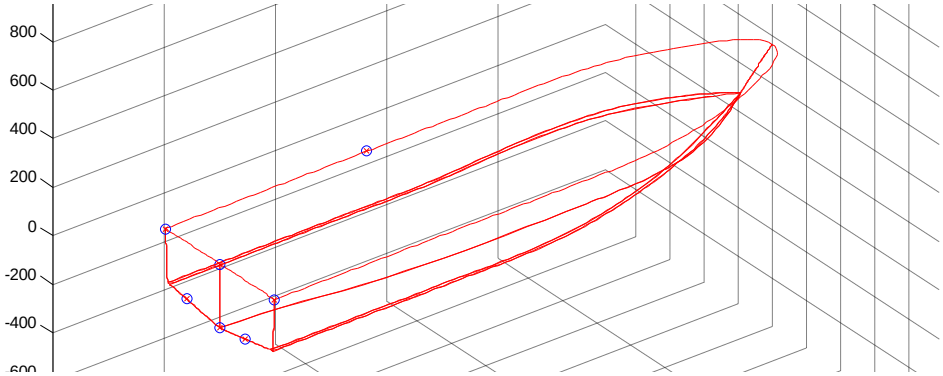


Figure 5.4. Registration points distribution used for the stern camera. In blue-redish points are the registration points used for the estimation of H.

Figure 5.5. Registration results of the stern. White circles are the operator's input. Red points are the projected registration points using the estimated H.

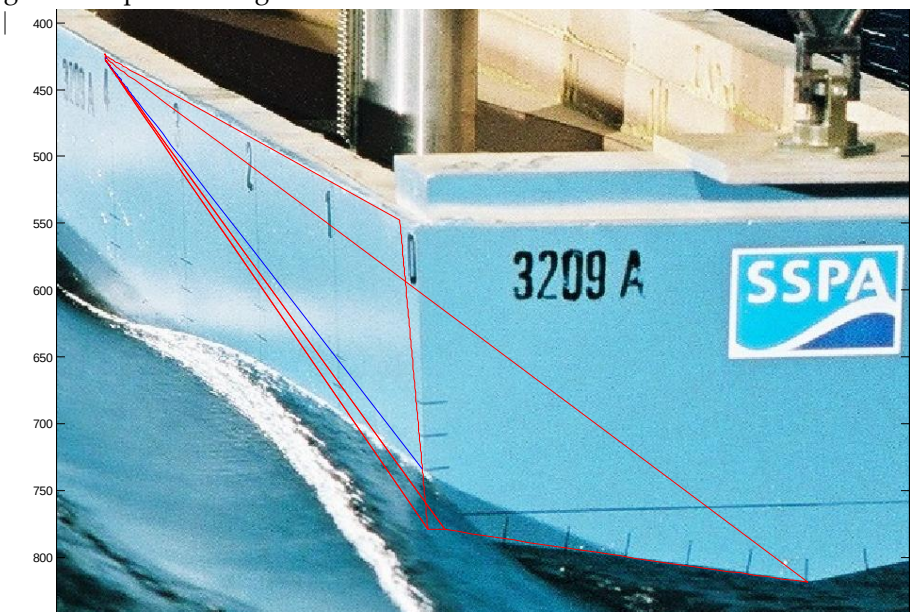


Figure 5.6. Projection of the surface boundaries and the detected waterline using the underwater camera. In blue the waterline, in red the boundaries of the surfaces the waterline crosses. In this case the registration was wrong and therefore also the projection.

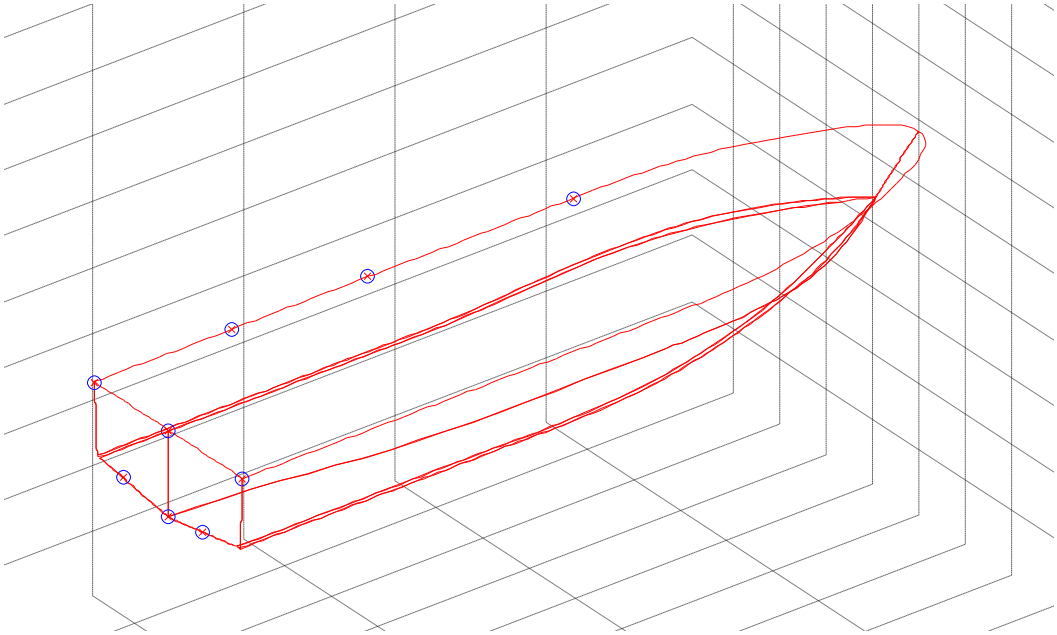


Figure 5.7 Second set of registration points distribution for the stern camera.

Figure 5.8 Projection of the surface boundaries and the detected waterline from the underwater camera using the second set of calibration points.

5.4. Waterline Estimation

***Description*:** Figure 5.9 shows the underwater image at zero speed, with no bubble interference. The red line is the waterline marked by an SSPA expert. The black line shows the waterline determined by the snake algorithm. The rms difference between them is 3.7 pixels, approximately 28 mm, allowing for the scale factor at the bottom of the hull. The mean difference was 1.6 pixels, approximately 13 mm. Closer inspection shows that the snake estimate often runs parallel to the manual estimate so there could be systematic differences which the semi-automatic algorithm needs to consider. This mean error translates into a fraction of a square metre error for the wetted surface.
The test was repeated for each towing speed (Figure 5.9, Figure 5.12, Figure 5.14, Figure 5.15). Figure 5.11 and Figure 5.13 are the weight maps using hot colour map. Average computation time was 73 seconds including the time that took me to select the points and the region.

Comment: It is not certain that the manual estimates are completely accurate. However the difference between the semi-automatic and the manual estimates seems acceptable. Further investigation of the differences would be worthwhile.

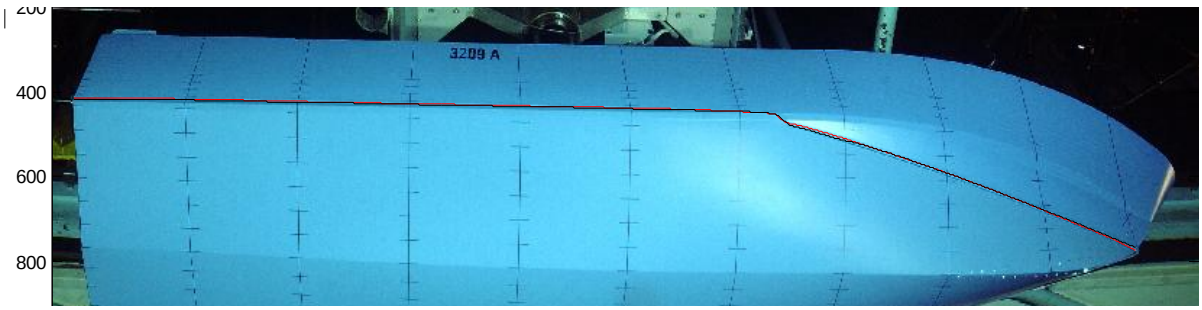


Figure 5.9 Waterline for 0 knots. In red, manual estimate of the waterline. In black, automated estimate of the waterline.

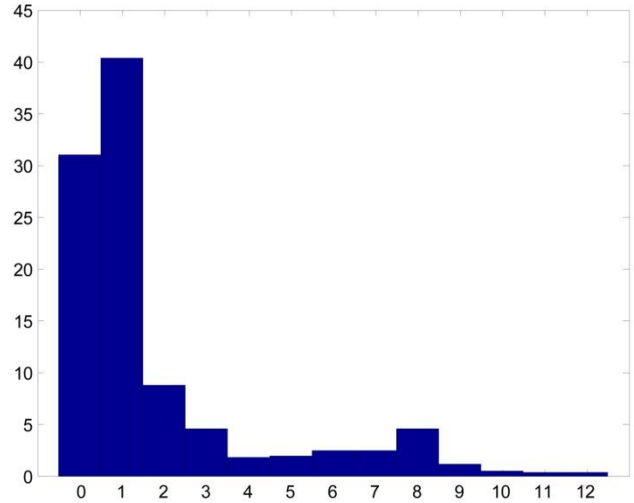


Figure 5.10 Histogram showing percentage of pixels for given pixel distance. Pixel percentage deviation between the manual and the snake waterline.

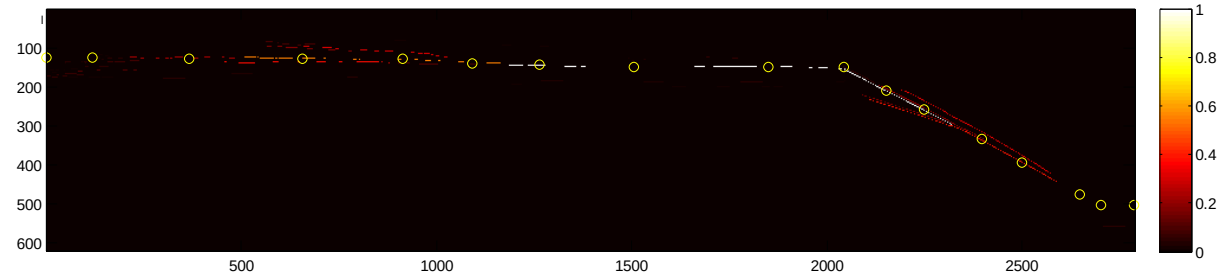


Figure 5.11. Weight map for 20 knots. Yellow circles are the operator's input. Lines are the weights using a hot colour map.

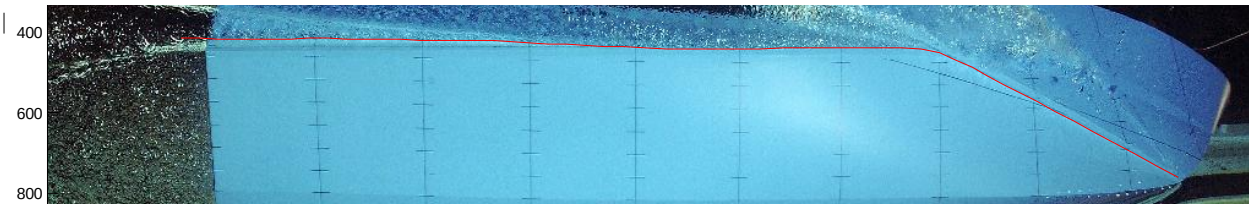


Figure 5.12. Waterline for 20 knots. In red the waterline.

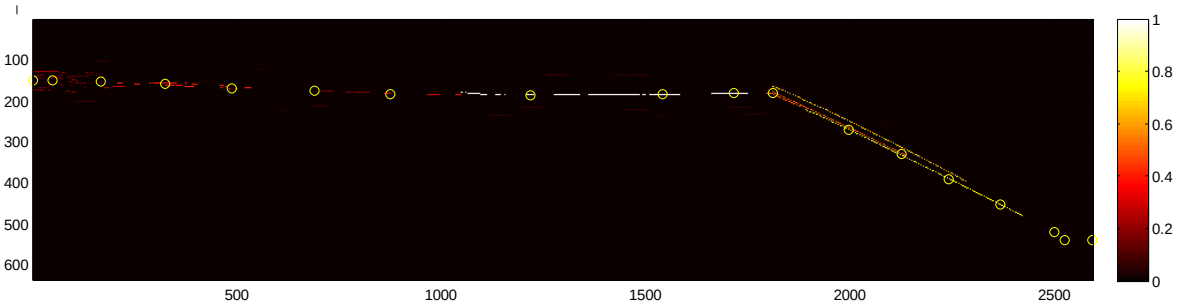


Figure 5.13. Weight map for 25 knots.

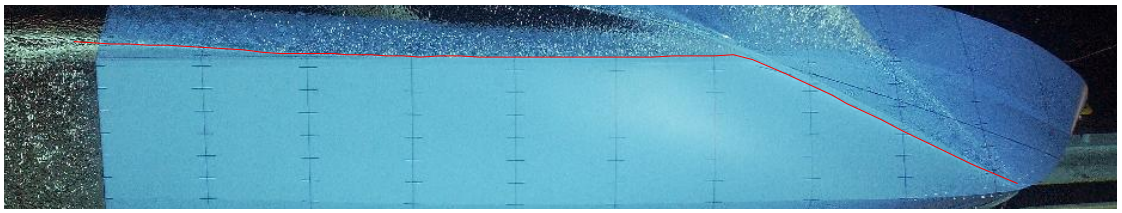


Figure 5.14. Waterline for 25 knots.

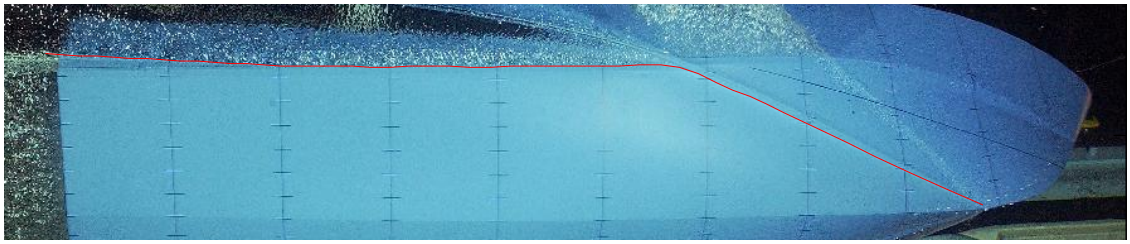


Figure 5.15. Waterline for 30 knots.

TABLE 5.1. TIME RESULTS

Speed	Time
knots	*seconds*
20	79.4
25	78.1
30	61.7

5.5. Back-projection tests

Overview: The purpose is to estimate the accuracy of the back-projection and splitting algorithms, indicating how much error is introduced at this stage. Here synthetic data is used. The perspective projection matrix used was derived from the registration of the underwater camera image. However it is irrelevant which matrix is used, I chose this matrix to overlap the results with the experimental images and give a clear indication of what is happening.

Description: I modified the digital model in Rhinoceros and created a line onto one of the surfaces of the boat. I name it "water" in order to recover it in MATLAB (Figure 5.16). I projected this line into the picture with the estimated H (the projection matrix) together with the boundaries of the surface of the boat (Figure 5.17). The line is back-projected from the image space to the parameter space of the surface (Figure 5.18). The zoomed section (Figure 5.19) shows waterline points, and their enclosing triangles. Figure 5.20 shows the operations of the splitting algorithm in parameter space, where two resulting surfaces are shown (k1 and k2). The wetted surface (k1) is re-meshed with the new contour line (Figure 5.21) and finally the area of this surface is estimated and plotted in Figure 5.22. The areas estimated using Rhinoceros software and MATLAB were compared and shown in TABLE 5.2.

***Comment*:** This result suggest that the back-projection and splitting algorithms introduce negligible error into the system. Therefore, errors will come from the registration and from the waterline estimate in the image analysis.

TABLE 5.2 ESTIMATED AREA FROM AN IDEAL BOAT

Method	Estimated area	Difference
	mm^2	%
Rhinoceros	230551.784 (+/- 0.019)	0.0721
MATLAB	230718.2	

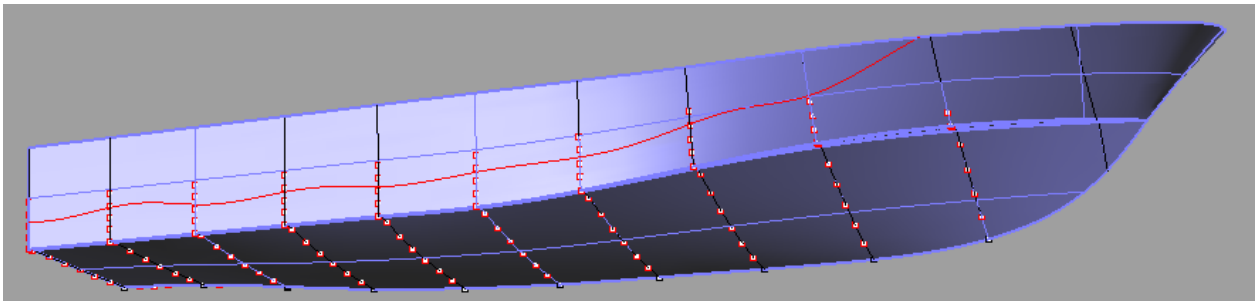


Figure 5.16. Rhinoceros model of the boat and a artificial waterline.

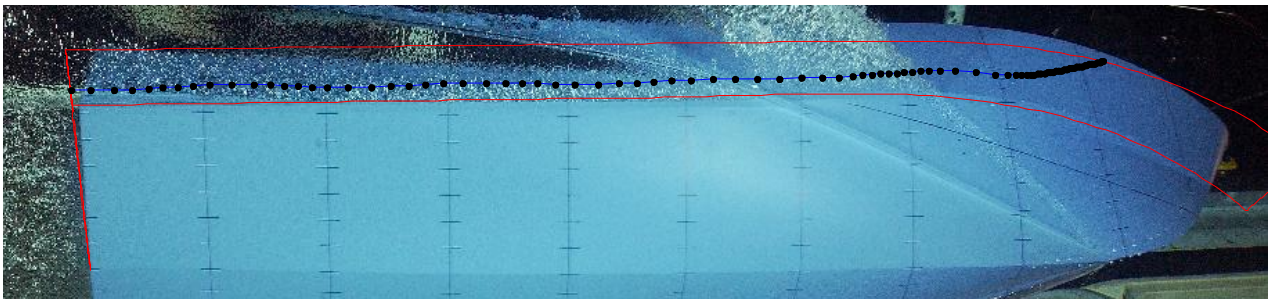


Figure 5.17. Projection of the waterline.

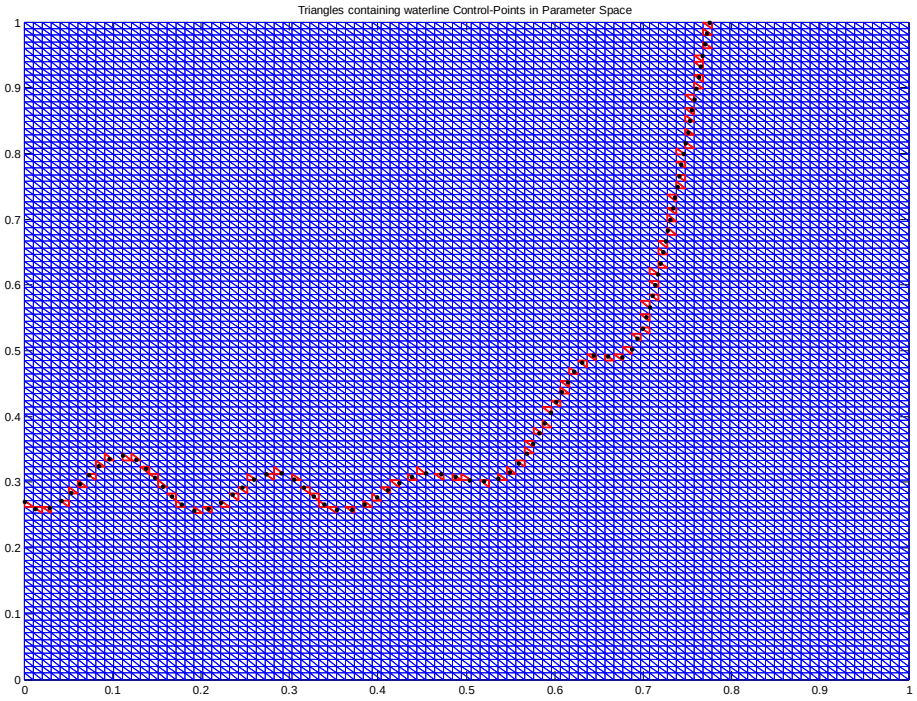


Figure 5.18. Parameter space plot of the surface. In red, the back-projected points of the waterline.

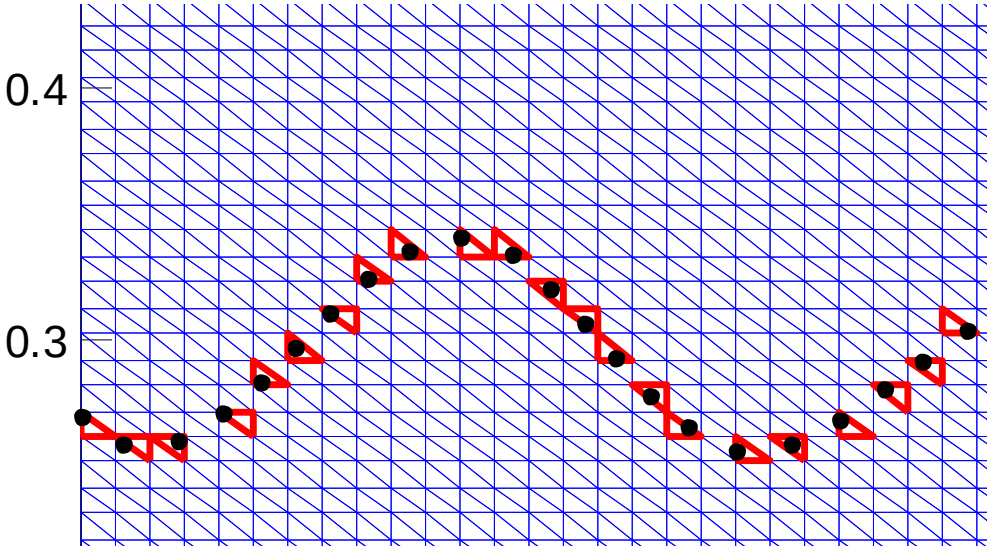


Figure 5.19. Parameter space zoom of the surface, some waterline points and its enclosing triangles (red).

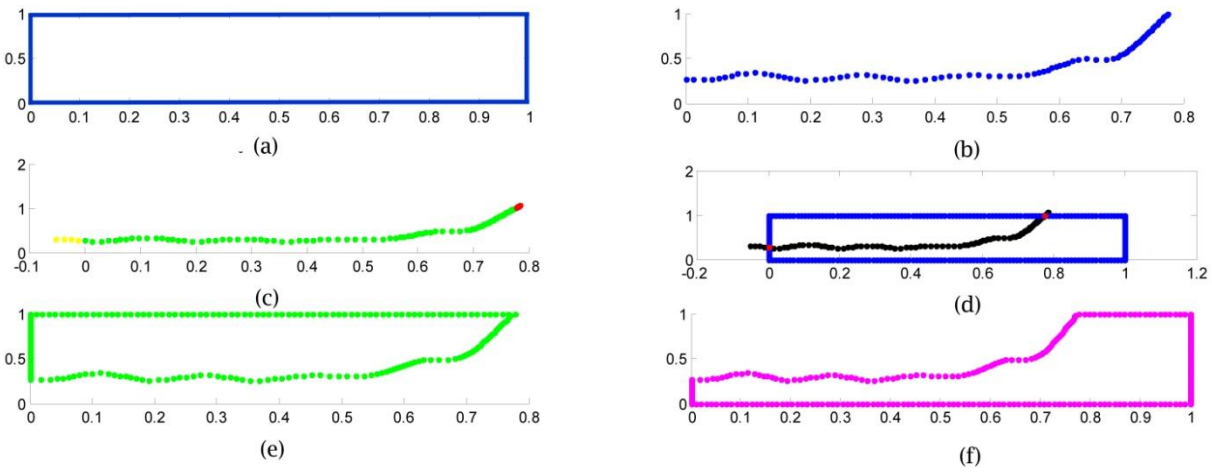


Figure 5.20 Illustration of the splitting algorithm. **(a)** Surface boundary in parameter space. **(b)** Back-projected waterline points within the boundary. **(c)** Waterline points extended to surface boundary **(d)** intersection of boundary with waterline points. **(e)** Polygon A: boundary of first split surface. **(f)** Polygon B: boundary of second split surface.

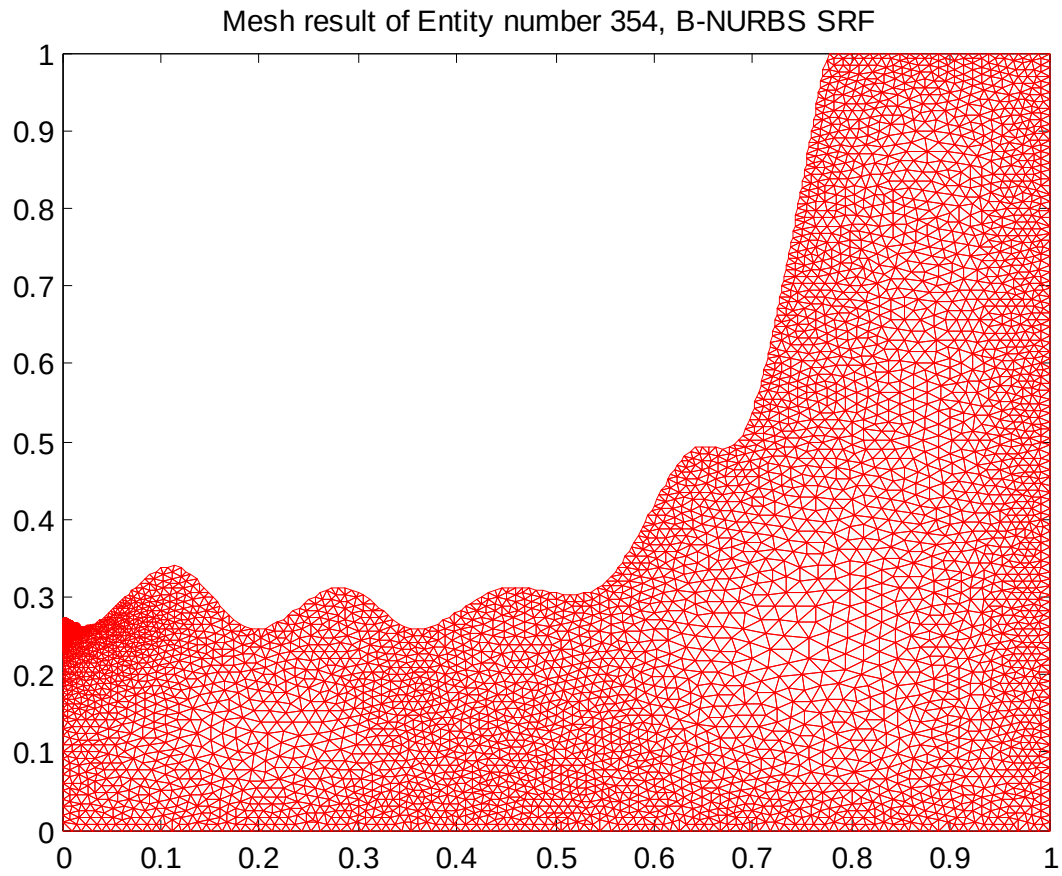


Figure 5.21. Mesh result of the wetted surface.

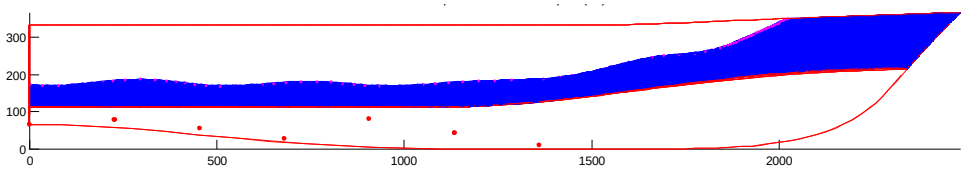


Figure 5.22. Wetted surface of the hull with hand-drawn waterline.

5.6. Complete system tests

***Description*:** The waterline is traced in each camera image, taken at different speeds, and projected back on to the model, where it divides each wetted surface. Figure 5.24, Figure 5.28 and Figure 5.32 shows the traced waterline together with the projection of the surface boundaries. Figure 5.30 shows graphically the estimated wetted area. The area of the stern was not considered. The manual and semi-automated area estimates are compared in TABLE 5.3, which also shows the percentage difference between these estimates at different towing speeds. Registration errors at individual registration points are given in Figure 5.23, Figure 5.27, and Figure 5.31. Wetted area estimates were compared for different registrations and waterline tracking operations at zero towing speed. For the same registration, the two estimates were 70.2 m^2 and 71.8 m^2, a difference of 2.8%. For the same waterline but

different registrations, the estimates were 71.6 m^2 and 72.2 m^2, a difference of 0.8%. The average of these estimates is given in Table 5.3.

Comment: There seem to be systematic difference with towing speed, but there are too few results to support any hypothesis. It would have been helpful to back-project the waterline from the stern camera image, and use this when tracing the waterline in the underwater-camera images. Unfortunately this could not be done. On the other hand the projections of the underwater-camera waterlines on stern camera images (Figure 5.26, Figure 5.29 and Figure 5.33) suggest that no great errors were made at the point where the waterline meets the stern.

Further comments and recommendations: I believe that above cameras should be mounted higher on the platform, giving a better view of the waterline, less obscured by waves. I also believe that using the same side for the underwater camera images would be better than the current arrangement, even though the waterline is assumed to be symmetric (See Figure 4.2).

TABLE 5.3 COMPARISON OF AREA ESTIMATES WITH MANUAL ESTIMATES

Speed	Measured area by thesis method	Manual measurements	Difference
Knots		m^2	%
0	71.45	68.16	4.7
20.00	71.98	71.94	0
25.00	67.00	69.95	-4.2
30.00	60.57	62.94	-3.8

Test for 20 knots

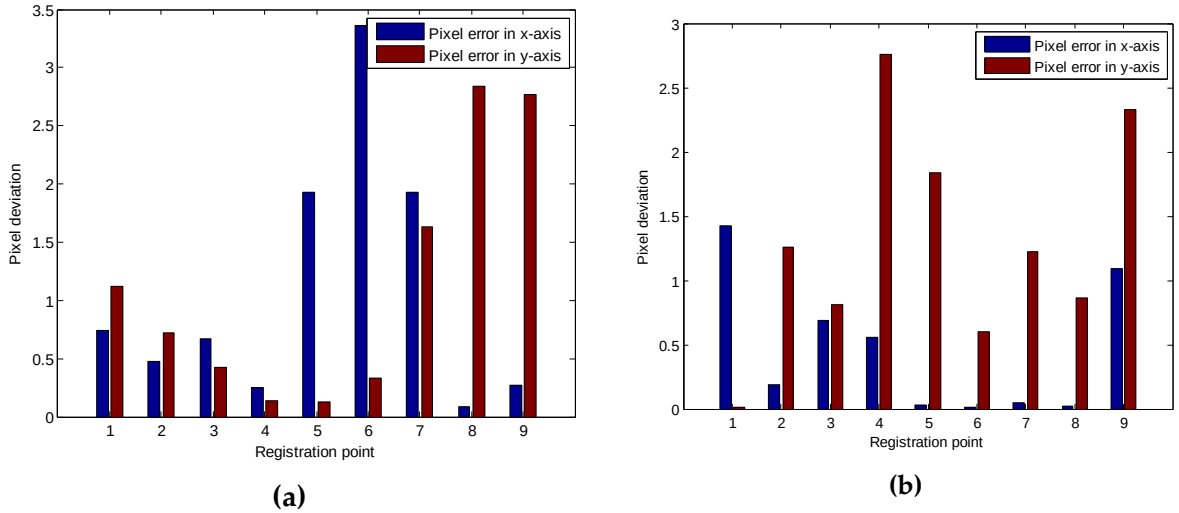


Figure 5.23 Results of the registration. (a) Registration error for the underwater picture**.**
(b) Registration error for the stern camera.

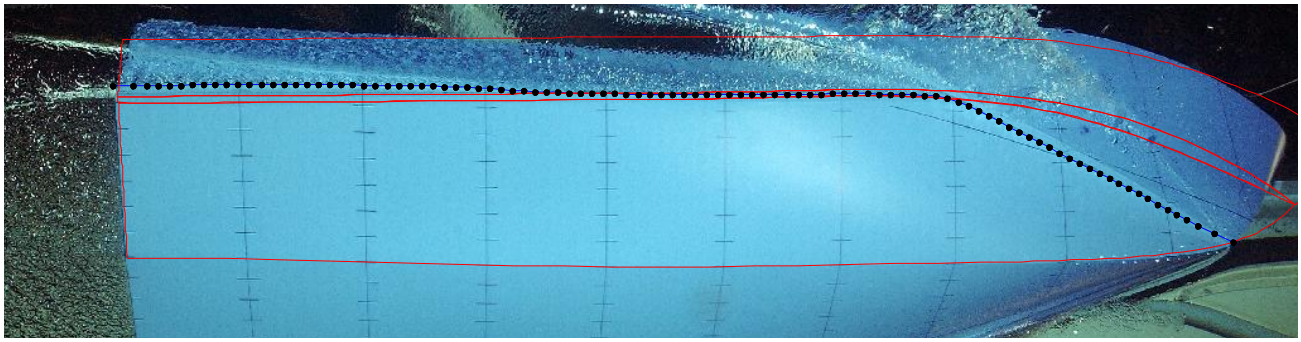


Figure 5.24 Waterline (black dotted line). Boundaries of the affected surfaces (red).

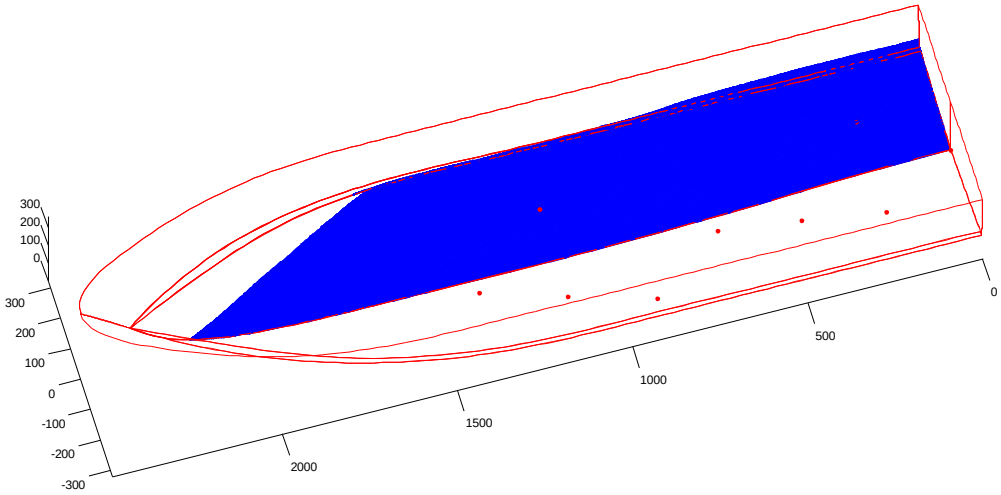


Figure 5.25 Tri-dimensional plot of the wetted surface.

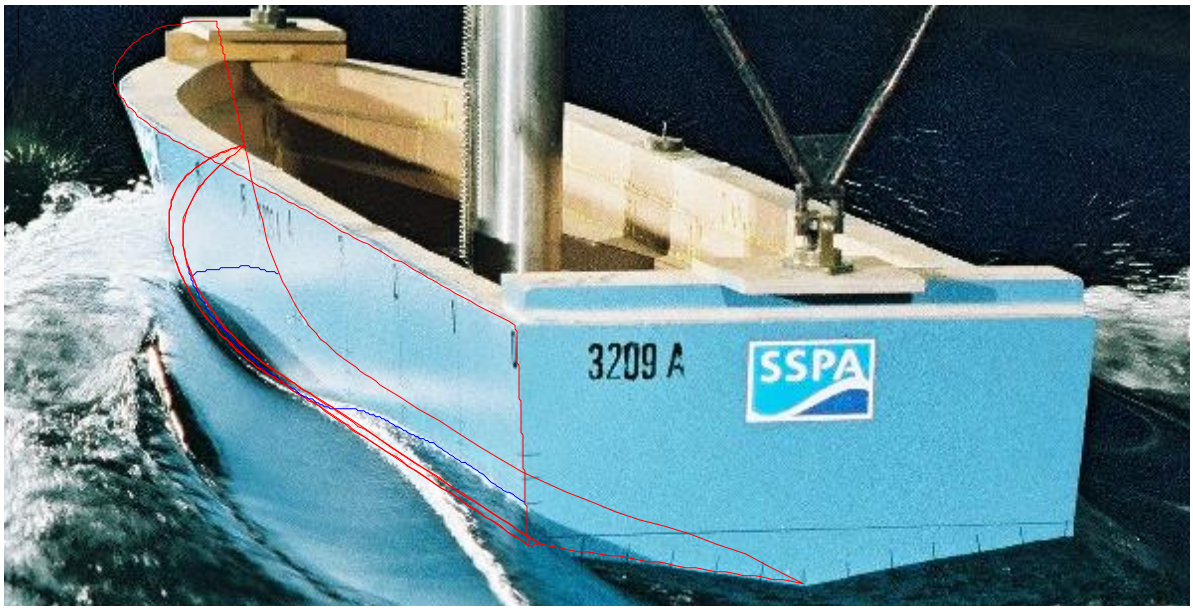
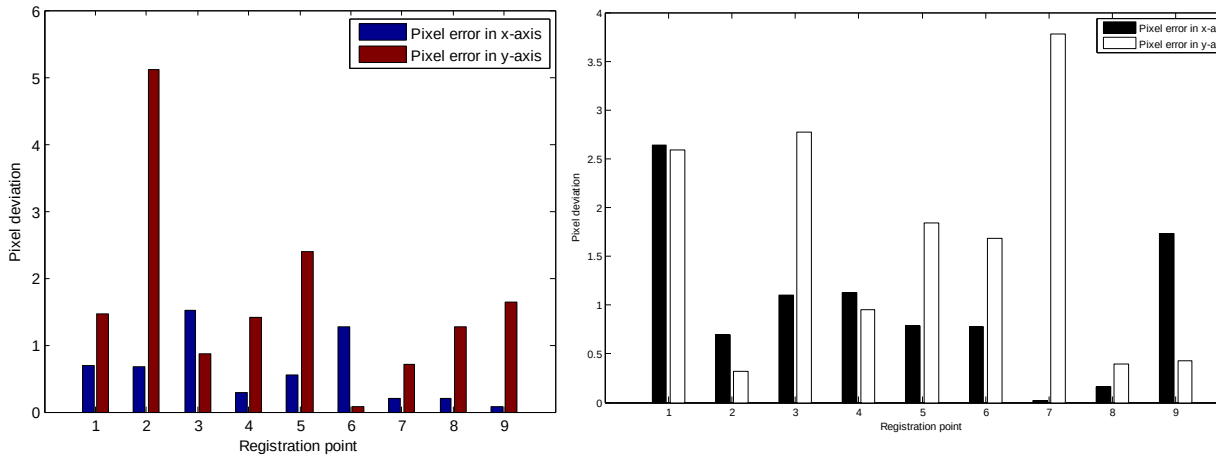


Figure 5.26 Re-projection of the waterline in the stern view.

Test for 25 knots



(a) **(b)**

Figure 5.27 Registration results: **(a)** Underwater view. **(b)** Stern view.

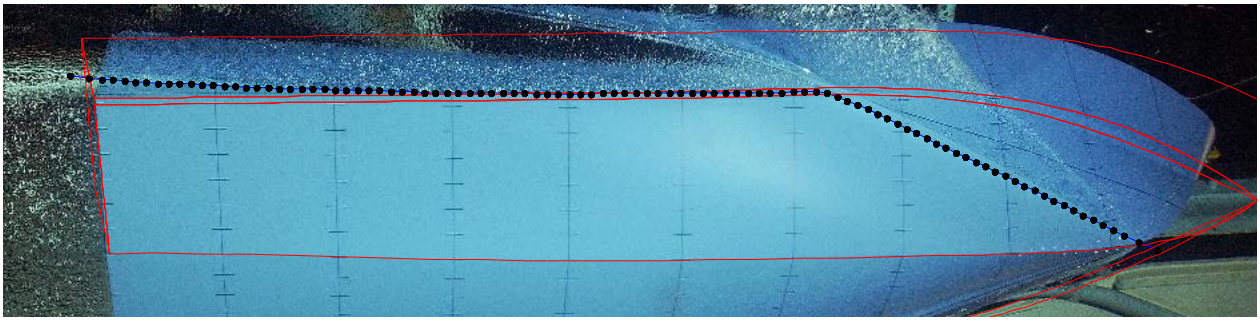


Figure 5.28 Waterline estimate (black dotted points) and the boundaries of the surfaces (red).

Figure 5.29 Re-projection of the waterline on the stern view.

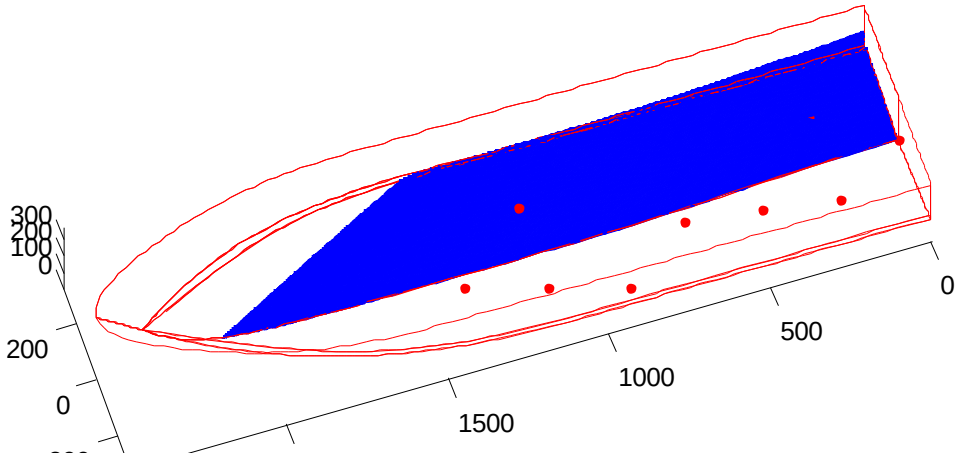
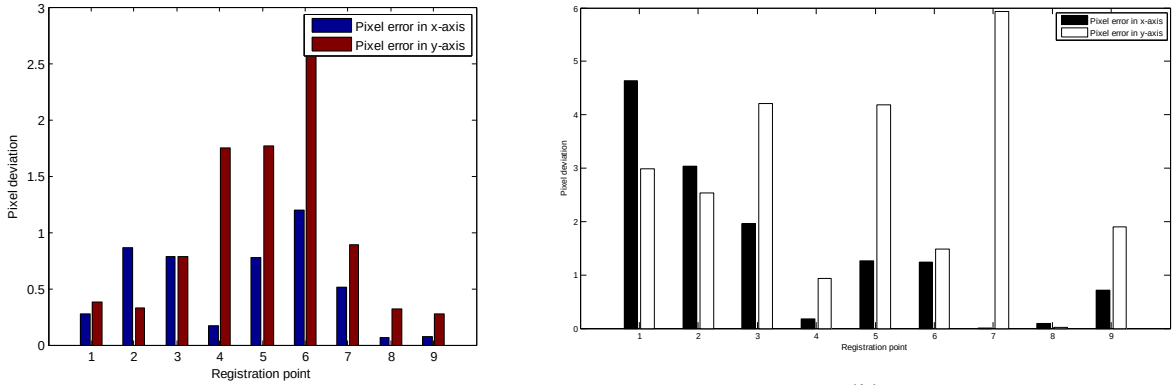


Figure 5.30. Graphical representation of the wetted surface estimated at 20 knots. (View from the underwater camera).

Test for 30 knots



(a) **(b)**

Figure 5.31 Registration results: **(a)** Underwater view. **(b)** Stern view.

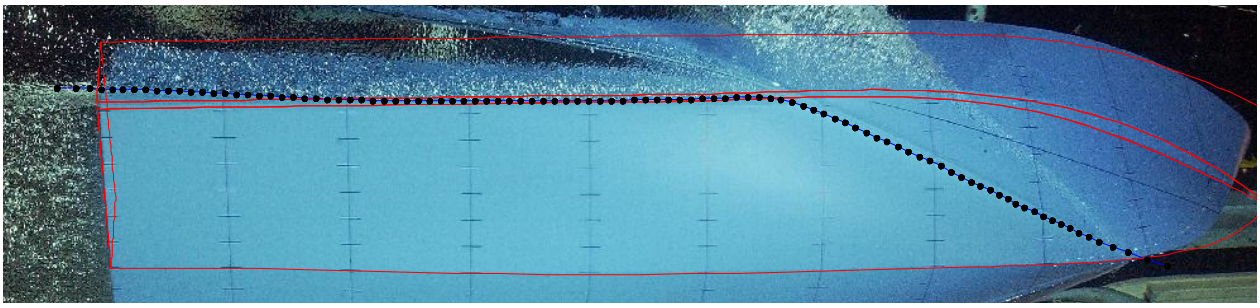


Figure 5.32 Waterline estimate (black dotted points) and the boundaries of the surfaces (red).

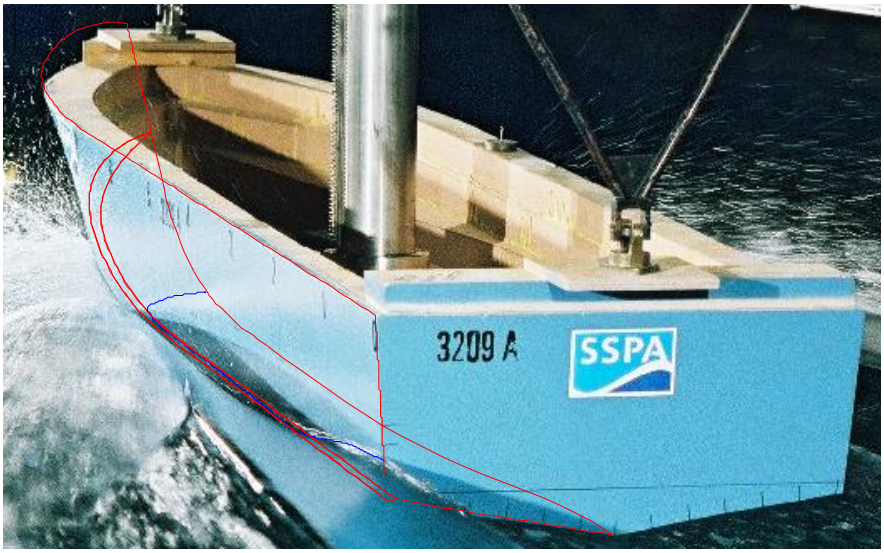


Figure 5.33 Re-projection of the waterline on the stern view.

5.7. Timing results

Description: The times taken for the whole operation of estimating the wetted surface at three different towing speeds, including my own contribution as an operator, are given in TABLE 5.4

TABLE 5.4. TIMING RESULTS

knots	*seconds*
20 knots	196
25 knots	256
30 knots	245

Chapter 6

Discussion

Alternative Approaches

At the beginning of this thesis I considered the alternative strategies of importing image analysis facilities into RHINO, or implementing 3D geometry facilities in MATLAB. It must be said that the chosen approach proved considerably more difficult than anticipated. Understanding of IGES2MATLAB software was mandatory to develop this thesis, and it was the hardest and the most time-consuming part of this research. There were also low-level tasks e.g., curve projection, meshing and surface splitting, which turned out to be difficult. On the other hand, carrying out the research has given me a geometrical insight which I would never have obtained, had I relied on the RHINO CAD system. Moreover the range of MATLAB facilities I have exploited in the research is so wide that I suspect the problems could have been equally hard choosing the alternative strategy of importing image analysis facilities into RHINO.

My solution has two important advantages. Firstly, it provides a framework for using improved image analysis techniques. The best example is the possibility to project points from other viewpoints to improve waterline tracing. Also, in a dynamically changing situation, a changing waterline could possibly be followed automatically over time. All this would be much harder to test in an environment other than MATLAB, at least for research and development purposes. Secondly my method offers an interface to other CAD systems than Rhinoceros.

Development Status

A system has been developed which could be used for experimental testing. A large body of code is available for further experimental work. There is a good basis to build on, and the research is anchored in well-understood computer technology (NURBS, snakes, etc).

Programming issues

Where execution time is considered excessive, MATLAB operations could be speeded up using “embedded code” facility. The splitting and meshing algorithms are already implemented in C/C++, so could readily be incorporated.

Future Work

Overview: This research focused on the geometric and image analysis aspects of the problem. Other aspects were tackled and discussed, but there is a chance for improvement and development.

Image registration
Unlike the current manual methods, the computer-assisted method is extremely sensitive to accurate registration of each camera image used to estimate the wetted surface of the hull. The method used in this research was registration of individual camera images with the Rhinoceros hull specification. This method needs to be investigated further, using custom-pained registration marks on the hull, and correcting for possible camera aberrations. If accuracy is still unsatisfactory, multiple-view registration will need to be considered, in which the unknowns are the position and attitude of the hull, the timing error of the underwater camera, and possibly the sight-line directions for the above-water camera.

Registration points on the model hull could almost certainly be detected automatically, particularly if they were chosen to stand out on one of the colour separations of the camera images.

Image analysis and waterline edition
From the point of view of the image analysis for the waterline delineation. The snake algorithm could be refined to take account of discontinuities in direction where the waterline crosses surface boundaries. Information from other views could be used to constrain the snake to pass through certain points.

Geometry
The surface splitting algorithm could be refined by executing image space allowing for multiple intersections. An adaptive meshing algorithm should also be incorporated to the system (for example, the one used in OpenSG).

Implementation
There is opportunity to build a GUI using the set of functions programmed in this project.

Chapter 7

Conclusions

General: This research has developed an effective method to compute the area of a 3D surface given camera images and the CAD model. Both the method of computer-assisted waterline tracing proposed here, and the geometrical interface between camera images and CAD model of the hull proved successful. The research has shown how a small number of points on the waterline selected by the operator can be combined with edge detection and a snake algorithm to delineate waterline sections. The waterline is stored as a NURBS curve defined in the surface of the vessel, so can be projected and modified by moving its control points from any camera viewpoint.

Wetted area estimates using the thesis method have been compared with manual estimates by SSPA showing a few percent difference. Part of this difference seems to be due to waterline tracing, and part to image registration – an operation which is less critical for the manual estimates.

Registration
Registration of the underwater images was accurate enough for successful back-projection from camera image to hull surface. Registration of the above-water (stern) camera image allowed the waterline to be plotted but insufficient registration points were not available for accurate back-projection.

Image analysis
The waterline tracing algorithm proved robust but needs testing on a wider range of images. Many variants of the snake approach were investigated in the course of the research, and some of these deserve further investigation.

Geometry
The method of back-projecting the waterline onto the hull surface proved to be fast and reliable, using the underwater camera images. Even though it is an approximation dependent on the size of the triangles used in meshing, there was a good agreement between the Rhinoceros and MATLAB measurements. While the method was demonstrated to be very accurate using synthetic data, there was a ± 5% difference between experimental area estimates and the estimates made manually by SSPA.

The current surface splitting algorithm does not take into account many possible hull shapes, although these are unlikely in these types of vessels. Alternative algorithms have been considered, but it was not possible to implement them in the research time available.

References

[1] SSPA. [Online]. http://www.sspa.se/research/projects/hydro-testing-alliance-hta

[2] Hydro Testin Alliance - Network of Excellence. [Online]. http://www.hta-noe.eu/

[3] Les Piegl and Wayne Tiller, *The NURBS Book*.: Springer-Verlag, 2nd Edition, 1997.

[4] A. Ryberg, A-K Christiansson, and K. Eriksson, "Accuracy Investigation of a Vision Based System for Pose Measurements".

[5] A. Ryberg, A-K. Christiansson, K. Eriksson, and B. Lennartson, "A new Camera Model and Algorithms for improved Accuracy and Convergence in Pose Calculations".

[6] A. Ryberg, A-K. Christiansson, B. Lennartson, and K. Eriksson, "A new Camera Model for Higher Accuracy Pose Calculations," 2006.

[7] Adrian Biran, *Ship Hydrostatics and Stability*.: Elsevier, 2003.

[8] Per Bergström. (2008) Mathworks. [Online]. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=13253&objectType=file

[9] Mark Spink. NURBS Toolbox for Matlab. [Online]. http://www.aria.uklinux.net/nurbs.php3

[10] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image Processing, Analysis, and Machine Vision, 3rd Edition*.: Thomson, 2008.

[11] Malcolm Slaney. (2009, February) Malcolm Slaney Homepage. [Online]. http://cobweb.ecn.purdue.edu/~malcolm/interval/1995-017/

[12] Peter Hultqvist and Hamidreza Taghavi, "Hull waterline detection. Image analysis project HT2007," Göteborg, 2007.

[13] JPR8, "Wetted Surface Estimation," in *Minutes, JPR8*, Holland, 2007.

[14] McNeel. Rhinoceros NURBS Modelling for windows. [Online]. http://www.rhino3d.com/

[15] D. F. Rogers, *An introduction to NURBS - with Historical Perspective*.: Morgan Kaufnann Publishers, 2001.

[16] Richard F. Riesenfeld, Gershon Elber Elaine Cohen, *Geometric modeling with splines : an introduction*.: AK Peters, cop. 2001.

[17] US PRO. (1996) What is IGES5.x for download. [Online]. http://www.iges5x.org/archives/version5x/

[18] Per Bergström, "Computational Methods for ON-Line Shape Inspection," Luleå University of Technology, Luleå, Licentiate 2009.

[19] Ming Xie, *Rundamentals of Robotics. Linking perception to action*. Singapore-MIT Alliance & Nanyang Technological University, Singapore: World Scientific, 2002.

[20] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision. 2nd Ed*.: Cambridge, 2003.

[21] C.-K. A User Guide to the Surface Subsystem of DesignMentor Schene. Design Mentor.

[Online]. http://lumimath.univ-mrs.fr/~jlm/cours/DMmanuals/surface/

[22] The IGES 5.x Preservation Society (IPS) HomePage. [Online]. http://www.iges5x.org/

[23] Wikipedia. (2008, July) IGES. [Online]. http://en.wikipedia.org/wiki/IGES

[24] Les A Piegl and Wayne Tillert, "Geometry-based triangulation of trimmed NURBS surfaces," *Computer-Aided Design*, vol. 30, no. 1, 1996.

[25] Bernd Hamann and Po-Yu Tsai, "A Tesselation algorithm for the representation of trimmed NURBS surfaces with arbitray trimming curves," vol. 28, no. 6/7, 1996.

[26] Jonathan Richard Shewchuk. A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. [Online]. http://www.cs.cmu.edu/~quake/triangle.html

[27] Darren Engwirda. MATLAB CENTRAL. [Online]. http://www.mathworks.com/matlabcentral/fileexchange/10307

[28] The University of Alabama at Birmingham Birminghan. Enabling Technology Laboratory at UAB. [Online]. http://me-wiki.eng.uab.edu/etlab/?page_id=261

[29] Ákos Balázs, Michael Guthe, and Reinhard Klein, "Efficient trimmed NURBS tessellation," vol. 12, no. 1-3, 2004.

[30] (2009, Jan) OpenSG. [Online]. http://opensg.vrsource.org/trac

[31] Alan Watt and Fabio Policarpo, *3D Games, Real-time Rendering and software Technology. Volume One and Two*.: Addison-Wesly, 2003.

[32] Anil. K. Jain, *Fundamentals of digital Image processing*.: Prentice Hall, 1988.

[33] Amir A. Amini, Terry E. Weymouth, and Ramesh C. Jain, "Using Dynamic Programming for Solving Variational Problems in Vision," vol. 12, no. 9, 1990.

[34] J.B. Antoine Maintz and Max A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1-36, 1998.

[35] Emanuele Trucco and Alessandro Verri, "Camera Parameters from the Projection Matrix," in *Introductory Techniques for 3-D Computer Vision*.: Prentice Hall, 1998, ch. 6, pp. 132-133.

[36] Richard Hartley and Richard Zisserman, "Basic equations," in *Multiple View Geometry*., ch. 7, pp. 178-179.

[37] (2009, Feb.) Wolfram. [Online]. http://mathworld.wolfram.com/NewtonsMethod.html

[38] Les Piegl and Wayne Tiller, "Point inversion and Projection for Curves and Surfaces," in *The NURBS Book,2nd Edition*, Springer-Verlag, Ed., 1997, ch. 6, pp. 229-234.

[39] Dirk-Jan Kroon. (2007, September) Matlab Central. [Online]. http://www.mathworks.com/matlabcentral/fileexchange/16448

[40] I. Reid, A. Zisserman A. Criminisi. A Plane measuring device. [Online]. http://www.robots.ox.ac.uk/~vgg/presentations/bmvc97/criminispaper/planedev.html

[41] Tomas Akenine-Möller and Eric Haines, *Real-Time Rendering, 2nd Edition*. Natick, Massachusetts: AK Peters, 2002.

[42] 2007 Robert McNeel & Assoc. OpenNURBS initiative. [Online].

http://www.opennurbs.org/

[43] Per Bergström and Ove Edlund, "Robust Surface Registration for Optical Single-Shot Measurements".

[44] Per Bergström, Ove Edlund, and Inge Söderkvist, "A method suitable for reiterated matching of surfaces," 2007.

[45] Mangesh P. Bhandarkar, Blair Downie, Martin Hardwick, and Rakesh Nagi, "Migrating from IGES to STEP: one to one translation of IGES," *Computers in Inudustry*, vol. 41, no. 261-277, 2000.

[46] A. Criminisi, I. Reid, and A. Zisserman, "Single View Metrology".

[47] M. Pauline Barker Donald Hearn, *Computer Graphics. C Version*.: Prentice-Hall International, Inc, 2nd Ed. 1997.

[48] Akemi Gálvez and Alberto Iglesias, "Numerical-Symbolic MATLAB Toolbox for Computer Graphics and Differential Geometry," vol. Volume 3482, 2005.

[49] Simon Gilles, Andrew Zisserman, and Andrew W. Fitzgibbon, "Markerless Tracking using Planar Structures in the Scene," *Proc. International Smposium on Augmented Reality*, vol. 120-128, 2000.

[50] Sankarappan Gopalsamy, Douglas H. Ross, Yasushi Ito, and Alan M. Shih, "Structured Grid Generation over NURBS and Facetted Surface Patches by Reparametrization," University of Alabama at Birmingham, AL, U.S.A,.

[51] David Jonsson, "Visualization of CAD Models - NURBS vs. Subdivision," Umeå, 8th July 2005.

[52] McNeel. Rhino Scripting Resources. [Online]. http://blog.rhino3d.com/2008/05/rhino-scripting-resources.html

[53] Perspective Transform Estimation. [Online]. http://alumni.media.mit.edu/~cwren/interpolator/

[54] Leslie A Piegl andArnaud M Richard, "Tessellating trimmed NURBS surfaces," vol. 27, no. 1, 1995.

[55] Bruce Simpson. How efficient are Delaunay Refined Meshes? An Empirical Study.

[56] Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherill, *Handbook of grid generation*.: CRC Press, 1999.

[57] TzuYi Yu and Bharat K. Soni, "Application of NURBS in numerical grid generation," *Computer-Aided Design*, vol. 27, no. 2, 1995.

[58] TrascenData. (2008, March) TrascenData. [Online]. www.transcendata.com/support/cadfix/faq/IGESexport.pdf

Appendix A

Estimation of the camera matrix

The relation between the 3-D coordinates (X, Y, Z) of a point in space and the 2-D coordinates (x′, y′) of its projection on the image plane can be written as [35]:

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.1}$$

with

$$\begin{aligned} x' &= \frac{x}{w} = \frac{h_{11}X + h_{12}Y + h_{13}Z + h_{14}}{h_{31}X + h_{32}Y + h_{33}Z + h_{34}} \\ y' &= \frac{y}{w} = \frac{h_{11}X + h_{12}Y + h_{13}Z + h_{14}}{h_{31}X + h_{32}Y + h_{33}Z + h_{34}} \end{aligned} \tag{A.2}$$

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \end{pmatrix} \tag{A.3}$$

where a 3x4 matrix H is called projection matrix. The matrix is defined up to an arbitrary scale factor. Assume that we have a number of point correspondences. $(x_i', y_i') \leftrightarrow (X_i, Y_i, Z_i), i = 1 \ldots N$ Then the entries of the H matrix can be determined through a linear system formed by writing Eq. A.1. for space–image point matches (we need at least 6 space-image point matches). This will lead to the following equation:

$$A \cdot h = 0, \tag{A.4}$$

with

$$A = \begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1'X_1 & -x_1'Y_1 & -x_1'Z_1 & -x_1' \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1'X_1 & -y_1'Y_1 & -y_1'Z_1 & -y_1' \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2'X_2 & -x_2'Y_2 & -x_2'Z_2 & -x_2' \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2'X_2 & -y_2'Y_2 & -y_2'Z_2 & -y_2' \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_N'X_N & -x_N'Y_N & -x_N'Z_N & -x_N' \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_N'X_N & -y_N'Y_N & -y_N'Z_N & -y_N' \end{pmatrix} \tag{A.5}$$

The vector h can be found using singular value decomposition of matrix A, as the singular vector corresponding to the zero (in practice the smallest) singular value of A. The entries of H are obtained up to an unknown scale factor.

Appendix B

Camera Specification

All three cameras attached to the platform were analogue and acquired pictures were scanned with a FUJI PHOTO FILM CO., Ltd. SP-2000.

Pixel dimension X, Y	1840-1232
Resolution (inch)	72, 72
Color spcae	sRGB
FileSource	DSC
Orientation	normal
Date	2007-02-03 T 13.51
Model	Fdi V 4.5/Frontier 350/370 7.6

Underwater camera[5]

Model	Nikon D1X
Pixel dimension X, Y	3008, 1960
Resolution (inch)	300, 300
Cell size	5.93 × 11.89 μm
Focal length	35 mm
F-stop	f/11 F-stop (lens)
Maximun aperture value	f/4.4 (lens)
Compress bits per pixel	SRGL
Exposure program	Manual
Sensor size	23.7x15.15mm (3.67cm^2)
Pixel density	1.4MP/cm^2

[5] http://www.dpreview.com/reviews/specs/Nikon/nikon_d1x.asp