Master Thesis

Automatic Optical Identification and Classification for the Industrial Processing of Fish

Uwe Meier
uwemei@gmail.com

Supervisor:

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary

Institute for Automation
Montanuniversität Leoben, Austria

Leoben, March 7, 2011

Affidavit

I hereby declare that this master thesis has been written only by me and without any assistance from third parties. Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Leoben, March 7, 2011 _____

Uwe Meier

Acknowledgements

I would like to thank my supervisor o.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary for his support and guidance during this work and during a part of my study. His door was always open, when I needed some advice.

Furthermore, I want to thank the whole staff at the Institute for Automation in Leoben, who supported me by some means or other. Especially, I am grateful to Doris Widek for her support with countless small things.

I also want to thank Tawepol Suesut, who initiated this thesis and provided the necessary data for the first part.

A special gratitude to my colleagues, Andre Pura and Peter Aldrian, with whom I had a great time during my studies and also during our summer projects.

Thanks to my family, without whose financial and moral support my studies in Leoben would not have been possible.

Last but not least, I am particularly grateful to Magdalena Niessner, who motivated me more than any other person to finish this thesis.

Kurzfassung

Diese Masterarbeit befasst sich mit der optischen Vermessung und Klassifizierung von Fischen, welche Voraussetzungen für eine automatisierte Sortierung darstellen. Zwei Kernpunkte werden in dieser Arbeit untersucht: Der erste Teil beschäftigt sich mit der geometrischen Vermessung mithilfe des Lichtschnittverfahrens. Mit dieser Methode sollen Größe und Volumen des Fisches bestimmt werden, wobei Annahmen bezüglich der Symmetrie des Fisches getroffen werden;

Der zweite Teil befasst sich mit den Charakteristiken von Umrissen zur Identifizierung verschiedener Fischarten. Sowohl Elliptical Fourier Descriptors, Cyclic Polynomial Descriptors, als auch die Krümmungsradien dieser Konturen werden untersucht. Es wird gezeigt, dass eine Klassifizierung allein anhand dieser Daten nicht möglich ist, da die Form und das Aussehen selbst innerhalb einer Fischart stark variieren kann. Allerdings kann die Methode, die zum registrieren der Krümmungen zweier Fische erstellt wurde, benutzt werden, z.B. um die Konturen von Walzprofilen mit den dazugehörigen CAD-Daten zu vergleichen.

Alle vorgeschlagenen und untersuchten Methoden wurden in *MATLAB* ® implementiert und mit an realen Fischen gemessenen Daten getestet.

Abstract

This thesis deals with the optical measurement and classification of fish, which is required as a prerequisite to enable automatic sorting. Two main issues are investigated in this thesis: Part I deals with the geometric measurement based on light sectioning. With this method the size and volume of the fish should be determined, whereby assumptions are made with respect to the symmetry of the fish;

Part II deals with profile description with the aim of identifying different species of fish. Elliptical Fourier Descriptors, Cyclic Polynomial Descriptors as well as the radius of curvature of this contours are analyzed. It is shown, that a classification based solely on this data is not possible, since even forms and appearances of fish within one species differ widely. However, the method applied to register two curvatures can be used e.g. to compare the contours of a rolling profile with its corresponding CAD-data.

All the proposed and investigated methods have been implemented in *MATLAB* ® and tested with data measured on real fish.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Cycle time is a crucial factor in the processing of perishable goods such as fish. However, a manual classification and measurement (e.g. for cannery processing) of these products is a very laborious task, thus costly. In addition, perishable goods are often sold at different prices depending on their size, weight or grade, making a highly reliable quality-control important. Poor sorting quality can lead to reduced earnings due to unfulfilled demands of the customer. Machines and also computer vision are faster than people but can not keep up with the human judgment yet [15]. Nonetheless automatic optical inspection is emerging as a means of classifying fish and other perishable goods due to the achieved reduction of processing time and increased productivity.

The aim of this work is to investigate optical arrangements and mathematical methods for the inspection and classification of foodstuffs — in particular fish. To do so several tasks were assigned beforehand:

1. Set up test measurement systems, including the optical arrangements;
2. Research possible evaluation strategies and/or algorithms and select plausible approaches;
3. Implement the selected algorithms in *MATLAB* ® and compare the results;
4. Document the results in a scientific manner.

The methods proposed in this thesis — although analysed off-line — represent basic investigations to a real-time image processing for perishable goods and can be applied in future experimental setups.

First attempts to apply computer vision to the food industry have already been researched in the 1990s [5]. Cadrin and Friedland [4] have shown the use of morphometric analysis to identify different fish stock of the same species. There have also been introduced methods to classify fish in situ (i.e. alive) [3, 11]. Strachan [21] has proposed an algorithm for length measurement with an error of 1% for oriented and an error of 3% for non oriented fish. Performing an analysis of a fish's contour in a still image based on moment invariants a succsessrate between 86% and 100% was achieved for three fish species [24]. Omar and de Silva describe a way for an automated portion control for canned fish [20, 9]. However, calculating the weight distribution along the fish by using a water replacement method seems to be very time-consuming. The use of Light Sectioning in order to classify fish was introduced in 2001 [7], combined with a neuronal network, that did not process the shape of the fish, but rather a pseudo image containing the width and maximal distortion along the length of the fish. Out of 251 fishes from 6 species, 5 were classified incorrectly. Using canonical discriminant analysis, White et al. [1] achieved a sorting reliability for seven species of fish up to 99.8%. They used 10 shape variables (equidistant width measurements) and 114 color variables, obtained from the RGB mean values of 38 grid sections.

However, no literature was found, that tries to use the three-dimensional shape of an object — which can be obtained via light sectioning — for classification or dimension measurement. Likewise no one used the contour of the fish itself.

1.1 Content

This thesis is divided into two parts: The first section deals with the artificial vision based method of light sectioning for the dimension measurement of fish. The method is described in detail as well as what is needed to get the corresponding

real-world values. Also, the mathematical methods that found use in this part are explained (e.g. Moments, Curvature) . After the analysis of the different approaches to complete the fish's geometry — based on the three-dimensional surface retrieved from the light sectioning — the final results are presented. Furthermore, it is shown, where there is potential for further improvements;

The second part's aim is to demonstrate an approach to identify fish using image processing and local polynomial spectra. It is shown how to binarize a greyscale image as well as how to use spectra for preprocessing data (e.g. smoothing) and their mathematical background. After presenting the used approach, it is shown, that using the contour exclusively to discriminate between fish species failed given that the characteristics of attributes can vary even within a species. An alternative field of application for the proposed method is given.

Part I

Measurement of Fish Dimensions using the Light Sectioning Method

Chapter 2

Problem Statement

Three-dimensional data is of vital importance in order to gain information about the surface and the volume of a fish. Light sectioning is a technique to obtain such data via computer vision in an optical way. A digital camera records two dimensional pictures of the object being measured. An image is a two-dimensional mapping of three-dimensional objects. Hence, it is necessary to reconstruct the missing dimension. A laser plane, that "cuts" the object delivers a distorted height information. If the position and the angle of the camera are known, this height information in the distorted image can be mapped onto a fixed plane representing a slice of the object. When moving the object or the laser, several images — thus several slices — add up to a three-dimensional representation of the object again. Another problem arises from the different units of the real world and image coordinates. In images, lengths are defined in terms of pixel, whereas the real world dimensions use metric (m) or imperial (inch) units. Hence a calibration target, whose dimensions are known a-priori, as well as a procedure to calibrate the system are necessary.

2.1 Principle of the light sectioning method

The light sectioning method is a technique often used for optical measurement of objects. A light plane, usually generated by a laser with a cylindrical lens, is pro-

jected onto the object. A camera views the object and the laser light scattered from the surface from a different direction. If the arrangement of the laser and the camera is known, the position of every point of the intersection can be assigned explicitly to a position in a real-world coordinate frame. A three-dimensional model of the object can be created sequentially when the object is moved, e.g. in a straight line crossing the light plane . As shown in Figure 2.1 only the top portion of the object can be measured. If the object is symmetric, the whole profile can be calculated using appropriate mathematical models. Otherwise additional lasers and cameras have to be added to cover the whole cross-section. In order to eliminate the influence of the ambient light, an optical narrow-band interference filter, which lets the wavelength emitted by the laser pass, is placed in front of the camera. In this manner, the laser is the main light source which impinges upon the camera sensor.

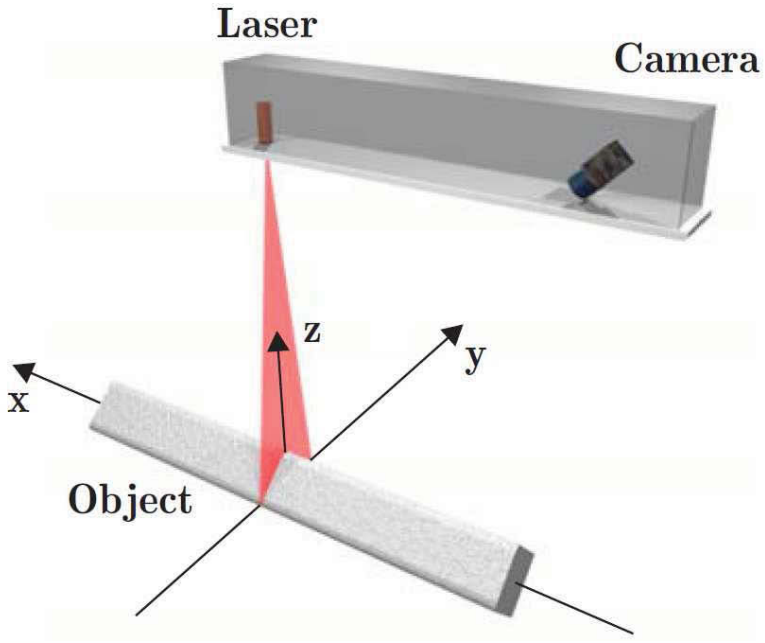


Figure 2.1: Principle of light sectioning [13].

For each section, the profile has to be extracted from the image using a line detection algorithm. Lining up the single two-dimensional profiles sequentially yields three-dimensional data representing the whole object.

2.2 Calibration methods for light sectioning

To extract the height information of an object from an intensity image there are two different methods: An algebraic approach uses the trigonometric relations between the light source, the object and the camera; the geometry can be seen in Figure 2.2. According to [10] the range r can be calculated as

$$r = \frac{B\,(f\,\tan(\alpha) - s)\,\cos(\alpha)}{\frac{f}{\cos(\alpha)} - (f\,\tan(\alpha) - s)\,\sin(\alpha)} = B\,\frac{f\,\tan(\alpha) - s}{f + s\,\tan(\alpha)}\,, \tag{2.1}$$

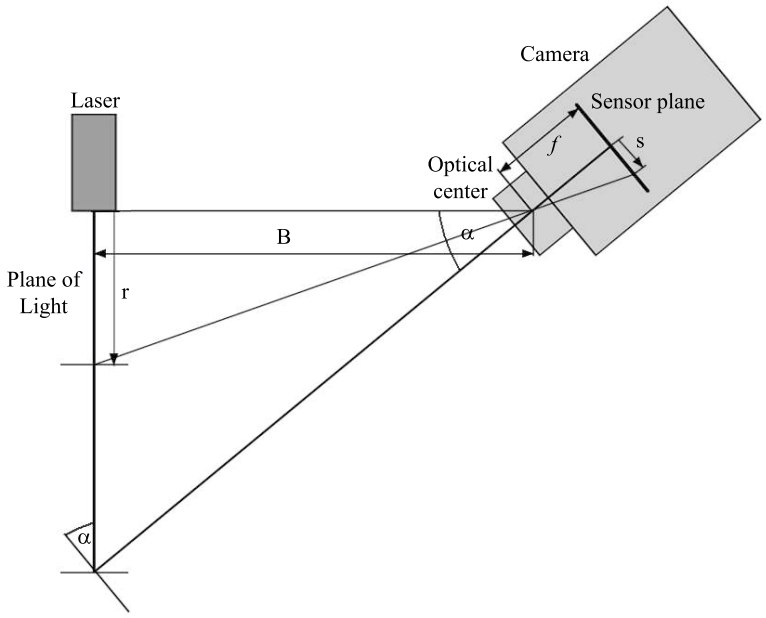


Figure 2.2: Geometric setup for light sectioning.

where f represents the focal length (i.e. normal distance between the optical center and the sensor plane) of the camera, B gives the distance between laser and optical center and s is the light impact's position on the sensor. Further, the world coordinate width x can be linked to the sensor coordinate t via the equation

$$x = \frac{-t\,a}{f}\,, \tag{2.2}$$

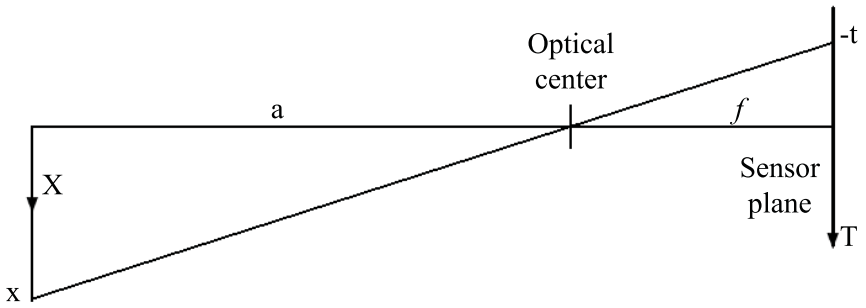


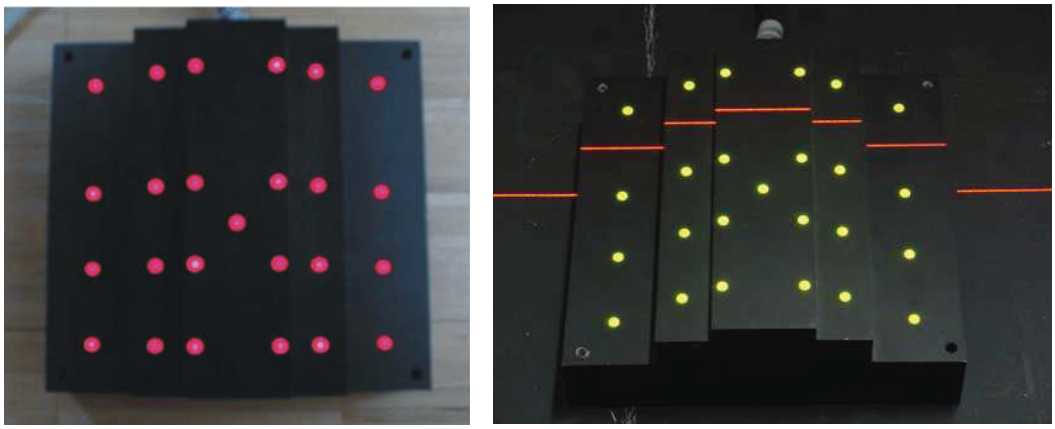
Figure 2.3: Geometry to calculate the width.

using the trigonometric relation shown in Figure 2.3, which can also be expressed with the system's setup paramaters as

$$x = \frac{-t\,B}{f\,\cos(\alpha) + s\,\sin(\alpha)}\,. \tag{2.3}$$

Note, that the above equations are only valid when the optical axis of the camera is normal to the the sensor plane.

However, in order to use this calibration method the exact setup parameters — especially the focal length of the camera — have to be known. Another approach, described in [6], is based on projective geometry. One single camera image that shows a simple calibration target with known geometry is needed as input, information about the camera's parameters (such as focal length or resolution) are not necessary. The calibration object needs at least two different planes to obtain the laser plane, as well as four or more calibration marks on each of them so they can be identified (a feasible realization can be seen in Figure 2.4). The homogeneous transformations of these planes are fully determined, if at least 4 corresponding points of each plane are known in the real world and the camera image. The plane of light can be determined, given an image with the intersection between the laser and a calibration target with at least two known planes. Thus, the homogeneous transformation relating the plane of the camera image and the real plane of light can be defined explicitly.



(a) Calibration target. (b) Image of the calibration process.

Figure 2.4: Calibration target for light sectioning. At least two planes with at least four calibration points each suffice to determine the position of the plane of light.

A projection H of a homogeneous point $\boldsymbol{p_0}$ on a plane to a point $\boldsymbol{p_r}$ on another plane can be defined as,

$$\boldsymbol{p_r} = \mathsf{H}\,\boldsymbol{p_0}\,, \tag{2.4}$$

where $\boldsymbol{p_r}$ and $\boldsymbol{p_0}$ are in homogeneous coordinates,

$$\boldsymbol{p_r} = \begin{bmatrix} x_r \\ y_r \\ w_r \end{bmatrix} \text{ and } \boldsymbol{p_0} = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}\,. \tag{2.5}$$

The homogeneous transformation called the homography is defined as

$$\mathsf{H} \triangleq \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\,. \tag{2.6}$$

The matrix H has nine degrees of freedom. However, given that $\|h\|_2 = 1$, one of the nine parameters h_{11} to h_{33} can be calculated by the other parameters. Thus, four corresponding points in both planes are sufficient to calculate the other eight

entries. Splitting these points into their affine x- and y-components x_r^a and y_r^a by dividing the projected points $\boldsymbol{p_r}$ by w_r gives

$$\begin{aligned} x_r^a &= \frac{x_r}{w_r} = \frac{h_{11}\,x_0 + h_{12}\,y_0 + h_{13}}{h_{31}\,x_0 + h_{32}\,y_0 + h_{33}}\,, \\ y_r^a &= \frac{y_r}{w_r} = \frac{h_{21}\,x_0 + h_{22}\,y_0 + h_{23}}{h_{31}\,x_0 + h_{32}\,y_0 + h_{33}}\,. \end{aligned} \tag{2.7}$$

This can be rewritten as,

$$\begin{aligned} -h_{11}x_0 - h_{12}y_0 - h_{13} + h_{21}0 + h_{22}0 + h_{23}0 + x_r^a\,(h_{11}x_0 + h_{12}y_0 + h_{13}) = 0, \\ h_{11}0 + h_{12}0 + h_{13}0 - h_{21}x_0 - h_{22}y_0 - h_{23} + x_r^a\,(h_{11}x_0 + h_{12}y_0 + h_{13}) = 0\,. \end{aligned} \tag{2.8}$$

Combining these equations for several points into a matrix we get

$$\begin{bmatrix} -x_0(1) & -y_0(1) & -1 & 0 & 0 & 0 & x_r^a(1)x_0(1) & x_r^a(1)y_0(1) & x_r^a(1) \\ 0 & 0 & 0 & -x_0(1) & -y_0(1) & -1 & y_r^a(1)x_0(1) & y_r^a(1)y_0(1) & y_r^a(1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -x_0(n) & -y_0(n) & -1 & 0 & 0 & 0 & x_r^a(n)x_0(n) & x_r^a(n)y_0(n) & x_r^a(n) \\ 0 & 0 & 0 & -x_0(n) & -y_0(n) & -1 & y_r^a(n)x_0(n) & y_r^a(n)y_0(n) & y_r^a(n) \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0\,, \tag{2.9}$$

or in short:

$$\mathsf{G}\,\boldsymbol{h} = 0\,. \tag{2.10}$$

If the number of points n is smaller than four the equation is under-determined and has no unique solution. For $n \geq 4$ the Equation 2.9 is an (over-)determined linear equation system which can be solved with respect to $\boldsymbol{h}$ by applying singular value decomposition (SVD). Recombining the single values of $\boldsymbol{h}$ gives the homography matrix H. A faster algorithm than the appliance of SVD alone is presented in [8].

With this, the homogeneous transformation $\mathsf{H}_{\mathsf{rw}-\mathsf{l}}$ between the real world coordinate frame and the laser coordinate frame and the homogeneous projection matrix $\mathsf{H}_{\mathsf{l}-\mathsf{c}}$ between the laser plane and the image can be calculated. During measurement the found image points can be mapped onto the laser plane using $\mathsf{H}_{\mathsf{l}-\mathsf{c}}$ and then transformed into the real world coordinate frame via $\mathsf{H}_{\mathsf{rw}-\mathsf{l}}$.

2.3 Limitations of the light sectioning method

A limiting factor when using light sectioning is the problem of occlusion. This may happen for two reasons. Either the laser light does not create a contour in an area that can be viewed by the camera. This so called laser occlusion occurs due to the object's structure blocking the laser plane from reaching other parts of the object. Or the camera can not display the whole intersecting line because the object is blocking the view onto parts of the line [10]. In order to circumvent these occlusions multiple camera and/or laser systems are needed to cover the whole object. However, more systems increase the complexity because of the need for individual coordinate systems and their registration to one real world system.

Another restriction is the possible sample rate a system can achieve. Not only is it necessary to discretize the continuous surface of the object, but also to buffer the usually large amount of data acquired with every image. A possible technique to increase the sampling rate is to reduce the sensor resolution, which leads to a loss of information, though. Another way is to implement a field programmable gate array (FPGA) inside the used camera, which transforms the image data into range data that is transferred to a programmable logic controller (PLC). Sampling rates up to 20000 frames per second are possible; however, this implementation is linked to higher costs [13].

Chapter 3

Mathematical Methods

The mathematical principles described here are the foundation for all further approaches in the following chapters. If not stated differently, the citations hold entirely for the individual subchapters.

3.1 Calculating moments

The nth moment about a point a is defined as [23],

$$\mu_n(a) = \left\langle (x-a)^n \right\rangle = \sum (x-a)^n P(x) \,, \tag{3.1}$$

of a known discrete distribution $P(x)$. If the underlying distribution is not known, the first order raw moment (i.e. $n = 1$ and $a = 0$) may be computed as the sample mean:

$$\bar{x} = \sum_{i=1}^{n} x_i \frac{1}{n} = \frac{1}{n} \sum_{i=1}^{n} x_i \,, \tag{3.2}$$

for a set of $\{x_i\}$ of n values. This moment is also called the arithmetic mean, denoted $\mu = \bar{x} = \langle x \rangle$. Since the sum over a discrete distribution equals one, $P(x)$ can be

replaced by a weighting factor if the equation is divided by the sum of all weights in order to normalize the weight:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i \, w_i}{\sum_{i=1}^{n} w_i} \, . \tag{3.3}$$

Given a two dimensional set of samples $\{(x_i, y_i)\}$ the weighted arithmetic means in this two directions are described as:

$$\bar{x} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \, w_{i,j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{i,j}} \quad \text{and} \quad \bar{y} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} y_{i,j} \, w_{i,j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{i,j}} \, . \tag{3.4}$$

The variance σ^2 for a discrete distribution is the second order central moment (i.e. moment about the mean) and is defined as,

$$\sigma^2 = \sum_{i=1}^{n} (x_i - \mu)^n \, P(x_i) \, . \tag{3.5}$$

It is a measure of how far a set of $\{x_i\}$ diverges from their expectation value $\langle x \rangle$. The positive square root of σ^2 is called standard deviation. If the distribution is unknown, a sample variance can be computed as,

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \, . \tag{3.6}$$

Similar to Equation 3.3, a weighted variance can be calculated:

$$s^2 = \frac{\sum_{i=1}^{n} w_i \, (x_i - \bar{x})^2}{\sum_{i=1}^{n} w_i} \, . \tag{3.7}$$

3.2 Thresholding

The basic process to segment a given data set (e.g. an image) into two groups is called binarization and can be described as,

$$I_{bin}(i,j) = \begin{cases} 1 \text{ if } I(i,j) > T \\ 0 \text{ otherwise,} \end{cases} \tag{3.8}$$

where a threshold T divides the data in two and assigns the values 0 and 1 respectively. Dealing with a bimodal data-set (i.e. its histogram shows two clear and distinct peaks) several methods to identify the threshold can be used.

One way to model a bimodal distribution is by representing it as the sum of two Gaussian distributions [2],

$$y = I_p \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.9}$$

where I_p is the peak value, μ is the mean and σ is the standard deviation. The point of intersection between these two functions can be found by solving

$$I_{p1} \mathrm{e}^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} = I_{p2} \mathrm{e}^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}, \tag{3.10}$$

for x. Applying the natural logarithm, expanding and collecting for x yields,

$$\left(\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}\right) x^2 + \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right) x + \frac{\mu_2^2}{2\sigma_2^2} - \frac{\mu_1^2}{2\sigma_1^2} + \ln\left(\frac{I_{p1}}{I_{p2}}\right) = 0\,. \tag{3.11}$$

Only one solution of this simple quadratic equation lies between the peaks of the two Gaussian functions. The x-value of this solution is used as threshold.

3.3 Spectral filtering with local polynomial moments

The spectrum s of the data y is determined by computing the discrete equivalent of an integral transform [19]:

$$\boldsymbol{s} = \mathsf{B}_\mathsf{a}^+ \, \boldsymbol{y} \, , \tag{3.12}$$

where B_a^+ is the Moore-Penrose pseudo inverse,

$$\mathsf{B}_\mathsf{a}^+ \triangleq \left(\mathsf{B}^\mathsf{T} \mathsf{B} \right)^{-1} \mathsf{B}^\mathsf{T} \, . \tag{3.13}$$

Discrete basis functions, which can be calculated as above are Fourier, cosine, Gábor and Wavelets, as well as polynomial moments. A signal can be synthesised by the synthesis function B_s as,

$$\hat{\boldsymbol{y}} = \mathsf{B}_\mathsf{s} \, \boldsymbol{s} \, . \tag{3.14}$$

Generally B_s and B_a are fundamentally the same type of basis functions, evaluated at different nodes or of different degrees. The reconstruction of a signal is computed by combining Equation 3.12 and 3.14:

$$\hat{\boldsymbol{y}} = \mathsf{B}_\mathsf{s} \, \mathsf{B}_\mathsf{a}^+ \, \boldsymbol{y} \, , \tag{3.15}$$

where the reconstruction error r is defined as,

$$r = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \mathsf{B}_\mathsf{s} \, \mathsf{B}_\mathsf{a}^+ \, \boldsymbol{y} = \left(\mathsf{I} + \mathsf{B}_\mathsf{s} \, \mathsf{B}_\mathsf{a}^+ \right) \, \boldsymbol{y} \, . \tag{3.16}$$

Thus, perfect reconstruction only occurs if the projection onto the orthogonal complement $\mathsf{I} + \mathsf{B}_\mathsf{s}\mathsf{B}_\mathsf{a}^+ = 0$, which can be achieved by using a unitary and complete basis (i.e. $\mathsf{B}^\mathsf{T}\mathsf{B} = \mathsf{B}\mathsf{B}^\mathsf{T} = \mathsf{I}$) for analysis and reconstruction. This yields $\mathsf{B}_\mathsf{a} = \mathsf{B}_\mathsf{s} = \mathsf{B}$. The complex conjugate transpose of any complete unitary basis is its inverse, giving $\mathsf{B}_\mathsf{a}^+ = \mathsf{B}^\mathsf{T}$. Methods to orthogonalize matrices as well as a synthesis algorithm for a unitary discrete polynomial basis B are shown in [19].

In order to model spectral filtering [17], a filter F is applied to the transform

$$\boldsymbol{s_F} = \mathsf{F}\,\mathsf{B}^{\mathsf{T}}\,\boldsymbol{y}\,, \tag{3.17}$$

and the filtered signal $\boldsymbol{y_F}$ is synthesised:

$$\boldsymbol{y_F} = \mathsf{B}\,\boldsymbol{s_F} = \mathsf{B}\,\mathsf{F}\,\mathsf{B}^{\mathsf{T}}\,\boldsymbol{y}\,. \tag{3.18}$$

We can define $\mathsf{F} \triangleq \mathsf{G}\,\mathsf{G}^{\mathsf{T}}$ if the filter function is factorable. With $\mathsf{D} \triangleq \mathsf{B}\,\mathsf{G}$ we get:

$$\boldsymbol{y_F} = \mathsf{B}\,\mathsf{G}\,\mathsf{B}^{\mathsf{T}}\,\mathsf{G}^{\mathsf{T}}\,\boldsymbol{y} = \mathsf{D}\,\mathsf{D}^{\mathsf{T}}\,\boldsymbol{y}\,, \tag{3.19}$$

where $\mathsf{P} = \mathsf{D}\,\mathsf{D}^{\mathsf{T}}$ has the dimension $n \times n$ for a set of n points and is an orthogonal projection onto the filtering basis function. Thus, every filtered point $\boldsymbol{y_F}$ is a linear combination of all input values.

When computing polynomial Savitzky-Goolay smoothing on a set of n data points, a support length l_s (i.e. the number of data points being used to smooth one single point) and a degree d is needed. In the case where $l_s = n$ global smoothing is performed. To get the projection matrix $\mathsf{P_C}$ for all points, a unitary basis B for l_s and d has to be computed, which gives the local projection matrix by $\mathsf{P} = \mathsf{B}\,\mathsf{B}^{\mathsf{T}}$ (this is an $l_s \times l_s$ matrix). The center row of P represents the projection at the center of the support $\boldsymbol{p_{x=0}}$(i.e. there are more than $(l_s - 1)\,/2$ data points left and right of the evaluated point). The rows below and above $\boldsymbol{p_{x=0}}$ represent the projection at the end and start of the data respectively. Thus, the top and bottom of P are placed at the start and end of $\mathsf{P_C}$ to generate the complete projection matrix. The center part of the matrix is filled diagonally with $\boldsymbol{p_{x=0}}$.

To get an approximation for the local derivatives of discrete data a similar method

to create a derivative matrix D is used. With the support length l_s (i.e. the number of data points being used to calculate the derivative) and the degree d a matrix N,

$$\mathsf{N} = \begin{bmatrix} -m^d & -m^{d-1} & \dots & -m^1 & 1 \\ -(m-1)^d & -(m-1)^{d-1} & \dots & -(m-1)^1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \\ (m-1)^d & (m-1)^{d-1} & \dots & (m-1)^1 & 1 \\ m^d & m^{d-1} & \dots & m^1 & 1 \end{bmatrix}, \tag{3.20}$$

is created, where $m = (l_s - 1)/2$. The Moore-Penrose pseudo inverse M of N is weighted according to the derivatives (i.e. m^d is multiplied by d and so on) and the last row is eliminated. The derivative matrix D for n data points has the dimension $n \times n$ and consist of three parts: the center part, which is filled diagonally with the last row of M; the top part, where the approximation is asymmetric to the right; the bottom part, where the approximation is asymmetric to the left. The bottom part $\mathsf{D_b}$ is computed by multiplying $\mathsf{D_b} = \mathsf{X}\,\mathsf{M}$, where

$$\mathsf{X} = \begin{bmatrix} 1^d & 1^{d-1} & \dots & 1^2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m^d & m^{d-1} & \dots & m^1 & 1 \end{bmatrix}. \tag{3.21}$$

The top part $\mathsf{D_t}$ is the same as $\mathsf{D_b}$ flipped horizontally and vertically.

3.4 Osculating circles

The osculating circle of a plane curve $C(x, y)$ at a given point $p_i = [x_i, y_i]$ is the circle passing through p_i with the same tangent as well as the same curvature as C at the

point p_i [2]. Figure 3.1 illustrates an example. Given a plane curve in Cartesian coordinates parametrically as $C(t) = [(x(t), y(t)]$, the curvature κ is

$$\kappa = \frac{x' y'' - y' x''}{(x'^2 + y'^2)^{3/2}}, \tag{3.22}$$

where derivatives are taken with respect to t:

$$x' = \frac{\mathrm{d}x}{\mathrm{d}t}, \quad y' = \frac{\mathrm{d}y}{\mathrm{d}t}, \quad x'' = \frac{\mathrm{d}^2x}{\mathrm{d}t} \quad \text{and} \quad y'' = \frac{\mathrm{d}^2y}{\mathrm{d}t}. \tag{3.23}$$

The radius $r(t)$ of the osculating circle in $p(t) = [(x(t), y(t)]$ — and thus the radius of the curve — is simply the radius of curvature:

$$r(t) = \frac{1}{|\kappa(t)|}. \tag{3.24}$$

The center (ξ, η) of the circle is given by,

$$\begin{aligned} \xi &= x - \frac{\left(x'^2 + y'^2\right) y'}{x' y'' - y' x''}, \\ \eta &= y + \frac{\left(x'^2 + y'^2\right) x'}{x' y'' - y' x''}. \end{aligned} \tag{3.25}$$

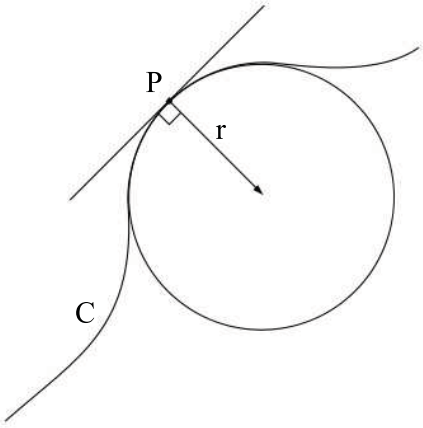


Figure 3.1: Osculating circle: Curve and circle have the same tangent as well as curvature in point P.

Chapter 4

Tested Approaches

In order to implement and test different approaches, the task was split into smaller parts. First, an algorithm to compute the fish's upper surface was implemented. For every frame of a video, showing a fish moving across a laser plane, the following steps were performed:

1. Extract the laser line[1] from the image;
2. Convert the laser line from the image coordinates to the real world coordinates and save the x- and y-values as vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$;
3. Separate $\boldsymbol{y}$ into two parts: $\boldsymbol{y}_f$, where the laser is reflected by the fish, and $\boldsymbol{y}_b$, where the laser is reflected by the background;
4. Patch holes in $\boldsymbol{x}$ and $\boldsymbol{y}_f$ to create a continuous surface for the fish.
5. Create a matrix X and concatenate $\boldsymbol{x}$, $\boldsymbol{y}_f$ and $\boldsymbol{y}_b$ of this frame with the ones from the previous frames in separate matrices Y, Z_f and Z_b, respectively;

The four matrices X, Y, Z_f and Z_b are the data, that were used to develop and test different approaches to complete the fish's surfaces (i.e. calculate the bottom surface, that can not be seen by the camera): simple mirroring of the upper surface;

[1] In this work, the term "laser line" always refers to the light reflected from an object, when it crosses a laser plane. The appearance of this reflection is not necessarily linear nor continuous but can be of any shape.

approximation of the cross section via ellipses or egg curves; mirroring with an additional extrapolation using the Taylor series; mirroring with an additional extrapolation using osculating circles. The different steps are described in the subsequent sections. The four matrices and the matrix Z_f^* (i.e. the matrix describing the bottom surface of the fish) were used to calculate the dimensions of the fish.

4.1 Extraction of the laser light

The input data is a video file that is separated in a set of single RGB-images (see Figure 4.1 for a sample image). To extract the desired information (i.e. the reflected laser light's position in the image) the intensity values of the single pixels must be evaluated, given that a pixel with an high value represents reflected laser light with a high intensity. Since the laser emits only red light the red component of the image was taken for further processing.

As a first test the pixel with the maximum intensity value of every column was taken as part of the laser line. As shown in Figure 4.2(a) this produced a rather scattered laser line due to the noise perturbed image. Thus, the image is smoothed with an averaging kernel A with the dimensions $m \times n$,

$$A = \frac{1}{m\,n} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}, \tag{4.1}$$

where the sum of the intensities of every pixel and its neighbours is divided by the number of pixels used. The position of the laser line in the image is more stable after convolution with the smoothing kernel. However, the accuracy is still bond to a pixel-level (i.e. integer values), creating steps, when the highest intensity of one column is located in a different row than the previous one. In addition, one single value gives no information about the structure of the laser line: whether there are more pixels with the found intensity value in each column; is the intensity location

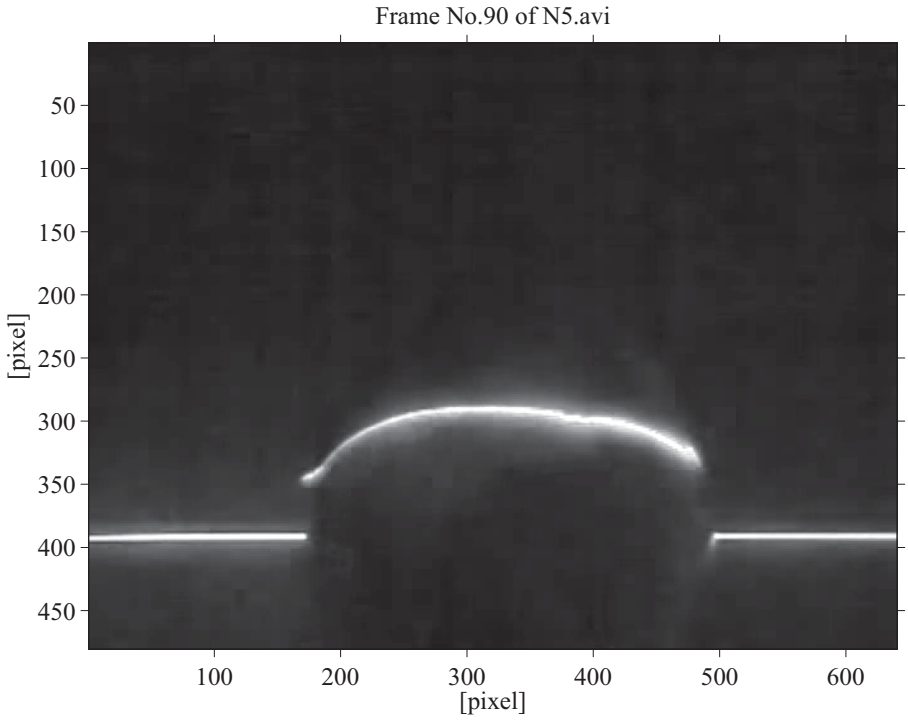


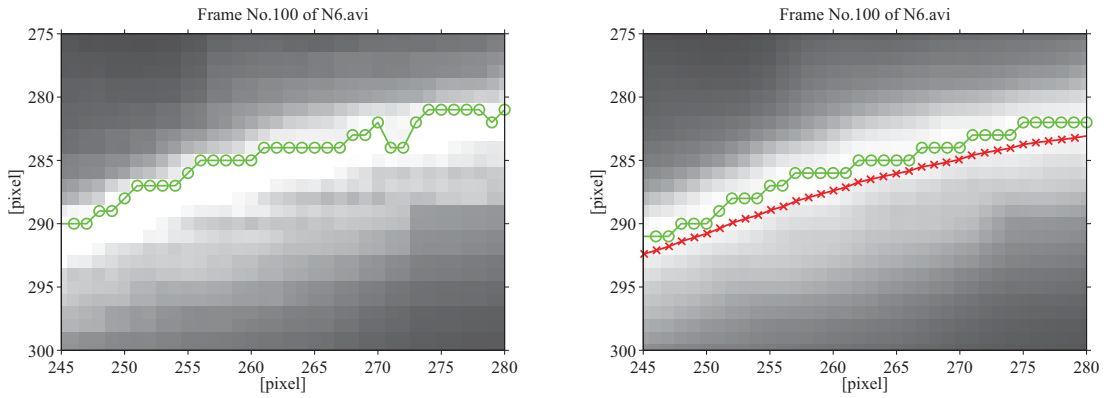
Figure 4.1: Image of a fish under a structured laser plane. The camera captures a high intensity (i.e. white areas in the image) where the laser intersects with an object. Due to the reflective character of the fishes surface, the laser light is scattered.

symmetrical or is there an edge; how far spreads the brighter area of the laser line until the intensity drops to a background level (i.e. the breadth of the laser line).

In order to include these aspects into the calculation of the position of the center of the line, the first intensity moment according to Equation 3.4 from Chapter 3.1 can be calculated:

$$x_i = \frac{\sum_{x=x_{max}-m}^{x_{max}+m} \sum_{y=y_{max}-n}^{y_{max}+n} x\, I^p(x,y)}{\sum_{x=x_{max}-m}^{x_{max}+m} \sum_{y=y_{max}-n}^{y_{max}+n} I^p(x,y)} \quad \text{and} \quad y_i = \frac{\sum_{x=x_{max}-m}^{x_{max}+m} \sum_{j=1}^{m} y\, I^p(x,y)}{\sum_{x=x_{max}-m}^{x_{max}+m} \sum_{y=y_{max}-n}^{y_{max}+n} I^p(x,y)}\,. \tag{4.2}$$

In this equation, x_i and y_i stand for the position of the point $\boldsymbol{p_i} = [x_i, y_i]$ of the new laser line of the ith column, and x_{max} and y_{max} stand for the position of the



(a) Line detection in the original image.

(b) Line detection in the smoothed image. A 3×3 averaging kernel was used.

Figure 4.2: Line detection in a gray level image. The green circles represent a detection using the intensity values of each column only. the red x's represent a line detection using a center of gravity calculation in both x- and y-direction.

maximum intensity of each column. $I^p(x, y)$ represents the intensity at the position (x, y) with a weighting factor p. The higher p, the stronger pixels with a higher intensity value are weighted. m and n are the horizontal and vertical distances in pixel, respectively, that are included in the calculation. That means, around the position of the maximum intensity an area in x-direction $\pm m$ and in y-direction $\pm n$ are used for the evaluation. For this work the values $m = 3$, $n = 10$ and $p = 3$ were used. The laser line computed in this manner is shown in Figure 4.2(b). Note, that a shift away from the maximum intensity is possible, if the intensity values around the maximum are asymmetrical. This method enables non-integer values for the points of the laser line, thus preventing steps due to the inaccuracy of the pixel-level. There also exists the possibility of outliers (i.e. wrong reflections that are not part of the actual laser line), which it is necessary to detect and remove. Only long and continuous parts can be regarded as real parts of the laser line. Thus, a segmentation of the laser line is performed: A new segment starts when the vertical difference of two neighbouring laser points is higher than a value d_v; the segment is discarded if it is shorter than a value l_{min} (i.e. only segments longer than l_{min} are processed as part of the actual laser line) and all its points x- and y-values are set

to "NaN"[2]; Finally, all laser points, whose column's maximum intensity value is below a threshold τ_i, are discarded and set to "NaN" as well (this is done, because there might exist columns that do not contain a part of the laser line e.g. due to occlusion). The values used were: $d_v = 5$ [pixel], $l_{min} = 15$ [pixel] and $\tau_i = 0.3$ (where 1.0 represents white and 0.0 represents black). These values depend on the problem that is dealt with.

The points of the laser line were transformed from the pixel coordinate frame to the real-world coordinate frame according to the second method described in Chapter 2.2. The real-world x- and y-values of the points of the laser line were saved in two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, with a length of the image width (i.e. one value for every column of the image).

4.2 Separation of the laser light

Figure 4.1 shows, that the laser line consists of three segments: The left, as well as the right segment are laser lines reflected from the background; the middle segment is the reflection from the fish. The task now is to segment the data so that the portions corresponding to the fish and the background are available separately.

A histogram of $\boldsymbol{y}$ (i.e. the vertical positions of the points of the laser line) is computed. An example of such an histogram in Figure 4.3 shows: the laser line is clearly bimodal with respect to the histogram of its y-values, $\boldsymbol{y}$. The left part represents the points reflected by the fish's surface, while the peak on the right corresponds to the line on the background. The method proposed in Chapter 3.2 could be used to find a threshold to separate this two parts. However, this determination method underlies the assumption, that the two parts can be modeled by Gaussian distributions, which is not the case here. While this might work for histograms with a large gap between the two parts (i.e. where the fish's surface is far away from the

[2]The notation "NaN" is borrowed from *MATLAB*® and represents "not a number". It stands for missing points, whose values are undefined. In this thesis, this notation will be used to denote missing points.

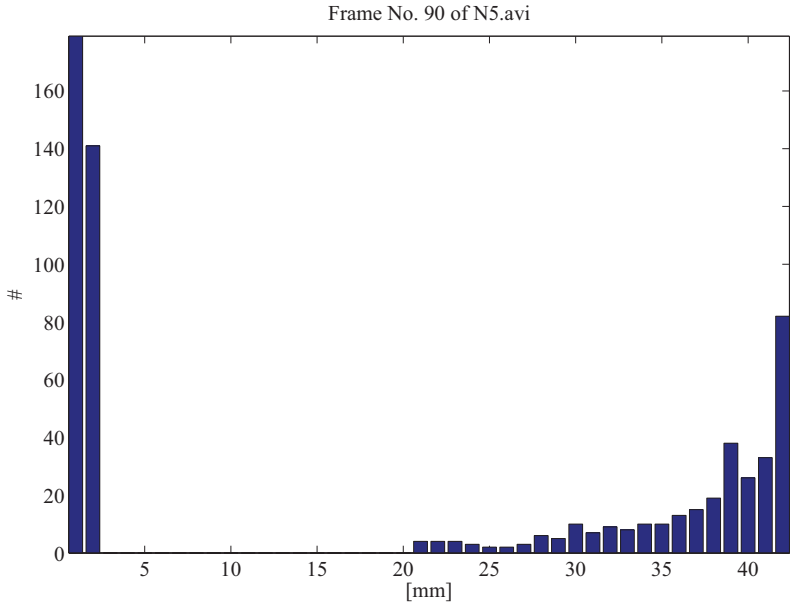


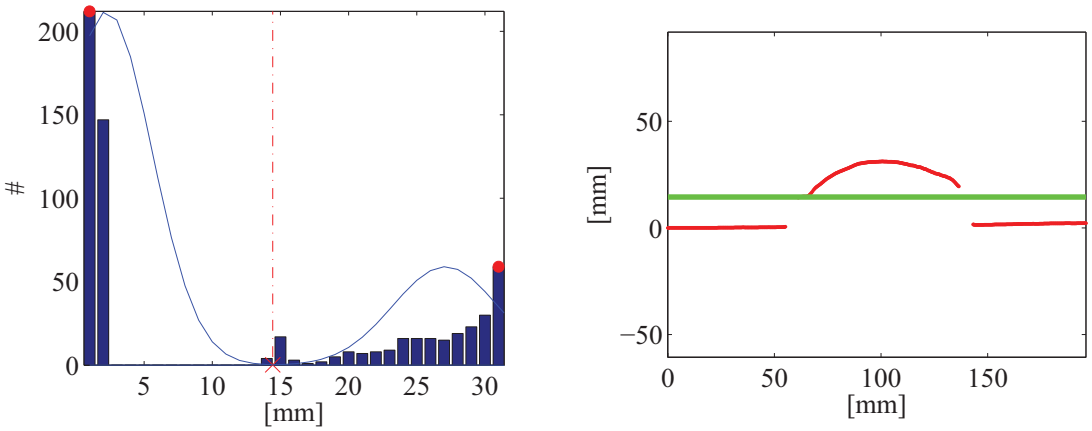
Figure 4.3: Histogram of the values of $\boldsymbol{y}$. In this highly asymmetric bimodal histogram, the left part represents the laser line reflected from the fish and the right part represents the background.

background), there arise problems, when the gap closes. This can be seen in Figure 4.4. It is visible that the highly asymmetric histogram is poorly modeled with Gaussian distributions. Thus, the determined threshold (represented by the dash-dotted line in Figure 4.4(a)) identifies parts of the laser line on the fish as part of the background. Hence another method to determine a threshold is needed, that takes the asymmetry of the histogram into account.

An alternative algorithm which uses normal distances between a line and points was used: Given the line $\boldsymbol{l}$ in homogeneous coordinates $\boldsymbol{l} = [l_1, l_2, l_3]$, which connects the two dominant peaks in the histogram, and the points $\boldsymbol{h}$ of the histogram, where $\boldsymbol{h_i} = [b_i, c_i, 1]^{\mathrm{T}}$, the vector of normal distances $\boldsymbol{n}$ is computed as follows,

$$\boldsymbol{n_i} = \boldsymbol{l}\,\boldsymbol{h_i}\,. \tag{4.3}$$

b_i corresponds to the bins of the histogram and c_i is the count of the respective bin. The central bin value b_i for the count c_i with the maximum normal distance to the



(a) Thresholding using two Gaussian distribution.

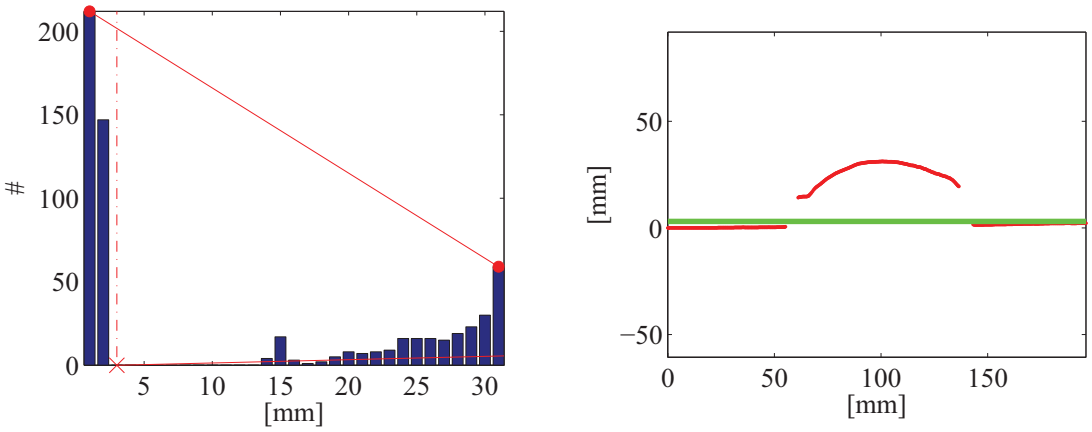
(b) Laser line and threshold.

Figure 4.4: Threshold identification using Gaussian distributions of Frame No. 168 of N1.avi. (a): The asymmetric behaviour of the histogram causes an incorrect modeling through Gaussian distributions, thus the intersection of the two curves creates a wrong threshold (dash-dotted line). (b): the wrongly determined threshold (green horizontal line) identifies parts of the fish as background.

line $\boldsymbol{l}$ is used as the threshold value τ_s, that separates the fish from the background. An example is shown in Figure 4.5. This method leads to a better threshold since it lies closer to the laser line reflected by the background. Therefore, it distinguishes better between background and fish even when the fish's surface is close to the background. This method has been adopted for use in this application.

The values $y(i)$ of $\boldsymbol{y}$ are assigned to a vector $\boldsymbol{y_f}$ (i.e. the fish) or to a vector $\boldsymbol{y_b}$ (i.e. the background) with the same length as $\boldsymbol{y}$, depending on their value and the threshold τ_s,

$$y(i) \rightarrow \begin{cases} y_f(i) & \text{if } y(i) > \tau_s \\ y_b(i) & \text{if } y(i) \leq \tau_s \end{cases} . \tag{4.4}$$



(a) Thresholding using the maximum normal distance.

(b) Laser line and threshold.

Figure 4.5: Threshold identification using the maximum normal distance of Frame No. 168 of N1.avi. (a): The histogram's point with the highest normal distance between the line through the two peaks and the histogram's counts is used as threshold (dash-dotted line). Note: the two continuous lines do not appear to form a right angle, due to an odd aspect ratio. This was done to improve the visibility of the histogram. (b): The threshold (green horizontal line) splits the laser line into background and fish in a better manner.

4.3 Patching holes and concatenating

For further processing, a continuous surface of the fish without missing data points (i.e. "NaN") between correct values was needed. Therefore, holes in $\boldsymbol{x}$ and $\boldsymbol{y}_f$ were patched. For $\boldsymbol{y}_f$ that means, positions j were identified, where the values of $\boldsymbol{y}_f(j)$ are "NaN". If there exist positions i and k with correct values $\boldsymbol{y}_f(i)$ and $\boldsymbol{y}_f(k)$, such that $i < j < k$, linear interpolation between the values of $\boldsymbol{y}_f(i)$ and $\boldsymbol{y}_f(k)$ is performed:

$$y_f(j) = y_f(i) + (j - i)\,\frac{y_f(k) - y_f(i)}{k - i}\,. \tag{4.5}$$

Holes in $\boldsymbol{x}$ were patched analogically.

The vectors $\boldsymbol{x}$, $\boldsymbol{y}_f$ and $\boldsymbol{y}_b$ are concatenated with the vectors from previous slices[3],

[3] A slice represents the laser line found in a frame of the video, since the laser plane crossing the objects can be seen as the surface of one slice of the object.

thus forming the matrices Y, Z_f and Z_b with a size of $i \times s$, where i is the width of the images from the video and s is the number of slices (i.e. number of frames of the video). The matrix X with the dimensions $i \times s$ is constructed as:

$$X = \begin{bmatrix} 1\,w & 2\,w & \dots & (s-1)\,w & s\,w \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1\,w & 2\,w & \dots & (s-1)\,w & s\,w \end{bmatrix}, \tag{4.6}$$

where w converts the frames to millimeter by $w = v/f$. f is the frames per seconds of the video and v is the speed the fish moves across the laser plane in millimeter per second.

The three matrices X, Y and Z_f were used to generate the complete surface and to calculate the dimensions of the fish and Z_b was used as a reference to calculate the complete surface.

4.4 Computation of the complete surface

The light sectioning method with one laser delivers the upper part of the surface only. Even though this model enables a measurement of the fish's length or height, it does not deliver sufficient information to calculate its volume. Since the bottom part of the fish is not touching the background at every point, computing the difference between Z_f and the plane described by the points in Z_b gives an incorrect volume. Thus, a method to reconstruct the missing surface is needed.

However, the appearance of the fish's rear side is unknown. Generally, the assumption can be made, that a fish lying on a flat surface is nearly symmetrical with respect to the horizontal axis. This can be used to approximate the bottom surface: Let $z_{b,i} = Z_b(:,i)$[4] be the ith column of Z_b and $z_{f,i} = Z_f(:,i)$ be the ith column of Z_f,

[4]The *MATLAB*® notation M(:,i) represents a vector consisting of the ith column of the matrix M. For clarity, this notation is used throughout this thesis.

the axis of symmetry, a_i, for every column i of Z_f (i.e. a slice of the fish's surface) is calculated by

$$a_i = \frac{\max(z_{f,i}) - \bar{z}_{b,i}}{2} + \bar{z}_{b,i} \,, \tag{4.7}$$

where $\max(z_{f,i})$ is the highest point of the fish's surface in column i, and $\bar{z}_{b,i}$ is the mean of the background in column i (it can be assumed, that all values in $z_{b,i}$ are the same; however, there might be irregularities, thus the mean over all values of a column is used). Be Z_f^* the matrix containing the values for the bottom surface, the bottom surface of the fish in column i, $z_{f,i}^*$, is created by mirroring $z_{f,i}$ around the axis of symmetry of the corresponding column i:

$$z_{f,i}^* = -(z_{f,i} - a_i) + a_i = -z_{f,i} + 2\,a_i \,. \tag{4.8}$$

If there exist values in $z_{f,i}$ located below the corresponding a_i, this values would be mirrored to the wrong direction (i.e. above a_i). In this case a_i was replaced by the minimum value of $z_{f,i}$. Mirroring around this new axis would lead to points below the background, thus all points in $z_{f,i}^*$ where compressed to the range of $[a_i, \bar{z}_b(i)]$.

When a fish lies on a flat surface, gravity deforms the symmetric shape towards the surface, consequently the fish's upper side has a stronger curvature than the lower one. Thus, if the fish touches the background, the lowest point of the fish's surface should be lower than or equal to the axis calculated in Equation 4.7. If, and only if, the fish does not touch the background (i.e. this slice of the fish is in the air), the lowest point should be above this axis. Practically, there can be missing points due to occlusion or the objects form, especially at the bounds of the fish's laser line missing points were observed. This happens due to the characteristics of the fish's surface: at the edges the facing is nearly vertical. This leads either to the laser light not reaching the object or to the camera not receiving enough reflected light. A gap between the laser line on the fish and the line on the background is the effect. Thus, this algorithm can not discriminate between slices, that really are in the air and the ones with missing points. However, an algorithm that enables this distinction is necessary.

First ideas were the use of an ellipse or an egg curve (e.g. Granville's egg) as ap-

proximations of the fish's cross section. However, ellipse-fitting with a horizontal major axis performed poorly and an additional constraint, that the ellipse for every slice i is not allowed to cross $\bar{z}_{b,i}$, would have led to computationally too expensive algorithms. The same held for egg curves, which are octic in nature.

Another approach was to fill the aforementioned gap between the background's and the fish's laser line using the informations of the data in Y and Z_f. Two methods were tested: extrapolate the points in Z_f for every column using the Taylor series; extrapolate the points in Z_f by calculating the osculating circles of the first and last correct (i.e. not "NaN") value of every column.

The first three terms of the Taylor series [23],

$$f(x) = f(a) + f'(a)\,(x-a) + \frac{f''(a)}{2!}\,(x-a)^2\,, \tag{4.9}$$

were used to expand each column $\boldsymbol{z}_{f,i} = \mathsf{Z}_f(:,i)$ at both ends of the correct values. The first and the last correct values of $\boldsymbol{z}_{f,i}$ were used as $f(a)$, respectively. Following Chapter 3.3, the derivatives of the values z of $\boldsymbol{z}_{f,i}$ were computed as $\boldsymbol{z'} = \mathsf{D}\,\boldsymbol{z}$ and $\boldsymbol{z''} = \mathsf{D}\,\boldsymbol{z'}$ with a support length of 31 and a degree of 2 for D. Sometimes the approximation led to an extrapolation, that veered away from the background (see Figure 4.6 for an example). Since the fish's surface at the borders should be monotonically decreasing to form a closed shape, this would be an incorrect representation of the fish. Furthermore, closer examination of the Taylor series showed, that it is an inappropriate method to support additional information whether the fish's cross-section touches the background or not. It can be assumed, that the surface of a fish reaches a point of symmetry, when its tangent is vertical (i.e. the tangent has an infinite slope). If this point lies above the axis of symmetry calculated following Equation 4.7, the cross-section does not touch the ground. The Taylor series can only expand a given function, using a polynomial approximation, calculated from the derivatives at a single point. A function with a vertical tangent in a point p is not differentiable in p though and thus can not be calculated using the Taylor series.

The radius of curvature of every point of the laser line on the fish's surface can be

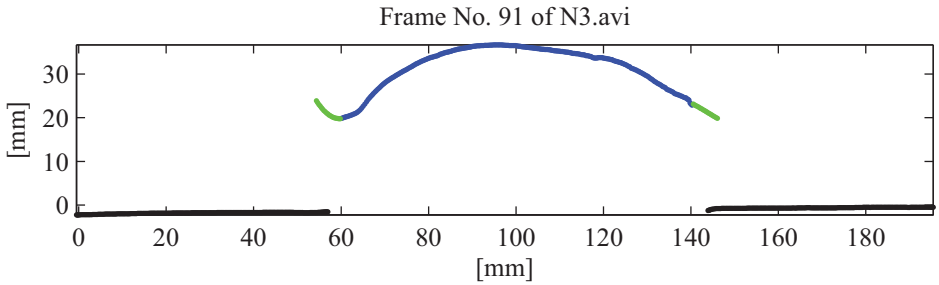


Figure 4.6: Extrapolation using the Taylor series. The new data points (green) on the left side, extrapolated from the fish data points (blue), veer away from the background (black), which should not happen.

calculated using Equations 3.22 and 3.24 from Chapter 3.4. The derivatives of the correct values of every column of Z_f and Y were calculated as $\boldsymbol{y}' = \mathsf{D}\,\boldsymbol{y}$, $\boldsymbol{y}'' = \mathsf{D}\,\boldsymbol{y}'$, $\boldsymbol{z}' = \mathsf{D}\,\boldsymbol{z}$ and $\boldsymbol{z}'' = \mathsf{D}\,\boldsymbol{z}'$ with a support length of 31 and a degree of 2 for D. With the radius and the center point from Equation 3.25, the osculating circles for the first and the last correct value, respectively, of $\boldsymbol{z}_{f,i}$ for all columns i can be calculated. Since a circle has two tangents with infinite slope, using it as an expansion of the laser line adds additional information, whether the fish touches the background or not. The procedure to complete the fish's surface of every column i (i.e. slice of the fish) was performed as follows:

1. Do not use the osculating circle, if it has a positive curvature (i.e. the circle's center lies above the fish's surface).

2. The initial horizontal axis a_i is calculated as described in Equation 4.7;

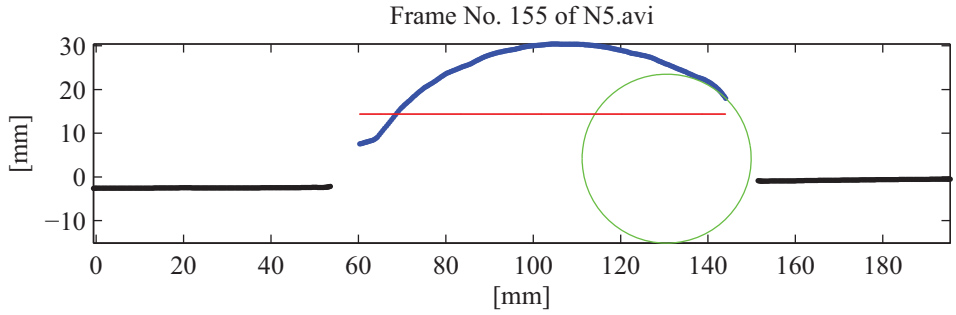
3. If the minimum value of $\boldsymbol{z}_{f,i}$ is below a_i, then this value is assigned as the new a_i. Only those points of the osculating circles are used, that are either above this point a_i or above the point where the circle's tangent is vertical;

4. If all points are above a_i, the laser points are expanded using the osculating circles until either one of them reaches the point, where the circles tangent is horizontal, or both reach the horizontal axis a_i. In the first case, this position is assigned as the new axis a_i and the fish is in the air at this slice. In the second case, the fish touches the background;

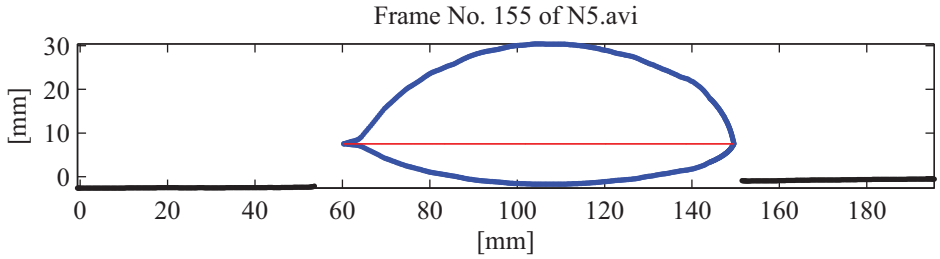
5. The values of the bottom half of the fish's surface, $z_f^*(j,i)$, are generated following Equation 4.8, using a_i calculated as described in the previous steps.

6. If the final value of the horizontal axis a_i lies below the initial one (i.e. the fish touches the background), all values of $z_{f,i}$ are compressed to the range of $[a_i, \bar{z}_{b,i}]$.

Figure 4.7(a) shows an example of a slice, where points of the fish (blue) lie below the initial mirror axis (red) and only one osculating circle (green) can be used.



(a) Fish's surface (blue) with one osculating circle (green) and the initial mirror line (red).



(b) Completed fish's surface (blue) with a compressed bottom half and the final mirror line (red).

Figure 4.7: Completing the fish's surface with osculating circles. (a): There are points of the fish's surface (blue), that lie below the initial mirror line (red). Only one osculating circle is used, since the other one would have been above the fish's surface. (b): The final cross-section of the fish (blue) with the used mirror line in red.

Chapter 5

Results

The data used to test the approach were video files in avi-format for 13 different fish together with their corresponding homographies. The data was provided by Taweepol Suesut[5]. The equipment he used was: a laser diode module RLLH650-16-3 with a wavelength of 650nm and a cylindrical glass lens; a Logitech Webcam Pro 9000 with a resolution of 640×480 and an optical interference filter matched to the wavelength of the laser; a conveyor with a 24VDC motor and a 16-bit micro controller by DS-PIC. The reference values for width, height and length of the fish were measured with a vernier, the weight was measured by a water replacement method.

The approaches described in Chapter 4 were tested on all 13 videos. An example of a final surface reconstruction based on this method can be seen in figure 5.1. In the middle and in the front section of this fish parts are visible, where the algorithm discarded the extrapolating osculating circle due to the restrains described in the previous chapter. Table 5.2 shows the results of the fish's height of the proposed method compared to the results of a manual measurement with a vernier. The maximal measured error was 3.69mm, whereas the minimal error was 0.16mm. The mean error was 1.55mm (or 5.44%) with a standard deviation of 1.28mm (or

[5]Department of Instrumentation Engineering, Faculty of Engineering, King Mongkut's Institute of Technology, Ladkrabang, Bangkok, 10520 Thailand

Fish No.	Height (Vernier) [mm]	Height (Measured) [mm]	Error [mm]	Error [%]
N1	41.72	40.99	0.73	1.75
N2	44.58	44.20	0.38	0.85
N3	40.88	40.72	0.16	0.38
N5	45.98	44.71	1.27	2.76
N6	50.89	51.10	0.21	0.40
N6T	28.94	27.31	1.63	5.63
N7	26.93	27.11	0.18	0.67
N8	32.72	29.49	3.23	9.86
N9	22.39	19.02	3.37	15.04
N10	32.69	31.57	1.12	3.41
N11	28.17	26.58	1.59	5.65
N12	27.61	23.92	3.69	13.38
N13	23.67	21.09	2.58	10.90

Table 5.1: Measurement and error of height.

5.20%). It can be seen, that the measurement for larger fish (i.e. N1-N6) is more accurate than for the smaller specimen (e.g. N12).

The difference between the conventionally measured width and the width calculated from the described method can be seen in Table 5.2. The maximal measured error was 31.20mm, the minimal error 0.08mm. The mean error was 6.31mm (or 10.88%) with a standard deviation of 8.19mm (or 16.60%). The extreme error of N11 originated from an osculating circle that spreads away from the original surface (see Figure 5.2). Without this misscalculation, the mean error would improve to 4.34mm (6.37%) and the standard deviaton to 3.50mm (3.49%). In the videos of N6 and N10 the laser light was reflected poorly in some border areas, which led to missing data points. As a result, the algorithm could not reconstruct the surface entirely and thus created the error.

The results for the length and volume measurement could not be compared, because the speed of the conveyor could not be determined. No correlation between the measured data and the results could be found, since the examination of the final surfaces suggested, that different speeds were used for different fish. Additionally,

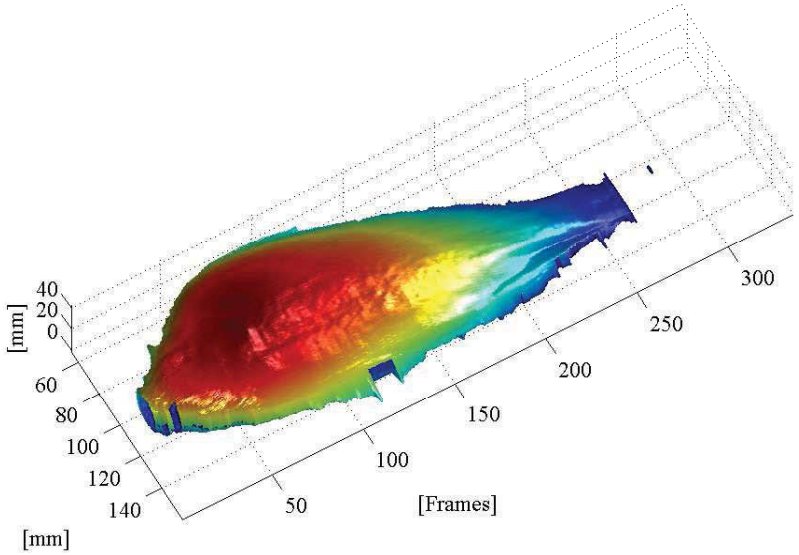


Figure 5.1: Reconstructed surface of N5. Several details are still visible (e.g. the eye). The missing parts in the front and middle section are due to the algorithm discarding the osculating circles. At the tail it is clearly visible, that the line separation algorithm has difficulties discriminating between underground and fish, when both are close together.

the algorithm sometimes has problems with the detection of the tail fin. If the fish's fin is close to the underground (i.e. laying directly on it) the thresholding algorithm can not discriminate between fish and underground any more. Thus, the tail fin is detected as part of the underground. This happened specially with the smaller fish (N6T - N13), the tails of the larger ones mostly did not touch the underground. This makes the measurement of the fish's length highly complicated even when the speed of the conveyor is known. The measurement of the volume would only be influenced marginally, though, since the volume of the tail fin is insignificant.

Fish No.	Width (Vernier) [mm]	Width (Measured) [mm]	Error [mm]	Error [%]
N1	94.17	86.87	7.30	7.75
N2	95.05	92.99	2.06	2.17
N3	95.22	92.43	2.79	2.93
N5	105.50	105.58	0.08	0.07
N6	131.84	118.38	13.46	10.21
N6T	47.81	43.44	4.37	9.15
N7	45.65	47.66	2.01	4.40
N8	51.44	47.53	3.91	7.59
N9	39.56	37.34	2.22	5.62
N10	54.76	48.64	6.12	11.18
N11	48.01	79.21	31.20	64.98
N12	42.82	38.66	4.16	9.72
N13	42.85	40.45	2.40	5.61

Table 5.2: Measurement and error of width.

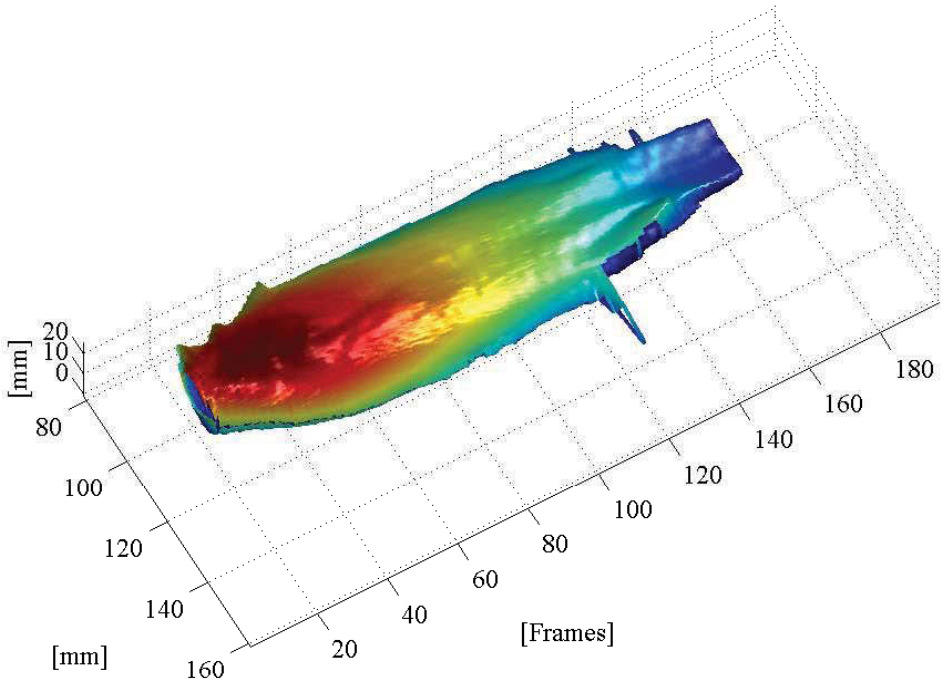


Figure 5.2: Reconstructed surface of N11. At the beginning of the last third incorrectly calculated osculating circles can be seen. This happens, when the surface representation of the fish at the border is too flat, thus creating a large radius of curvature.

Chapter 6

Discussion and Conclusion

An approach for the measurement of different dimensions of fish was presented in this part of the thesis. The proposed methods are based on the principle of light sectioning and all approaches — laser light extraction, laser line separation, completion of the surface and dimension calculation — were tested on real fish. The advantages of this method are:

1. The method can be implemented in-line. No manual work is involved which increases the possible throughput rate enormously;
2. The measurement is realized without contacting the fish, thus the good can not be damaged;
3. There is no wear on the measurement setup due to the lack of moving parts;
4. Depending on the hardware, the measurement can be performed within seconds.

However, during the development and after the result comparison there also emerged drawbacks:

1. A complicated three-dimensional calibration of the system is necessary beforehand. Errors, that occur during this calibration process are not repairable during measurement. An incorrect calibration leads to incorrect data;

2. Ambient light may influence the results of the laser light detection;
3. The bottom part of the fish still remains a factor of uncertainty;
4. the calculation of the osculating circles might lead to incorrect measurements. This can happen, if there are data points missing at the borders of the fish, thus leaving a flat surface at the borders;
5. the speed of the conveyor holding the fish as well as the frame rate of the camera have to be known and steady in order to enable a correct measurement of the length and volume;

In addition, it was noticed, that the laser light is not reflected evenly by the fish's surface.

6.1 Proposed improvements and future work

Some improvements can be made in order to reduce the previously mentioned disadvantages. To minimize the influence of ambient light, a housing over the measurement setup can be installed. The image quality and resolution of 640×480 of the Logitech Webcam Pro 9000 is not adequate for an application in the industrial field. Figure 4.1 shows noisy distortion and compression artifacts. A better camera may also improve the identification of the laser line at the border of the fish, where the majority of the missing data points occur, thus reducing the possibility of incorrect osculating circles.

Future investigations should include, why the fish reflects the laser light unevenly and whether there is a means to reduce this effect. Also a testing of the proposed methods in a real processing line should deliver answers to the question at which speed the measurement can be performed. A final idea to improve this approach is, the use of multiple lasers and cameras. Two of the camera-laser systems shown in Figure 2.1 are placed at an angle left and right of the object, respectively. Surfaces that are nearly vertical to the laser reflect the light worse than horizontal surfaces.

With this new setup it can be enabled, that even the edge regions of the fish reflect the laser light properly. Hence, missing data points would be reduced and the performance of the calculation of the osculating circles improved. Additionally, this can suppress the effect of occlusion.

Part II

Fish Identification using Contour Descriptors

Chapter 7

Problem Statement

Describing and classifying objects is one of the main problems in computer vision [16]. In order to classify objects four main activities are required: Image acquisition, preprocessing of the images, feature extraction, and classification . The method of image acquisition is highly dependent on the purpose of the classification. For the classification of two-dimensional shapes, an image taken with a camera orthogonal to the object is sufficient. A single-coloured background with a high contrast to the object's colour highly improves further computation for this kind of setup. In the preprocessing of the images for shape or contour description usually binary images are created. From the binary images, features can be extracted. The description of the contour can be seen as the description of the shape of the object itself, since the contour is an essential property of an object. Thus, an effective contour descriptor is a key component of object description. However, objects have to be compared regardless of their position, size or orientation (i.e. the computation needs to be independent of similarity transformation). Hence, representations of the objects have to be shift, scale and rotation invariant.

Chapter 8

Mathematical Methods

The mathematical principles described here are the foundation for all further approaches in the following chapters. If not stated differently, the citations hold entirely for the individual subchapters.

8.1 Fourier series and transform

The Fourier series of a periodic function $f(x)$ is a composition of an infinite sum of sines and cosines [23]. It uses the fact, that the sine and cosine functions are orthogonal. The Fourier series of a function $f(x)$ in the interval $[-\pi, \pi]$ is given by

$$f(x) = \frac{1}{2} a_0 + \sum_{n=1}^{\infty} a_n \cos n\,x + \sum_{n=1}^{\infty} b_n \sin n\,x\,, \tag{8.1}$$

where,

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\,\mathrm{d}x\,, \tag{8.2}$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\,\cos(n\,x)\,\mathrm{d}x\,, \tag{8.3}$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\,\sin(n\,x)\,\mathrm{d}x\,, \tag{8.4}$$

are called the Fourier coefficients of f for $n = 1, 2, \ldots$. Extending the Fourier series to complex coefficients for a function $f(x)$ periodic in $[-L/2, L/2]$ yields:

$$f(x) = \sum_{n=-\infty}^{\infty} A_n \, \mathrm{e}^{2\,j\,\pi\,n\,x/L} \,, \tag{8.5}$$

with

$$A_n = \frac{1}{2\,\pi} \int_{-L/2}^{L/2} f(x) \, \mathrm{e}^{2\,j\,\pi\,n\,x/L} \, \mathrm{d}x \,. \tag{8.6}$$

This is the basis for the generalized Fourier transform as the length $L \to \infty$. The discrete A_n is replaced with the continuous $F(k)\,\mathrm{d}k$, while letting $n/L \to k$, and the sum is changed to an integral. The forward Fourier transform is

$$F(k) = \mathcal{F}_x[f(x)](k) = \int_{-\infty}^{\infty} f(x) \, \mathrm{e}^{-2\,\pi\,j\,k\,x} \, \mathrm{d}x \,, \tag{8.7}$$

the inverse Fourier transform is

$$f(x) = \mathcal{F}_k^{-1}[F(k)](x) = \int_{-\infty}^{\infty} F(k) \, \mathrm{e}^{2\,\pi\,j\,k\,x} \, \mathrm{d}k \,. \tag{8.8}$$

The Fourier transform relates a function $f(t)$ (also called signal) with another function $F(k)$. While the original signal is time dependent, it is also called the time domain representation of the signal. Whereas $F(k)$ is called the frequency domain of the signal, since it shows the frequency spectrum of the signal.

Writing this for a discrete function f_k, with $k = 0, \ldots, N-1$, gives the discrete Fourier transform,

$$F_n = \mathcal{F}_k[\{f_k\}_{k=0}^{N-1}](n) = \sum_{k=0}^{N-1} f_k \, \mathrm{e}^{-2\,\pi\,j\,n\,k/N} \,, \tag{8.9}$$

and its inverse transform,

$$f_k = \mathcal{F}_n^{-1}[\{F_n\}_{n=0}^{N-1}](k) = \frac{1}{N} \sum_{n=0}^{N-1} F_n \, \mathrm{e}^{2\,\pi\,j\,k\,n/N} \,. \tag{8.10}$$

Discrete Fourier transforms reveal periodicities in input data as well as the relative

strength of the different periodic components, thus being extremely useful for the analysis of the input data.

8.2 Elliptic Fourier descriptors and shape signatures

The coefficients of the Fourier transform form the Fourier descriptors of the shape of a function in the frequency domain. The basic idea behind elliptic Fourier descriptors is, that a closed curve (e.g. the contour of an object) can be seen as a periodic function with the values $[x(t), y(t)]$ with a continuous parameter $t = 0, ..., N - 1$. To apply the Fourier transform to a closed curve as a function of two variables, it has to be transformed into a one-dimensional function. This representation is called shape signature. There are two main shape signatures [22]: complex coordinates and centroid distance.

The complex coordinates function $z(t)$ is the complex number generated from the curves points as

$$z(t) = [x(t) - \bar{x}] + j\ [y(t) - \bar{y}]\ , \tag{8.11}$$

where [$\bar{x}$,$\bar{y}$] is the centroid of the curve, calculated by the coordinates mean. Due to the shift to the centroid, $z(t)$ is translation invariant. A rotation of the object would cause circular shift and scaling only changes $z(t)$ linearly.

The centroid distance function $r(t)$ is expressed by the distance of the curves point from the curve's centroid:

$$r(t) = \sqrt{[x(t) - \bar{x}]^2 + j\ [y(t) - \bar{y}]^2}\ . \tag{8.12}$$

Again, $r(t)$ is translation invariant due to the relation to the centroid. Rotation also causes circular shift and scaling introduces only linear change in $r(t)$.

8.3 Vector norm and Euclidean metric

For a vector, $\boldsymbol{x} = (x_1\, x_2\, \cdots\, x_n)^{\mathrm{T}}$, the p-norm is defined as

$$\|\boldsymbol{x}\|_p \triangleq \left(\sum_i |x_i|^p\right)^{1/p}, \tag{8.13}$$

where $|x_i|$ is the absolute value of x_i, if x_i is a real number, or the complex modulus, $|z|$,

$$|z| = |x + j\,y| \triangleq \sqrt{x^2 + y^2}\,, \tag{8.14}$$

if x_i is a complex number [23].

The Euclidean metric, or Euclidean distance, of two vectors, $\boldsymbol{x} = (x_1\, x_2\, \cdots\, x_n)^{\mathrm{T}}$ and $\boldsymbol{y} = (y_1\, y_2\, \cdots\, y_n)^{\mathrm{T}}$, in Euclidean n-space is given by

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|x - y\|_2 = \sqrt{(x_1 - y_1)^2 + ... + (x_1 - y_1)^n}\,, \tag{8.15}$$

and gives the distance between these two vectors.

8.4 Convolution

A convolution is the integral of the product of one function g shifted over another function f and is written as $f * g$ [23]. It is defined as:

$$[f * g]\,(t) \triangleq \int_{-\infty}^{\infty} f(\tau)\, g(t - \tau)\, \mathrm{d}\tau = \int_{-\infty}^{\infty} g(\tau)\, f(t - \tau)\, \mathrm{d}\tau\,. \tag{8.16}$$

Before shifting and integrating one of the two functions has to be mirrored (e.g. $g(\tau) \rightarrow g(-\tau)$). A Convolution expresses the amount of overlap of the two func-

tions. If one function g_T is periodic with the period T, the convolution with another function f is also periodic and can be defined as an integral over a finite interval

$$[f * g_T](t) \triangleq \int_{t_0}^{t_0+T} f_T(\tau)\, g_T(t-\tau)\, \mathrm{d}\tau\,, \tag{8.17}$$

where t_0 is an arbitrary start point and f_T is a periodic summation of f. In the case when g_T is a periodic summation of a function g as well, $f * g_T$ is called a circular or cyclic convolution of f and g.

A convolution of two discrete sequences $f(n)$ and $g(n)$ for nonnegative integer n is called the Cauchy Product [23] and is defined by:

$$(f \circ g)(n) \triangleq \sum_{k=0}^{n} f(k)\, g(n-k) = \sum_{k=0}^{n} g(k)\, f(n-k)\,. \tag{8.18}$$

The circular discrete convolution for a period N is build corresponding to Equation 8.17.

The convolution theorem states, that a convolution can be calculated by multiplying the Fourier spectra of the two functions and taking its inverse Fourier transform:

$$f * g = \mathcal{F}^{-1}(\mathcal{F}[f]\ \mathcal{F}[g])\,. \tag{8.19}$$

8.5 Spectral filtering with local cyclic polynomial moments

Processing spectral data with cyclic polynomial moments can be done similar to normal local polynomial filtering as described in Chapter 3.3. But here $\mathsf{P} = \mathsf{D}\,\mathsf{D}^{\mathsf{T}}$ from Equation 3.19 forms a circulant matrix, since the start and end point are adjacent for cyclic data (e.g. a contour). In circulant matrices the vector $\boldsymbol{p}$ in every

row is a phase shifted copy of the previous row. Hence, $\boldsymbol{y_F} = \mathsf{P}\,\boldsymbol{y}$ can be computed using the Fast Fourier Transform (FFT)

$$\boldsymbol{y_F} = FFT^{-1}\left\{FFT\left\{\boldsymbol{p}\right\}\ FFT\left\{\boldsymbol{y}\right\}\right\}\ , \tag{8.20}$$

although the bases are polynomial [18]. One advantage of the local polynomials over elliptical Fourier descriptors is, that filtering can be performed even for partially occluded contours, whilst elliptical Fourier descriptors need a closed contour.

When computing local cyclic polynomial Savitzky-Goolay smoothing on a set of n data points, a support length l_s (i.e. the number of data points being used to smooth one single point) and a degree d is needed. To get the projection matrix $\mathsf{P_C}$ for all points, a unitary basis B for l_s and d has to be computed, which gives the local projection matrix by $\mathsf{P} = \mathsf{B}\,\mathsf{B}^{\mathsf{T}}$ (this is an $l_s \times l_s$ matrix). The center row of P represents the projection at the center of the support $\boldsymbol{p_{x=0}}$. The core region of $\mathsf{P_C}$ (i.e. all rows further than $(l_s - 1)/2$ from the start or end of the matrix) is filled diagonally with $\boldsymbol{p_{x=0}}$. For the top and bottom of $\mathsf{P_C}$, $\boldsymbol{p_{x=0}}$ is wrapped around to use the needed data points from the end and start, respectively, thus creating a circulant matrix.

To compute cyclic spectra for polynomials of degree d, a polynomial basis S for n points can be generated as follows: Generate a unitary basis B with degree d and a support length of l_s. take the last column of B as $\boldsymbol{p_{x=0}}$ and create a circulant matrix S with the size $n \times n$ as described in the previous paragraph.

Chapter 9

Fish classification

In this chapter, the methods used to identify and differentiate different species of fish are described. Image acquisition and processing of the images as the prerequisites for a classification are presented in the first part. Elliptic Fourier and cyclic polynomial descriptors as the examined contour-based identification methods are described in the second part.

9.1 Image acquisition and preprocessing

Mounting the camera perpendicular to the inspection table ensures that the images are subject only to a similarity transformation, i.e. rotation, translation and scaling. There is no perspective distortion. Radial distortion, associated with the optics of the camera, can be neglected when the objects to be measured are located at or near the center of the image. A white background was chosen to improve the contrast to the fish. Figure 9.1 shows an example of an image taken in this manner.

The bimodal behaviour of the grayscale intensity value histogram of the image (for an example, see Figure 9.2(a)) can be used to identify a threshold via Gaussian distributions[6]. Some pixels may be incorrectly binarized, as can be seen in Figure

[6]See Chapter 3.2, Thresholding, for the mathematical background.

Figure 9.1: Image of a char (Salvelinus) taken from a perpendicular mounted camera. The radial distortion barely effects the fish in the center.

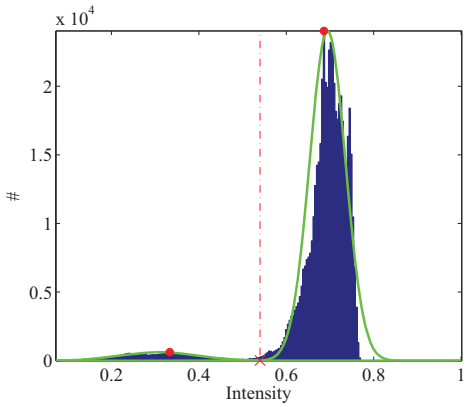
9.2(b). To eliminate this disturbance, morphological operations were applied to the binary image: The holes within the segments[7] in the binary image were filled (i.e. black pixels, completely surrounded by white pixels, are set to white); the segments were labeled, the largest segment was assumed to be the fish and all other segments were eliminated (i.e. set to black).

The contour of the largest segment was determined. In order to make different contours comparable, all contours were resampled to 1000 points.

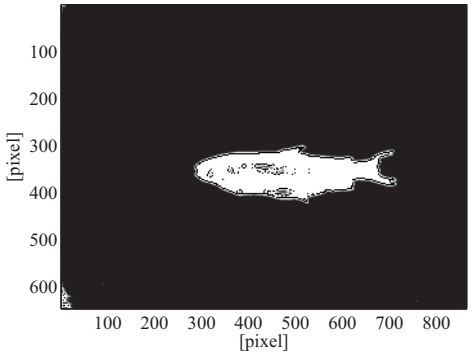
9.2 Feature extraction and classification

Elliptical Fourier - and cyclic polynomial descriptors were implemented and tested as a means of extracting features from the contours.

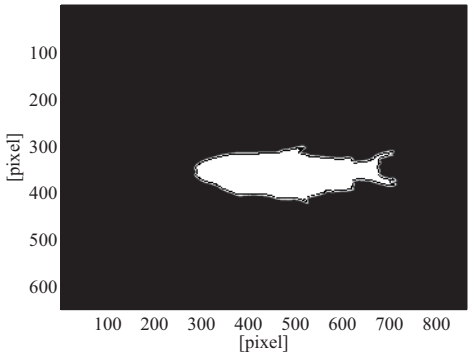
[7] in the context of binary morphology, the term "segment" refers to linked white areas within the binary image.



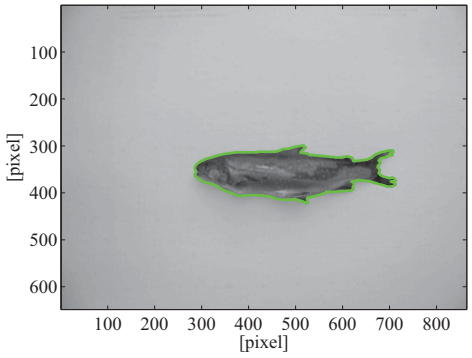
(a) Histogram of the grayscale intensity values of Figure 9.1.



(b) Binary image after thresholding and inverting.



(c) Binary image after morphological processing.



(d) Contour of the fish.

Figure 9.2: Identifying the fish's contour: (a): A threshold (red dash-dotted line) is identified through the intersection point of two Gaussian distributions (green) modeling the histogram. (b): Binary image generated using the found threshold. Incorrect pixels inside the fish and at the bottom left corner are still visible. (c): The morphologically processed binary image only consists of one object without holes, which represents the shape of the fish. (d): The initial image with the final fish's contour (green).

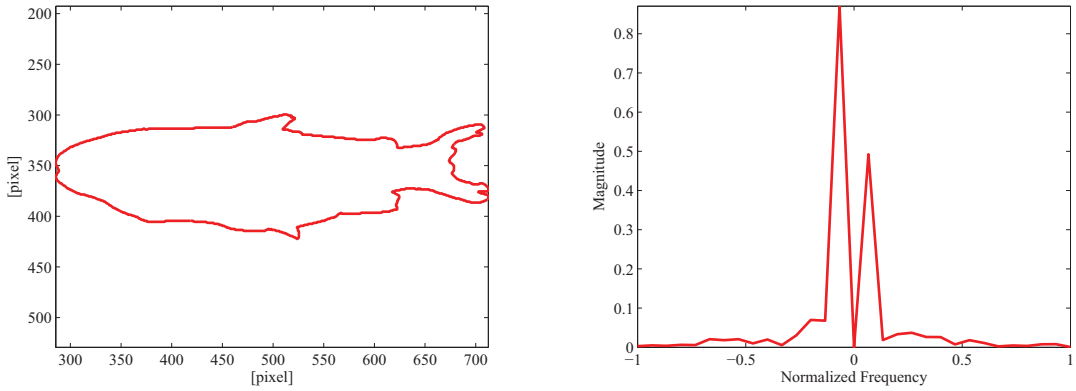
Elliptic Fourier descriptor

The Fourier spectral coefficients, i.e. the Fourier descriptors, $\boldsymbol{a}$, are computed via an FFT. The x and y coordinates of the points on the contour are regarded as complex,

i.e. $p_i = x_i + j\,y_i$, for the Fourier transform. Shift invariance is achieved by eliminating a_0, the first Fourier coefficient. Rotational invariance is achieved by utilizing the shift invariant property of the magnitude of a Fourier spectrum. Consequently, the magnitude of the Fourier coefficients are used as the descriptors. Furthermore, scale invariance is obtained by scaling the coefficients, such that $\|\boldsymbol{x}\|_2 = 1$, where $\boldsymbol{x} = (a_1, a_n)$. Thus, the Fourier descriptors are rescaled by

$$\boldsymbol{b} = \frac{\boldsymbol{a}}{\sqrt{a_1^2 + a_n^2}}\,. \tag{9.1}$$

The lower frequency descriptors describe the general features of the shape, while higher frequency descriptors describe small details. A subset of $\boldsymbol{b}$ consisting of the most significant descriptors is sufficient to compare different Fourier spectra. Figure 9.3 illustrates the first 30 significant descriptors of the contour in Figure 9.3(a) after the descriptors were made shift, scale and rotation invariant.



(a) Contour of a fish.

(b) Detail of the corresponding Fourier spectrum.

Figure 9.3: Contour and its corresponding Fourier spectrum. (a): The contour of a fish generated as described in Chapter 9.1. (b): The first 30 significant descriptors of the Fourier spectrum after it was made shift, scale and rotation invariant.

The Euclidean distance between the first 500 descriptors of the magnitudes, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, of two shapes is used to calculate a similarity factor. The smaller this factor, the higher the similarity.

Cyclic polynomial descriptor

The spectra for polynomials of degree 2 and 3 are shift invariant, but not rotation invariant. However, the radius of curvature[8] is both shift and rotation invariant and can be computed via local polynomial approximations.

It is a prerequisite that two curvature descriptors have the same starting point, if their similarity is to be evaluated. Since this can not be guaranteed, when creating an object's contour, the two curvature descriptors have to be registered to each other. This can be done by computing the circular discrete convolution of the two curvature descriptors, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ of length N,

$$[c_1 \circ c_2]\,(n) \triangleq \sum_{k=0}^{N} c_1(k)\, c_2(N-k)\,, \tag{9.2}$$

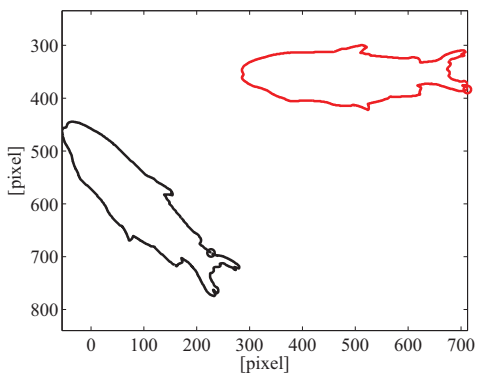
identifying the shift m, where $[c_1 \circ c_2]\,(n)$ has its maximum and performing a rotational shift by m samples on $\boldsymbol{c_2}$:

$$\boldsymbol{c_2} = [c_2(N-m+1)\,, ..., c_2(N),\, c_2(1)\,, ...,\, c_2(N-m)]\,. \tag{9.3}$$

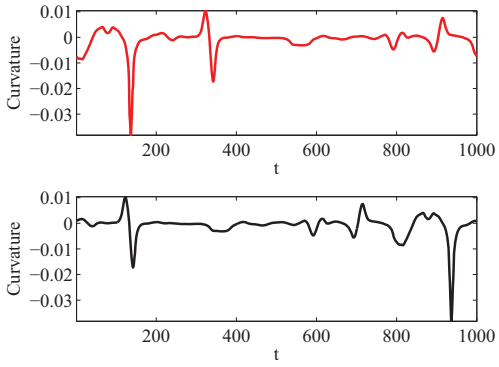
An example, where the curvature descriptor of a fish's contour is registered to a shifted and rotated version with a different starting point, can be seen in Figure 9.4.

The Euclidean distance between the contour descriptors, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, of two shapes is used to calculate a similarity factor. The smaller this factor, the higher the similarity.

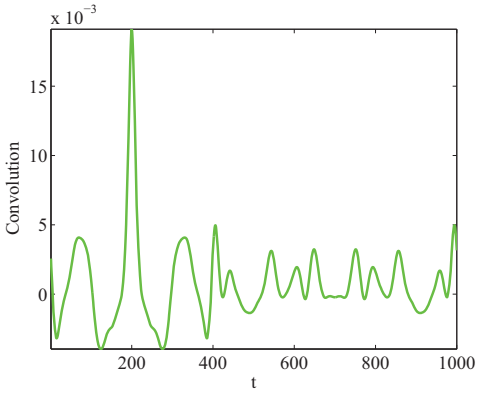
[8] For the method of calculation, see Chapter 3.4.



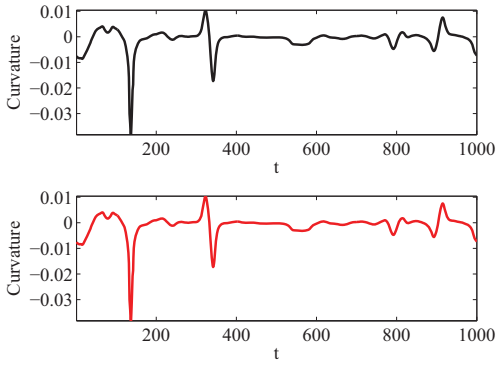
(a) Two shifted and rotated contours of the same fish.



(b) Curvature descriptors of the two contours.



(c) Convolution of the two curvature descriptors.



(d) Registered curvature descriptors.

Figure 9.4: Registration of two curvature descriptors: (a): The contour of a fish (red) and a shifted and rotated version with a different starting point (black). The respective starting points are marked as circles. (b): The curvature descriptors of the contour (top) and the transformed version (bottom). It is clearly visible, that the descriptors are shift, and rotation invariant. (c): The convolution of the two curvature descriptors. The peak at t=200 represents the shift of the starting point. (d): The curvature descriptors after the registration.

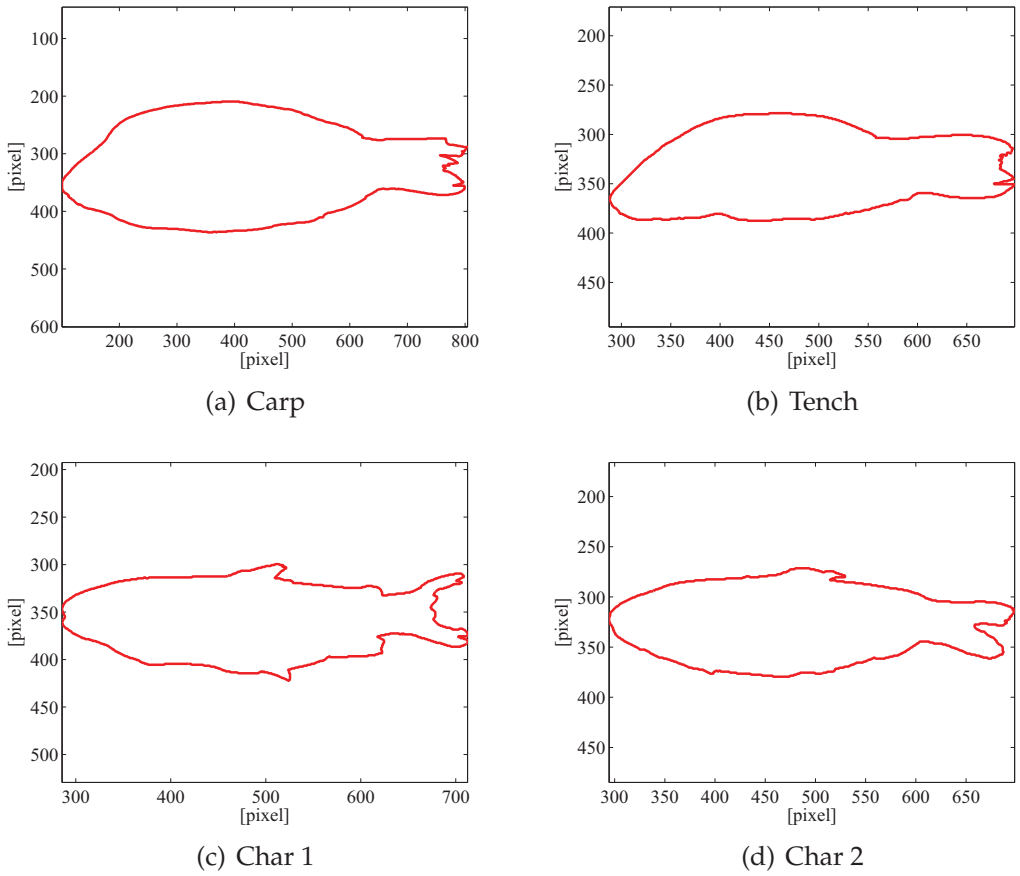
Chapter 10

Results

The fish used to test the two methods were a carp (Cyprinus carpio), a tench (Tinca tinca) and two chars (Salvelinus). Two images of each fish — one where the fish is facing left and one where it is facing right — were taken with a Canon PowerShot SX110 IS with a resolution of 9 Megapixels. The images can be seen in Appendix A. The contours of the four fish facing left can be seen in Figure 10.1.

Both methods — elliptic Fourier descriptors (EFD) and cyclic polynomial descriptors (CPD) — were applied to the overall eight images. The Euclidean distances between the descriptors of the individual fish contours were calculated, the results for the first 500 Fourier descriptors can be seen in Tables 10.1 and 10.2. It is evident, that "Char 1" has a smaller Euclidean distance to the other char, than to the carp and the trench. However, the Euclidean distance between "Char 2" and "Char 1" is not (or only marginal) better than the Euclidean distance of "Char 2" to the other two fish. This can be explained when considering the contours in Figure 10.1: "Char 1" has a very distinct contour, whereas the contour of "Char 2" is quite similar to the contours of the carp and also the tench.

The results for the CPD can be seen in Tables 10.3 and 10.4. The curvature is very prone to noise that may influence the contours shape. Since some curvatures were not registered correctly when a small support length was used to generate B_1 and B_2, a support length of 101 was chosen. However, this leads to heavily smoothed contours and curvatures, which lack even basic details such as dorsal fins. This, and



(a) Carp (b) Tench

(c) Char 1 (d) Char 2

Figure 10.1: Contours of the four fish.

Fish facing left	Carp	Char 1	Char 2	Tench
Carp	0	0.0793	0.0644	0.0625
Char 1	0.0793	0	0.0611	0.0833
Char 2	0.0644	0.0611	0	0.0484
Tench	0.0625	0.0833	0.0484	0

Table 10.1: Euclidean distances between the first 500 Fourier descriptors of fish facing left.

the fact, that the two chars have very distinct contours, led to a poor performance of the CPD.

CPD are not suitable for the classification of different sort of fish, because even the

Fish facing right	Carp	Char 1	Char 2	Tench
Carp	0	0.1152	0.0834	0.104
Char 1	0.1152	0	0.0685	0.0793
Char 2	0.0834	0.0685	0	0.0553
Tench	0.104	0.0793	0.0553	0

Table 10.2: Euclidean distances between the first 500 Fourier descriptors of fish facing right.

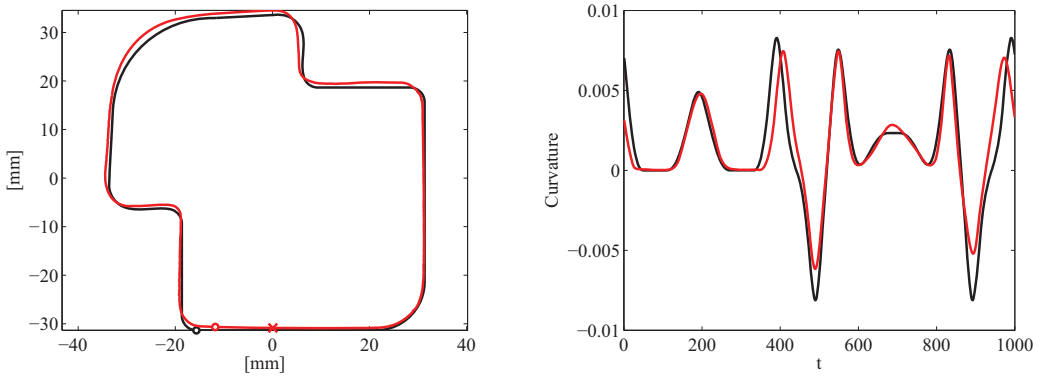
Fish facing left	Carp	Char 1	Char 2	Tench
Carp	0	0.1033	0.0573	0.0253
Char 1	0.1033	0	0.1	0.1152
Char2	0.0573	0.1	0	0.0605
Tench	0.0253	0.1152	0.0605	0

Table 10.3: Euclidean distances between CPD of fish facing left.

Fish facing right	Carp	Char 1	Char 2	Tench
Carp	0	0.1116	0.0806	0.0688
Char 1	0.1116	0	0.0952	0.0913
Char 2	0.0806	0.0952	0	0.0483
Tench	0.0688	0.0913	0.0483	0

Table 10.4: Euclidean distances between CPD of fish facing right.

contours of fish from the same species can vary to a point, where a differentiation is not possible. However, the method used to register two contours to a common starting point can be used for other applications, e.g. registering the measured contour of a profile to the corresponding CAD-data. A test on a rolled profile is shown in Figure 10.2. The gaps between the peaks at $t = 400$ and $t = 100$ in Figure 10.2(b) occur due to rolling errors.



(a) Contours (b) Curvatures

Figure 10.2: Contours and curvatures of a rolled profile and the corresponding CAD-data. (a): The CAD-data (black) and its starting point (circle) compared to the data of the rolled profile (red) with its original (cross) and registered starting point (circle). (b) The registered curvatures of the CAD-data (black) and the data of the rolled profile (red). Note the gaps between the peaks at t=400 and t=1000, which occur due to errors during the rolling process.

Chapter 11

Discussion and Conclusion

A method to extract a fish's contour from an image and two methods to discriminate between different species of fish were presented in this part of the thesis, and tested on real fish. It was shown, that the method to extract the contour from an image works, even when small parts of the background or the fish are identified incorrectly.

The advantage of the elliptic Fourier descriptors (EFD) is, that they are very robust with respect to noisy data, since small irregularities only affect higher frequencies. However, they can only be applied to complete contours without missing points. On the other side, curvature based cyclic polynomial descriptors (CPD) are highly prone to noisy data due to the calculation of the curvature, thus smoothing is needed which influences the results. The fact, that they can be applied to incomplete (i.e. occluded) contours, can be an advantage for different applications.

On this set of contours the EFD performed better than the CPD. The example of the two chars showed that even the shape and appearance of fish from the same species can vary widely, even though only a small amount of specimen was available. Thus, the Euclidean distance between the contour descriptors of two fish is not sufficient for a correct discrimination.

A benefit of this thesis is the fact, that the method used to register two curvatures

to a common start point can be used e.g. to compare the measured data of a rolled profile to the corresponding CAD-data.

11.1 Improvements and alternatives

Another approach to discriminate fish species would be the use of a neural network or machine learning instead of the Euclidean distance. However, that would require a much bigger amount of specimen. Also, the use of other elliptic Fourier descriptors, as described by Zhang and Lu [22], may deliver better results. A way to avoid calculations with the noise-prone curvature could be the utilization of integral invariants [12, 14], which are way more robust with respect to noise.

Bibliography

[1] "Automated measurement of species and length of fish by computer vision". In: *Fisheries Research* 80.2-3 (2006). Pp. 203 –210. ISSN: 0165-7836. DOI: 10.1016/j.fishres.2006.04.009.

[2] Hans-Jochen Bartsch. *Taschenbuch mathematischer Formeln*. München: Fachbuchverl. Leipzig im Carl Hanser Verl., 2007. ISBN: 3446408959.

[3] Sébastien Cadieux, Francois Lalonde, and Francois Michaud. "Intelligent system for automated fish sorting and counting". In: *In Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*. 2000.

[4] Steven X. Cadrin and Kevin D. Friedman. "The utility of image processing techniques for morphometric analysis and stock identification". In: *Fisheries Research* 43 (1-3 1999). Pp. 129–139.

[5] "Computer vision technology for food quality assurance". In: *Trends in Food Science and Technology* 7.8 (1996). Pp. 245 –256. ISSN: 0924-2244. DOI: 10.1016/0924-2244(96)10028-5.

[6] Ewald Fauster, Peter Schalk, and Mark Tratnig. "Calibration Method for Light Sectioning Measurement Systems". In: *ICOSSIP*. 2002. Pp. 1891–1898.

[7] "Fish species recognition using computer vision and a neural network". In: *Fisheries Research* 51.1 (2001). Pp. 11 –15. ISSN: 0165-7836. DOI: 10.1016/S0165-7836(00)00254-X.

[8] Matthew Harker and Paul O'Leary. "Computation of Homographies". In: *British Machine Vision Conference*. Ed. by W.F. Clocksin, A.F. Fitzgibbon, and P.H.S. Torr. Vol. 1. Oxford, UK 2005. Pp. 310–319.

[9] "High-speed model-based weight sensing of complex objects with application in industrial processes". In: *Measurement* 33.1 (2003). Pp. 23 –33. ISSN: 0263-2241. DOI: 10.1016/S0263-2241(02)00029-5.

[10] "Image acquisition techniques for automatic visual inspection of metallic surfaces". In: *NDT & E International* 36.8 (2003). Pp. 609 –617. ISSN: 0963-8695. DOI: 10.1016/S0963-8695(03)00081-1.

[11] "In situ measurement of fish body length using perspective-based remote stereo-video". In: *Fisheries Research* 82.1-3 (2006). Pp. 327 –331. ISSN: 0165-7836. DOI: 10.1016/j.fishres.2006.08.017.

[12] "Integral invariants for robust geometry processing". In: *Computer Aided Geometric Design* 26.1 (2009). Pp. 37 –60. ISSN: 0167-8396. DOI: 10.1016/j.cagd.2008.01.002.

[13] Norbert Koller. "Fully Automated Repair of Surface Flaws using an Artificial Vision Guided Robotic Grinder". Doctoral thesis. Leoben: Institut für Automation, 2007.

[14] Siddharth Manay et al. "Integral invariants for shape matching". In: *IEEE PAMI* 28 (2006). Pp. 1602–1618.

[15] James L. McClelland, David E. Rumelhart, and G. E. Hinton. "The appeal of parallel distributed processing". In: *Parallel Distributed Processing: Foundations*. Ed. by David E. Rumelhart, James L. McClelland, and the PDP Research Group. Cambridge, MA, USA: MIT Press. Chap. 1, pp. 3–44. ISBN: 0-262-68053-X.

[16] Muharrem Mercimek, Kayhan Gulez, and Tarik V. Mumcu. "Real object recognition using moment invariants". In: *Sadhana* (). Pp. 765–775.

[17] P. O'Leary and M. Harker. "Discrete Polynomial Moments and Savitzky-Golay Smoothing". In: *Submitted to ICPRCV*. Vol. 18-20 Dec, Singapore. 2010.

[18] P. O'Leary, M. Harker, and R. Neumayr. "Savitzky-Golay Smoothing for Multivariate Cyclic Measurement Data". In: *IEEE Instrumentation and Measurement Technology Conference, 2010*. Vol. 3-6 May, Austin, TX, USA, 2009. 2010.

[19] Paul O'Leary and Matther Harker. "An Algebraic Framework for Discrete Basis Functions in Computer Vision". In: *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. 2008. Pp. 150–157. ISBN: 978-0-7695-3476-3.

[20] "Optimal portion control of natural objects with application in automated cannery processing of fish". In: *Journal of Food Engineering* 46.1 (2000). Pp. 31 –41. ISSN: 0260-8774. DOI: 10.1016/S0260-8774(00)00068-6.

[21] N.J.C. Strachan. "Length measurement of fish by computer vision". In: *Computers and Electronics in Agriculture* 8 (2 1993). Pp. 93–104.

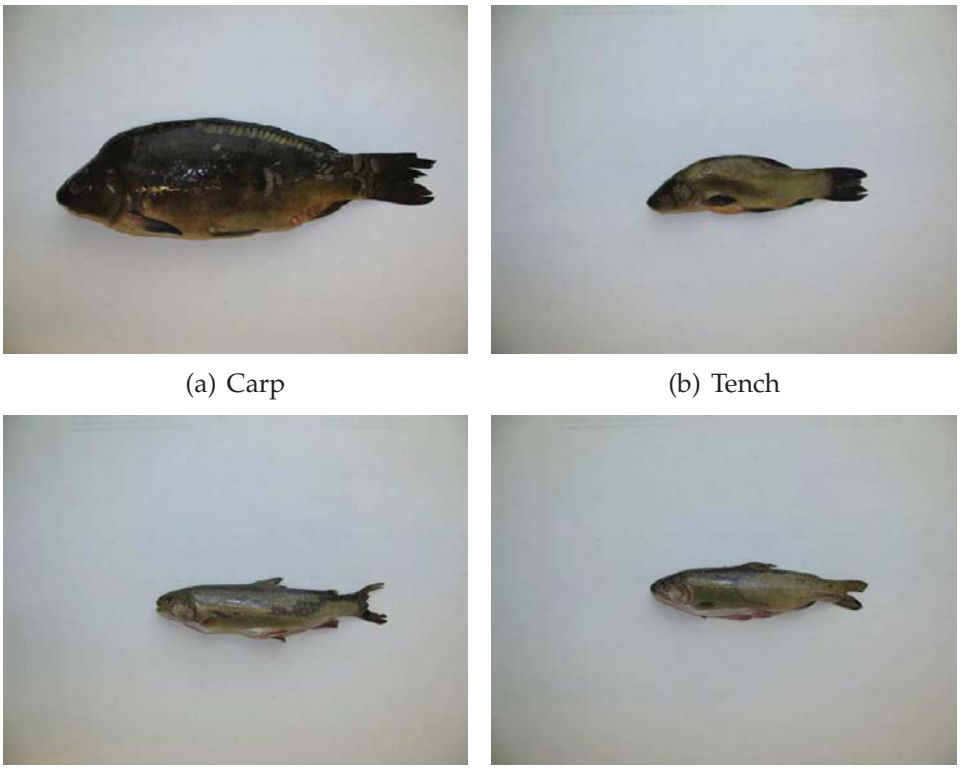
[22] "Study and evaluation of different Fourier methods for image retrieval". In: *Image and Vision Computing* 23.1 (2005). Pp. 33 –49. ISSN: 0262-8856. DOI: DOI:10.1016/j.imavis.2004.09.001.

[23] *Wolfram MathWorld*. 2011. URL: http://mathworld.wolfram.com/ (visited on 01/04/2011).

[24] B. Zion, A. Shklyar, and I. Karplus. "Sorting fish by computer vision". In: *Computers and Electronics in Agriculture* 23 (3 1999). Pp. 175–187.

Appendix A

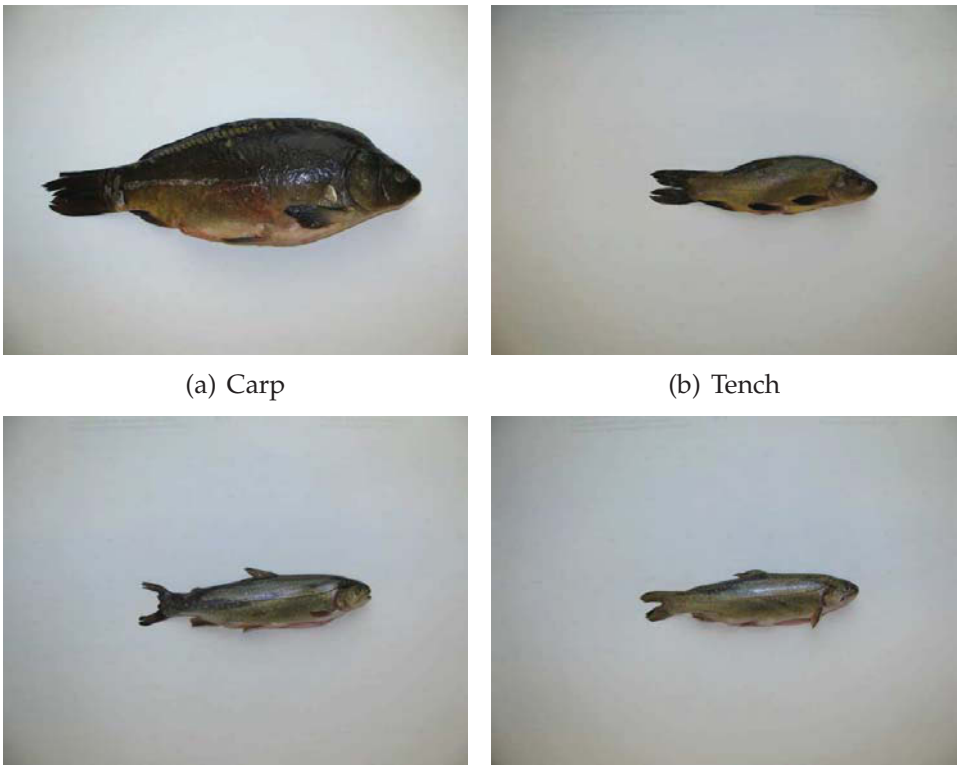
Fish images



(a) Carp (b) Tench

(c) Char 1 (d) Char 2

Figure A.1: Images of fish facing right.



(a) Carp (b) Tench

(c) Char 1 (d) Char 2

Figure A.2: Images of fish facing right.

Appendix B

Matlab Source Codes

B.1 Part I

Listing B.1: Fish3dGUI.m

```
clear all;
close all;

quit =0;
X =[];

while(¬quit)

choice =menu('','Detect Line from AVI','Load Line','Plot Lines','Complete Fish','Plot Fish','Get Measurements','
              Quit');
   switch choice
      case 1
         clear all;
         % Homography.mat with correct homography h needed in saveLine
         saveLine;
         quit =0;
      case 2
         clear all;
         quit =0;
         pathName='C:\Dokumente und Einstellungen\iii\Desktop\Uni\Diplomarbeit\Code\101111\MetricVisionFish\';
         [fileName,pathName] =uigetfile(strcat(pathName,'*.mat'),'Load ');
         if fileName =0
            fig1 =figure('Position',[600,600,170,80],'Toolbar','none',...
               'NumberTitle','off','Name','No file selcted!','MenuBar','none');
            h =uicontrol('Parent',fig1,'Style', 'pushbutton', 'String', 'Ok',...
               'Position', [30 20 110 40], 'Callback', 'close');
         else
            load([pathName,fileName]);
         end
      case 3
         if isempty(X);
            fig1 =figure('Position',[600,600,170,80],'Toolbar','none',...
               'NumberTitle','off','Name','Load a Line first!','MenuBar','none');
            h =uicontrol('Parent',fig1,'Style', 'pushbutton', 'String', 'Ok',...
               'Position', [30 20 110 40], 'Callback', 'close');
```

```
35          else
36             fig1=figure;
37             surf(X,Y,Zfish,'EdgeColor', 'none', 'FaceColor', 'interp', 'Facelighting', 'phong');
38             hold on;
39             surf(X,Y,Zline,'EdgeColor', 'none', 'FaceColor', 'interp', 'Facelighting', 'phong');
40             axis equal; axis ij;
41             xlabel('[Frames]');
42             ylabel('[mm]');
43             zlabel('[mm]');
44             material metal;
45             light;
46             camlight headlight;
47             view(-30,60);
48          end
49       case 4
50          [width, noFrames] =size(Y);
51          Yfish(1:width,1:noFrames) =NaN;
52          ZfishNew(1:width,1:noFrames) =NaN;
53          ZfishStar(1:width,1:noFrames) =NaN;
54
55          degree =2;
56          sl =31;
57
58          for i=1:noFrames
59             [Yfish(:,i), ZfishNew(:,i), ZfishStar(:,i)] =completeFish(Y(:,i),Zfish(:,i),Zline(:,i),degree,sl);
60          end
61       case 5
62          if isempty(ZfishNew) ∨isempty(ZfishStar);
63             fig1 =figure('Position',[600,600,170,80],'Toolbar','none',...
64                'NumberTitle','off','Name','No surface!','MenuBar','none');
65             h =uicontrol('Parent',fig1,'Style', 'pushbutton', 'String', 'Ok',...
66                'Position', [30 20 110 40], 'Callback', 'close');
67          else
68             fig1=figure;
69             surf(X,Yfish,ZfishNew,'EdgeColor', 'none', 'FaceColor', 'interp', 'Facelighting', 'phong');
70             hold on;
71             surf(X,Yfish,ZfishStar,'EdgeColor', 'none', 'FaceColor', 'interp', 'Facelighting', 'phong');
72             axis equal; axis ij;
73             xlabel('[Frames]');
74             ylabel('[mm]');
75             zlabel('[mm]');
76             material metal;
77             light;
78             camlight headlight;
79             view(-30,60);
80          end
81       case 6
82          [w l] =size(X);
83          width =0;
84
85          noNan =find(¬isnan(ZfishNew));
86          length =max(X(noNan))-min(X(noNan))
87          for i=1:l
88             if width < max(Yfish(:,i))-min(Yfish(:,i))
89                width =max(Yfish(:,i))-min(Yfish(:,i));
90             end
91          end
92          width
93          height =max(ZfishNew(noNan))-min(ZfishStar(noNan))
94          volume =sum(sum(ZfishNew(noNan)-ZfishStar(noNan)));
95          num2str(volume)
96       case 7
97          quit =1;
98       otherwise
99          disp('Click a button!');
100   end
```

```
101 end
```

Listing B.2: saveLine.m

```
% To Save the detected lines in corresponding .mat-files

load Homography;

pathName='C:\Dokumente und Einstellungen\iii\Desktop\Uni\Diplomarbeit\3D Measurement\Test\';
[fileName,pathName] =uigetfile(strcat(pathName,'*.avi'),'Load image data');

if(fileName≠0)
   % create blurring filter to smooth the
   % image before finding the laser line
   h =fspecial('average', 3);

   % iterate through frames, find laser line
   nfo =aviinfo(strcat(pathName, fileName));

   noFrames =nfo.NumFrames;

   m =aviread(strcat(pathName, fileName),1);
   % filter out everything but laser color
   Img =m.cdata(:,:,1);
   [height width] =size(Img);

   % filter (i.e. blur) image, and convert from uint8 to double
   ImgFilt =imfilter(im2double(Img),h);

   LaserPoints =findLine2D(ImgFilt);
   [noX noY] =size(LaserPoints);

   X(1:noY,1:noFrames-1) =NaN;
   Y(1:noY,1:noFrames-1) =NaN;
   Z(1:noY,1:noFrames-1) =NaN;
   Zfish(1:noY,1:noFrames-1) =NaN;
   Zline(1:noY,1:noFrames-1) =NaN;

   % progress bar
   h_wait =waitbar(0,'Please wait...',...
         'Position',[250,320,270,50]);

   for i=1:(noFrames-1)

      m =aviread(strcat(pathName, fileName),i);

      % filter out everything but laser color
      Img =m.cdata(:,:,1);

      % filter (i.e. blur) image, and convert from uint8 to double
      ImgFilt =imfilter(im2double(Img),h);

      X(:,i) =ones( noY, 1 ) * i;

      % find the line in each frame
      if i=1
         [Y(:,i), Z(:,i), Zfish(:,i), Zline(:,i), refY, refZ] =getLine(ImgFilt, H);
      else
         [Y(:,i), Z(:,i), Zfish(:,i), Zline(:,i)] =getLine(ImgFilt, H, refY, refZ);
      end

      % patch holes in the line of the fish
      Zfish =patchHoles(Zfish);
      Y =patchHoles(Y);

```

```
62        waitbar(i/(noFrames-1),h_wait);
63
64    end
65
66    close(h_wait)
67
68    uisave({'X','Y','Z','Zfish','Zline'},strcat(fileName,'.mat'));
69
70 else
71    fig1 =figure('Position',[600,600,170,80],'Toolbar','none',...
72       'NumberTitle','off','Name','No file selected','MenuBar','none');
73    h =uicontrol('Parent',fig1,'Style', 'pushbutton', 'String', 'OK',...
74    'Position', [30 20 110 40], 'Callback', 'close');
75 end
```

Listing B.3: findLine2D.m

```
1  function Laser =findLine2D(Img)
2  %
3  % Purpose : Detects the position of a projected light-line in a greyscale
4  %     image
5  %
6  % Use (syntax):
7  %     Laser = findLine2D(Img)
8  %
9  % Input Parameters :
10 %     Img: a greyscale image (mxn matrix)
11 %
12 % Return Parameters :
13 %     Laser: position of the light-line (2xn matrix)
14 %
15 % Description and algorithms:
16 %     Calculates the moment in two dimensions of the intensity values
17 %     within a certain range around the maximum intensity for each column
18 %     and deletes outliers
19 %
20 % References :
21 %
22 % Author : Uwe Meier
23 % Date : 22. April 2010
24 % Version : 1.0
25 %
26 % (c) 2010 Uwe Meier, uwemei@gmail.com
27 %-------------------------------------------------------------
28 %
29 % Init
30 %
31
32 % define the range in which the moments should be calculated
33 udOffset =10;
34 lrOffset =3;
35 % weighting factor for the single intensity-values
36 weight =3;
37
38 imgLength =size(Img,2);
39 imgDepth =size(Img,1);
40
41 % initialize matrices
42 Laser(1:2,1:imgLength) =NaN;
43 LaserPoints(1:2,1:imgLength) =NaN;
44 int(1:imgLength) =NaN;
45
46 % factors to determine and delete outliers
47 maxDeviation =5;
48 minSegmentlength =15;
```

```
length =1;
startSegment =1;

Img1 =Img;
%
% Loop - Determine single laserpoints columnwise
%
for i =1:imgLength

   % only use red-values of image to detect the maximum intensity
   [int(i),pointer] =max(Img1(:,i));

   % calculate vertical boundaries
   up =pointer - udOffset;
   if up < 1
      up =1;
   end
   down =pointer + udOffset;
   if down > imgDepth
      down =imgDepth;
   end

   % calculate horizontal boundaries
   left =i - lrOffset;
   if left < 1
      left =1;
   end
   right =i + lrOffset;
   if right > imgLength
      right =imgLength;
   end

   %
   % Calculate moments start
   %
   Cut =Img1(up:down,left:right).^weight;

   % calculate the sum of the x*I^weight
   p =(left:right);
   P =diag(p);
   EnumeratorX =sum(sum(P*Cut'));

   % calculate the sum of the y*I^weight
   q =(up:down);
   Q =diag(q);
   EnumeratorY =sum(sum(Q*Cut));

   % calculate the sum of I^weight
   Denominator =sum(Cut(:));

   MomentX =EnumeratorX / Denominator;
   MomentY =EnumeratorY / Denominator;
   %
   % calculate moments end
   %

   LaserPoints(1,i) =MomentX;
   LaserPoints(2,i) =MomentY;

   % delete outliers
   if i>1
      % if the distance between two values is to large or the end of the
      % image is reached, and the segment is long enough, then save all
      % values of this segment
      if (abs(LaserPoints(2,i)-LaserPoints(2,i-1)) > maxDeviation \/i =imgLength) /\(length > minSegmentlength)
         Laser(1:2,startSegment:i-1) =LaserPoints(1:2,startSegment:i-1);
```

```
115          length =1;
116          startSegment =i;
117       % if the distance between two values is below a thresh, then
118       % increment counter for the segment-length
119       elseif abs(LaserPoints(2,i)-LaserPoints(2,i-1)) ≤maxDeviation
120          length =length + 1;
121       % if the distance between two values is to large and the segment is
122       % not long enough, reset values (i.e. dont save the values of this
123       % segment)
124       else
125          length =1;
126          startSegment =i;
127       end
128    end
129
130 end
131
132 % Delete al values that are below a certain intensity-value
133 Laser(1,int<0.35) =NaN;
134 Laser(2,int<0.35) =NaN;
```

Listing B.4: getLine.m

```
1  function [X, Y, Yfish, Yline, refX, refY] =getLine(ImgFilt, H, refX, refY)
2  %
3  % Purpose : Identifies a line in a greyscale image, converts the points to
4  %     another coordinate system using a given homographymatrix,
5  %     seperates the line into two parts and align the points to
6  %     referencepoints
7  %
8  % Use (syntax):
9  %     [X, Y, Yfish, Yline, refX, refY] = getLine(ImgFilt, H)
10 %     [X, Y, Yfish, Yline] = getLine(ImgFilt, H, refX, refY)
11 %
12 % Input Parameters :
13 %     ImgFilt: a mxn greyscale image
14 %     H: 3x3 homography matrix
15 %     refX: Referencepoint in x direction
16 %     refY: Referencepoint in y direction
17 %
18 % Return Parameters :
19 %     X: x coordinates of line
20 %     Y: all y coordinates of line
21 %     Yfish: y coordinates of line representing the line on fish
22 %     Yline: y coordinates of line representing the line on ground
23 %     refX: Referencepoint in x direction
24 %     refY: Referencepoint in y direction
25 %
26 % Description and algorithms:
27 %
28 % References :
29 %     findLine2D.m
30 %     findThresh.m
31 %     separateLine.m needed
32 %
33 % Author : Uwe Meier
34 % Date : 27. April 2010
35 % Version : 1.0
36 %
37 % (c) 2010 Uwe Meier, uwemei@gmail.com
38 %-------------------------------------------------------------
39
40 [height width] =size(ImgFilt);
41
42 % find points along the laserline
```

```
LaserPoints =findLine2D(ImgFilt);

% convert image-laserline to rw-laserline
LaserPointsH =[ LaserPoints(2,:); LaserPoints(1,:); ones(size(LaserPoints(2,:))) ];
LaserPointsHR =H*LaserPointsH;

X =LaserPointsHR(1,:)./LaserPointsHR(3,:);
Y =LaserPointsHR(2,:)./LaserPointsHR(3,:);

% get reference point to subtract offset so that first found laserpoint
% is at X=0 Y=0
if nargin < 3
   refX =X(1);
   refY =Y(1);
end

[counts,bins] =hist(Y,(1:1:height));

% get thresh to separate fish from line only if it is possible to
% distinguish between them
if ( range(Y) < 4 )
   thresh =0;
else
   [thresh] =findThresh(bins,counts);
end

% separate laser into fish and line
[Yfish, Yline] =separateLine(Y, thresh);

% relate all X and Y points to reference point
X =X - refX;
Y =-Y + refY;
Yfish =-Yfish + refY;
Yline =-Yline + refY;
```

Listing B.5: findThresh.m

```
function [thresh, peak1At, peak1Val, peak2At, peak2Val] =findThresh(bins,counts)
%
% Purpose : The purpose of this function is to find a threshold of a
%     bi-modal histogram that has two distinctive peaks
%
% Use (syntax):
%     [thresh, peak1At, peak1Val, peak2At, peak2Val] =
%     findThresh(bins,counts)
%
%     thresh = findThresh(bins,counts)

% Input Parameters :
%     bins: locations of the bins (1xn vector)
%     counts: count of the bins (1xn vector)
%
% Return Parameters :
%     thresh: the threshhold that seperates the histogram
%     peak1At: position of first peak
%     peak1Val: value o first peak
%     peak2At: position of second peak
%     peak2Val: value o Second peak
%
% Description and algorithms:
%     This function first identifies the two peaks of a bi-modal
%     histogram using find2Peaks, then determines the line connecting
%     those points. The normal distance of the histogram points to this
%     line are determined and the value of the bin with the maximum
%     normal distance is used as threshold.
```

```
29 %
30 % References :
31 %  Needs function find2Peaks
32 %
33 % Author : Uwe Meier
34 % Date : 22. April 2010
35 % Version : 1.0
36 %
37 % (c) 2010 Uwe Meier, uwemei@gmail.com
38 %-------------------------------------------------------
39
40 %
41 % Determine the 2 Peaks
42 %
43 rangeBins =find(counts>0);
44 first =rangeBins(1);
45 last =rangeBins(end);
46 [peak1At, peak1Val, peak2At, peak2Val] =find2Peaks( bins(first:last), counts(first:last) );
47 peak1At =peak1At + first - 1;
48 peak2At =peak2At + first - 1;
49 %
50 % Determine the line connecting the peaks
51 %
52 lineP =[ peak1Val - peak2Val, ...
53       bins(peak2At) - bins(peak1At), ...
54       (peak2Val*bins(peak1At)) - (peak1Val * bins(peak2At)) ];
55 scale =sqrt( lineP(1)^2 + lineP(2)^2);
56 lineP =lineP / scale;   % ensure a Euclidean metric
57 %
58 % Set up the points as a set of column vectors (note only the points
59 % between then two peaks are used.
60 %
61 points =[ bins(peak1At:peak2At); counts(peak1At:peak2At); ones(size((peak1At:peak2At)))];
62 %
63 % Determine the normal distances and find the maximum
64 %
65 ds =lineP * points;
66 [maxD, maxDAt] =max( abs(ds) );
67 maxDAt =maxDAt + peak1At - 1;
68
69 thresh =bins( maxDAt );
70
71 % %
72 % % Plot results (for tests only)
73 % %
74 % % LineT doesn't seem normal to lineP because axis are not equal
75 %
76 % SetUpGraphics(15);
77 %
78 % lineT = [ -lineP(2), lineP(1), 0];
79 % lineT(3) = - lineT * [bins( maxDAt ); counts( maxDAt );1];
80 %
81 % fig1 = figure(1); clf(fig1);
82 % bar(bins(first:last),counts(first:last));
83 % hold on;
84 % axis tight;
85 % title('Frame No. 90 of N5.avi');
86 % xlabel('[mm]');
87 % ylabel('#');
88 % plot( bins(peak1At), peak1Val, 'ro', 'MarkerFaceColor', 'r');
89 % plot( bins(peak2At), peak2Val, 'ro', 'MarkerFaceColor', 'r');
90 % plot( bins( maxDAt ), counts( maxDAt ), 'rx','MarkerSize',12);
91 % plotLine( lineP, 'r');
92 % plotLine( lineT, 'r');
93 % plot( [thresh, thresh], [0, max([peak1Val, peak2Val])], 'r-.');
```

Listing B.6: find2Peaks.m

```
function [peak1At, peak1Val, peak2At, peak2Val] =find2Peaks( bins, counts )
%
% Purpose : The purpose of this function is to find the two peaks of a
%     bi-modal hostigram
%
% Use (syntax):
%     [peak1At, peak1Val, peak2At, peak2Val] = find2Peaks( bins, counts )
%
% Input Parameters :
%     bins: locations of the bins (1xn vector)
%     counts: count of the bins (1xn vector)
%
% Return Parameters :
%     peak1At: position of first peak
%     peak1Val: value o first peak
%     peak2At: position of second peak
%     peak2Val: value o Second peak
%
% Description and algorithms:
%     simply devides bins and counts in two and determines the maximum
%     for each part
%
% References :
%
% Author : Uwe Meier
% Date : 22. April 2010
% Version : 1.0
%
% (c) 2010 Uwe Meier, uwemei@gmail.com
%------------------------------------------------------------

n =length( bins );
n2 =round( n/2);
%
bins1 =bins(1:n2-1);
counts1 =counts(1:n2-1);
%
bins2 =bins(n2:end);
counts2 =counts(n2:end);
%
[peak1Val peak1At] =max( counts1 );
%
[peak2Val peak2At] =max( counts2 );
peak2At =peak2At + n2 - 1;
```

Listing B.7: separateLine.m

```
function [yfish, yline] =separateLine(y, thresh)
%
% Purpose : seperates a line (represented by values of a vector) acording
%     to a thresh into a groundline and an objectline
%
% Use (syntax):
%     [yfish, yline] = seperateLine(y, thresh,nbins)
%
% Input Parameters :
%     y: the whole line (nx1 vector)
%
% Return Parameters :
%     yfish: line on object (mx1 vector)
%     yline: line on ground (lx1 vector)
%
% Description and algorithms:
```

```
%     Seperates line using a thresh and deletes groundlinepoints that are
%     outside the double standard deviation of the mean line as well as
%     objectpoints that are closer to the mean of the groundline than the
%     mean of the objectline
%
% References :
%
% Author : Uwe Meier
% Date : 23. April 2010
% Version : 1.0
%
% (c) 2010 Uwe Meier, uwemei@gmail.com
%-------------------------------------------------------

% initialize vectors with NaNs
yfish(1:length(y),1) =NaN;
yline(1:length(y),1) =NaN;

% seperate line
yfish( round(y) < thresh) =y( round(y) < thresh);
yline( round(y) ≥thresh) =y( round(y) ≥thresh);

% Delete points, that are too much below/above the line
mnL =sum(yline(¬isnan(yline)))/length(yline(¬isnan(yline)));
sigma =std(yline(¬isnan(yline)));
yline( (yline > (mnL+2*sigma)) ∨(yline < (mnL-2*sigma)) ) =NaN;

% Delete points of object, that are closer to the mean of the line than
% to the mean of the object-points
mnF =sum(yfish(¬isnan(yfish)))/length(yfish(¬isnan(yfish)));
yfish( abs(yfish-mnF) > abs(yfish-mnL) ) =NaN;
```

Listing B.8: patchHoles.m

```
function result =patchHoles( data );
%
% Purpose :
%  Measurement data acquired with a plane of ligh sensor may commonly have
% holes generated by occlusion. The aim of this routine is to patch up
% these holes so as to ensure that the data grid is complete, regular and
% consistent.
%
% Use (syntax): result = patchHoles( data );
%
% Input Parameters : the data field, it is assumed that the colums are to
%               be patched.
%
% Return Parameters : result, the patched data
%
% Description and algorithms: Linear interpolation is used to patch the
% data.
%
% References : none
%
% Author : Uwe Meier
% Date : 9 November 2010
% Version : 1.0
%
% (c) 2010, Uwe Meier, uwemei@gmail.com
%-------------------------------------------------------------------------

result =zeros( size( data ) );
%
% patching is done in each column
%
```

```
[noRows, noCols] =size( data );
%
for k =1:noCols
   %
   col =data(:,k);
   %
   if ( sum(isnan(col)) < length(col) )

      at =1;
      while at ≤(noRows - 2)
         at =at + 1;
         if isnan( col(at) ) ∧¬isnan( col(at-1) )
            % start of a hole has been found
            start =at;
            % find the end of the hole
            while isnan(col(at))
               if at =length(col)
                  break;
               end
               at =at + 1;
            end;
            stop =at;
            %
            % generate the linear interpolation
            %
            vals =linspace( col(start-1), col(stop), stop - start + 1);
            %
            % fill the hole
            %
            col(start:stop) =vals';
         end;
      end;
   end
   %
   result(:,k) =col;
   %
end;
```

Listing B.9: completeFish.m

```
function [Xfish, YfishNew, YfishStar] =completeFish(X,Yfish,Yline,degree,sl)
%
% Purpose : completes the fish, calculates the reverse side of it and returns
%     the new X and Y values of the fish as well as the Y values of the
%     reverse side
%
% Use (syntax):
%     [Xfish, YfishNew, YfishStar] =
%     completeFish(X,Yfish,Yline,degree,sl)
%
% Input Parameters :
%     X: x-values of a laserline (nx1 vector)
%     Yfish: y-values of a laserline on object (nx1 vector)
%     Yline: y-values of a laserline on ground (nx1 vector)
%     degree: degree used to calculate derivative of laserline (scalar)
%     sl: supportlengt used to calculate derivative of laserline (scalar)
%
% Return Parameters :
%     Xfish: new x-values of a laserline on object (nx1 vector)
%     YfishNew: new y-values of a laserline on object (nx1 vector)
%     YfishStar: mirrored y-values of the laserline on object (nx1 vector)
%
% Description and algorithms:
%
% References :
```

```
26 %     needs getOsculatingCircle.m
27 %
28 % Author : Uwe Meier
29 % Date : 23. April 2010
30 % Version : 1.0
31 %
32 % (c) 2010 Uwe Meier, uwemei@gmail.com
33 %--------------------------------------------------------
34
35 [width, height] =size(X);
36
37 % initialize return values
38 Xfish(1:width,1) =NaN;
39 YfishNew(1:width,1) =NaN;
40 YfishStar(1:width,1) =NaN;
41
42 % flag, if fish touches the ground or not
43 touchesLine =1;
44
45 % only use values that are not NaN
46 noNaNYfish =find(¬isnan(Yfish));
47 y =Yfish(noNaNYfish);
48 x =X(noNaNYfish);
49
50 % only do, if there is a fishline
51 if length(y) > degree
52
53    noPoints =length(x);
54
55    % create supportlength for derivative basis depending on the
56    % number of values
57    slD =round(sl/2);
58    if mod(slD,2) =0
59       slD =slD+1;
60    end
61    if slD ≥noPoints
62       slD =round(noPoints/4);
63       if mod(sl,2) =0
64          slD =slD+1;
65       end
66    end
67
68    % create supportlength for Savitzky-Golay basis depending on the
69    % number of values
70    if sl ≥noPoints
71       sl =round(noPoints/4);
72       if mod(sl,2) =0
73          sl =sl+1;
74       end
75    end
76
77    if sl < degree
78       sl =degree + 1;
79    end
80
81    % create Basis for Savitzky-Golay smoothing matrix that aproximates a
82    % derivative operator for the generation of the osculating circles
83    S =generateS( noPoints, sl, degree );
84    xc =linspace(-1,1,noPoints )';
85    D =generateD( xc, sl, 2 );
86
87    % smooth values using S-G
88    ys =S * y;
89
90    % calculate derivatives
91    yd =D * ys;
```

```
92    ydd =D * yd;
93    xd =D * x;
94    xdd =D * xd;
95
96    % Curvature C and Osculating Circle Radius R
97    C =(xd .* ydd - yd .*xdd)./((xd.^2 + yd.^2).^(3/2));
98    R =1./C;
99
100   % scale derivative from xc (linspace) to x (original points)
101   deltaX1 =x(2)-x(1);
102   deltaXc1 =xc(2)-xc(1);
103   k1 =yd(1)*deltaXc1/deltaX1;
104   deltaXend =x(end-1)-x(end);
105   deltaXcend =xc(end-1)-xc(end);
106   kend =yd(end)*deltaXcend/deltaXend;
107
108   % get points for osculating circles at start and end of fish
109   if R(1) <0
110      [ xCircle1, yCircle1, center1 ] =getOsculatingCircle( [x(1) ; y(1)], k1, R(1) );
111   else
112      xCircle1 =[];
113      yCircle1 =[];
114      center1 =[];
115   end
116   if R(end) <0
117      [ xCircle2, yCircle2, center2 ] =getOsculatingCircle( [x(end) ; y(end)], kend, R(end) );
118   else
119      xCircle2 =[];
120      yCircle2 =[];
121      center2 =[];
122   end
123
124   % find middleline between max of fishline and groundline -> mirrorLine
125   maxX= max(y);
126   noNaNYline =find(~isnan(Yline));
127   groundLine =mean(Yline(noNaNYline));
128   mirrorLine =(maxX-groundLine)/2 + groundLine;
129   mLold =mirrorLine;
130
131   % if the center of one osculating circle lies above mirrorLine, then
132   % set a flag that the fish doesn't touch the ground and reset
133   % mirrorLine
134   if ~isempty(center1) &center1(2) > mirrorLine
135      touchesLine =0;
136      mirrorLine =center1(2);
137   end
138   if ~isempty(center2) &center2(2) > mirrorLine
139      touchesLine =0;
140      mirrorLine =center2(2);
141   end
142
143   % if there are points below the morrorLine, reset it
144   if min(y) < mirrorLine
145      mirrorLine =min(y);
146      if min(y) < mLold
147         touchesLine =1;
148      end
149   end
150
151   % only use these parts of circles, that are outside the fish and
152   % above mirror line
153   xCircle1Cut =[];
154   yCircle1Cut =[];
155   xCircle2Cut =[];
156   yCircle2Cut =[];
157
```

```
158    if ¬isempty(center1) ∧center1(1) > x(1)
159       xCircle1Cut =xCircle1( ((xCircle1 < x(1)) ∧(yCircle1 ≥mirrorLine) ∧(yCircle1 ≥center1(2))) );
160       yCircle1Cut =yCircle1( ((xCircle1 < x(1)) ∧(yCircle1 ≥mirrorLine) ∧(yCircle1 ≥center1(2))) );
161    end
162    if ¬isempty(center2) ∧center2(1) < x(end)
163       xCircle2Cut =xCircle2( ((xCircle2 > x(end)) ∧(yCircle2 ≥mirrorLine) ∧(yCircle2 ≥center2(2))) );
164       yCircle2Cut =yCircle2( ((xCircle2 > x(end)) ∧(yCircle2 ≥mirrorLine) ∧(yCircle2 ≥center2(2))) );
165    end
166
167    if (¬isempty(xCircle1Cut) ∧xCircle1Cut(1) < min(X))
168       xCircle1Cut =[];
169       yCircle1Cut =[];
170    end
171
172    if (¬isempty(xCircle2Cut) ∧xCircle2Cut(end) > max(X))
173       xCircle2Cut =[];
174       yCircle2Cut =[];
175    end
176
177    % add found Points to the fish and save
178    first =(noNaNYfish(1) - length(yCircle1Cut));
179    last =(noNaNYfish(end) + length(yCircle2Cut));
180    YfishNew(noNaNYfish) =y;
181    YfishNew(first:(noNaNYfish-1)) =yCircle1Cut;
182    YfishNew((noNaNYfish(end)+1:last)) =yCircle2Cut;
183    Xfish(noNaNYfish) =x;
184    Xfish(first:(noNaNYfish-1)) =xCircle1Cut;
185    Xfish((noNaNYfish(end)+1:last)) =xCircle2Cut;
186
187    y =[ yCircle1Cut'; y; yCircle2Cut' ];
188
189    % Mirror the fishline to the bottom
190    yStar =-y +2*mirrorLine;
191
192    % if fish touches line then rescale points of mirrored fish
193    if touchesLine
194       scalingfactor =(mirrorLine-groundLine)/(maxX-mirrorLine);
195       yStar =(yStar-mirrorLine)*scalingfactor + mirrorLine;
196    end
197
198    YfishStar(noNaNYfish) =yStar( (length(yCircle1Cut)+1):(length(yStar)-length(yCircle2Cut)) );
199    YfishStar(first:(noNaNYfish-1)) =yStar(1:length(yCircle1Cut));
200    YfishStar((noNaNYfish(end)+1:last)) =yStar((length(yStar)-length(yCircle2Cut)+1):length(yStar));
201
202 end
```

Listing B.10: getOsculatingCircle.m

```
1  function [ X, Y, center ] =getOsculatingCircle(point,slope,radius,noPoints)
2  %
3  % Purpose : Calculates the center of the osculating circle in a given point
4  %     with given slope and returns values for the circumference
5  %
6  % Use (syntax):
7  %     [ X, Y, center ] = getOsculatingCircle(point,slope,radius,noPoints)
8  %     [ X, Y, center ] = getOsculatingCircle(point,slope,radius)
9  %
10 % Input Parameters :
11 %     point: point in which the osculating circle should be calculated
12 %     slope: i.e. derivative in point
13 %     radius: radius of the osculating cicle
14 %     noPoints: number of points the circle should consist of
15 %
16 % Return Parameters :
17 %     X: x-values of circle
```

```
%     Y: y-values of circle
%     center: center of circle
%
% Description and algorithms:
%     Creates the tangent in the given point, calculates the orthogonal
%     line and determines the center of the circle by using the radius
%
% References :
%     needs getCircle
%
% Author : Uwe Meier
% Date : 22. April 2010
% Version : 1.0
%
% (c) 2010 Uwe Meier, uwemei@gmail.com
%--------------------------------------------------------


% calculate tangent with slope in point
p1=[ point(1); point(2); 1 ];
d =point(2) - point(1)*slope;
xp =point(1) + 1;
yp =slope*xp + d;
% 2. point to create line
p2 =[ xp ; yp ; 1 ];

% create parameters for line orthogonal to tangent (i.e. line between point
% and center of osculating circle)
dx =p2(1) - p1(1);
dy =p2(2) - p1(2);
dist =sqrt( dx^2 + dy^2 );
ux =- dy/dist;
uy =dx/dist;

% calculate center of osculating circle
xp3 =p1(1) + radius*ux;
yp3 =p1(2) + radius*uy;

% center of circle
center =[ xp3 yp3 1];

% get points for osculating circle in point of fish either with or without
% given noPoints
if nargin =4
   [ X Y ] =getCircle(center,radius,noPoints);
else
   [ X Y ] =getCircle(center,radius);
end
```

Listing B.11: getCircle.m

```
function [ X , Y ] =getCircle(center,radius,noPoints)
%
% Purpose : Returns values for the circumference of a circle with given
%     center and radius
%
% Use (syntax):
%     [ X , Y ] = getCircle(center,radius,noPoints)
%     [ X , Y ] = getCircle(center,radius)
%
% Input Parameters :
%     center: center of the circle (2x1, 3x1, 1x2, 1x3)
%     radius: radius of the circle
%     noPoints: number of points the circle should consist of
%
```

```
% Return Parameters :
%     X: x-values of circle
%     Y: y-values of circle
%
% Description and algorithms:
%
% References :
%
% Author : Uwe Meier
% Date : 22. April 2010
% Version : 1.0
%
% (c) 2010 Uwe Meier, uwemei@gmail.com
%-------------------------------------------------------

% If no noPoints given, chose a noPoints depending on the radius
if nargin =2
   noPoints =round(2*abs(radius)*pi*5);
   if(noPoints > 10000)
      noPoints =10000;
   end
end

rho =radius*ones(1,noPoints);
theta =linspace(0,2*pi,noPoints);

[X,Y] =pol2cart(-theta, rho);

X =X + center(1);
Y =Y + center(2);
```

B.2 Part II

Listing B.12: saveContour.m

```
clear all;
close all;

pathName='';
[fileName,pathName] =uigetfile(strcat(pathName,'*.jpg'),'Load image data');

name =[pathName fileName];
Img =imread(name);

% resize Image to 1/4 of original size (for faster computation during test
% phase)
Img =imresize(Img, 0.25);

% convert to Grayscale
ImgG =rgb2gray(Img);

% convert to double and scale from 0-1
ImgG =double(ImgG)/255;

% create Histogramm
[counts bins] =hist(ImgG(:),0:(1/255):1);

% Calculate threshold to seperate fish from background
thresh =findThreshGauss(bins,counts);
```

```
25
26 % Seperate Fish from background and fill holes
27 ImgBW =¬im2bw(ImgG,thresh);
28
29 ImgBW =imfill(ImgBW,'holes');
30 % delete parts of the background that got through the treshold - start
31 % get all found objects
32 [labels, noLs] =bwlabel( ImgBW, 4 );
33 % get size of the objects
34 for k=1:noLs
35    pos =find( labels =k );
36    area(k) =length(pos);
37 end;
38 % sort areas by size (largest is last)
39 [list, indices] =sort( area );
40 % get label of largest area
41 labelBiggest =indices(end);
42 % get positions of pixels with this label
43 inds =find( labels =labelBiggest );
44 % create new bw-Image that includes only the largest object
45 ImgBW =zeros( size( ImgBW ));
46 ImgBW(inds) =1;
47 % delete parts of the background that got through the treshold - end
48
49 % get the contour and its x- and y-values
50 % Contour also contains some wrong values (don't know why), thus you have
51 % to use the Handle, and get its childrens X- and YData. Since there exists
52 % only one contour, you don't have to deal with different children
53 figure;
54 [Contour,H] =contour(ImgBW,1);
55 T =get(H);
56 Child =get(T.Children);
57 xData =Child.XData;
58 yData =Child.YData;
59
60
61 % ---------------------------------------------------------------------
62 %
63 % plot results
64 %
65
66 % figure;
67 % colormap gray;
68 % imagesc(ImgBW);
69 % axis equal tight;
70 % xlabel('[pixel]');
71 % ylabel('[pixel]');
72 % figure;
73 % colormap gray;
74 % imagesc(ImgG);
75 % axis equal tight;
76 % hold on;
77 % plot(xData,yData,'g','LineWidth',2);
78 % xlabel('[pixel]');
79 % ylabel('[pixel]');
80
81 % ---------------------------------------------------------------------
82 %
83 % save results
84 %
85 save(strcat(fileName,'.mat'),'Img','ImgG','ImgBW','xData','yData');
```

Listing B.13: saveSpectrum.m

```
1 clear all;
```

```
close all;

name ='Schraubenzieher2.JPG';
mat ='.mat';
load(strcat(name,mat));

% more general way to delete NaNs:
noNaNx =find(¬isnan(xData));
noNaNy =find(¬isnan(yData));
xData =xData(noNaNx);
yData =yData(noNaNy);

% resample to 1000 points
lngth =length(xData);
noPoints =1:lngth/1000:lngth;
xData =interp1(1:lngth,xData,noPoints,'spline')';
yData =interp1(1:lngth,yData,noPoints,'spline')';

% create Fourier spectrum of Contour
spectrum =fft( xData + 1i*yData );
magnitude =fftshift( abs( spectrum ));

save(strcat(name,'-S',mat),'Img','ImgG','ImgBW','xData','yData','spectrum','magnitude');
```

Listing B.14: testCurvature.m

```
clear all;
close all;

% generateCyclicB creates a length(data) x length(Data) x degree - matrix.
% Using the single layers, you create a spectrum with the dimensions
% degree x length(Data).
% The first row represents the original x- and y-values with an aplied
% low-pass-filter with a cut given by the SupportLength (sl). These datas
% are NOT shift invariant (xOrig = xS/sqrt(sl)) but can be made shift
% invariant by subtracting their mean.
% The second and third row are shift, but not rotational invariant. Only
% the curvature is shift als well as rotational invariant.

%
% first object
%

sl =101;
degree =2;

load Saibling2-2.JPG-S.mat;

xData1 =xData - mean(xData);
yData1 =yData - mean(yData);
noPoints1 =length(xData1);

% generates a noPoints-by-noPoints-by-degree+1 circulant matrix with
% supportlength sl
B1 =generateCyclicB( noPoints1, sl, degree);

% initialize spectrum
sX1 =zeros( degree + 1, noPoints1 );
sY1 =zeros( degree + 1, noPoints1 );
% create spectrum by multiplying every single layer of B with the x- and
% y-Data
for i=1:degree + 1
   b1(:,:) =B1(:,:,i);
   sX1(i,:) =b1 * xData1;
   sY1(i,:) =b1 * yData1;
```

```
end;

% get derivatives of x and y
xd1 =sX1(2,:);
xdd1 =sX1(3,:);
yd1 =sY1(2,:);
ydd1 =sY1(3,:);
% calculate Curvature
k1 =(xd1.*ydd1 - yd1.*xdd1)./ (xd1.^2 + yd1.^2).^(3/2);

%
% second object
%
load Saibling1-2.JPG-S.mat;

xData2 =(xData - mean(xData));
yData2 =(yData - mean(yData));
noPoints2 =length(xData2);

% generates a noPoints-by-noPoints-by-degree+1 circulant matrix with
% supportlength sl
B2 =generateCyclicB( noPoints2, sl, degree);

% initialize spectrum
sX2 =zeros( degree + 1, noPoints2 );
sY2 =zeros( degree + 1, noPoints2 );
% create spectrum by multiplying every single layer of B with the x- and
% y-Data
for i=1:degree + 1
   b2(:,:) =B2(:,:,i);
   sX2(i,:) =b2 * xData2;
   sY2(i,:) =b2 * yData2;
end;

% get derivatives of x and y
xd2 =sX2(2,:);
xdd2 =sX2(3,:);
yd2 =sY2(2,:);
ydd2 =sY2(3,:);
% calculate Curvature
k2 =(xd2.*ydd2 - yd2.*xdd2)./ (xd2.^2 + yd2.^2).^(3/2);

% convolute the two curvatures and relocate the start of the Datasets to be
% the same (if the objects are identical) or as close as possible (if
% different)

c =cconv(k1,fliplr(k2),length(k1));
[ val, pos ] =max(c);
k2g =[ k2(end-pos+1:end) , k2(1:end-pos) ];
sX2g =[ sX2(:,end-pos+1:end) , sX2(:,1:end-pos) ];
sY2g =[ sY2(:,end-pos+1:end) , sY2(:,1:end-pos) ];
xData2g =[ xData2(end-pos+1:end) ; xData2(1:end-pos) ];
yData2g =[ yData2(end-pos+1:end) ; yData2(1:end-pos) ];

err =sqrt((sum((k1 - k2g).^2)))
```

Listing B.15: testSpectra.m

```
clear all;
close all;

% Img, ImgBG, ImgG, xData, yData, spectrum, magnitude
load Saibling2-4.JPG-S.mat;

xData1 =xData;
```

```
yData1 =yData;

spectrum1 =spectrum;

%shift invariance
spectrum1(1) =0;

% scale according to spectrum1(2)^2 + spectrum1(end)^2 = 1 to get scale
% invariance
r1 =abs(spectrum1(2))^2 + abs(spectrum1(end))^2;
spectrum1 =spectrum1/sqrt(r1);

% low-pass-filter
cut =250;
spectrum1( cut+1:end-cut) =0;

gen1 =ifft( spectrum1 );
gx1 =real( gen1 );
gy1 =imag( gen1 );

load Saibling1-5.JPG-S.mat

spectrum2 =spectrum;

%shift invariance
spectrum2(1) =0;

% scale according to spectrum2(2)^2 + spectrum2(end)^2 = 1 to get scale
% invariance
r2 =abs(spectrum2(2))^2 + abs(spectrum2(end))^2;
spectrum2 =spectrum2/sqrt(r2);

% low-pass-filter
spectrum2( cut+1:end-cut) =0;

gen2 =ifft( spectrum2 );
gx2 =real( gen2 );
gy2 =imag( gen2 );

% magnitudes for rotation invariance
magnitude1 =fftshift(abs(spectrum1));
magnitude2 =fftshift(abs(spectrum2));

% calculate euclidean distance between the two magnitudes
delta =magnitude2 - magnitude1;
d12 =sqrt( sum( delta.^2 ) )
```