UNIVERSITÄT OSNABRÜCK

Institute for Computer Science
Knowledge-Based Systems

Doctoral Dissertation

Multi-wavelength laser line profile sensing for agricultural applications

Wolfram Strothmann

07/22/2016

First reviewer:	Prof. Dr. Joachim Hertzberg
Second reviewer:	Prof. Dr. Arno Ruckelshausen
Third reviewer:	Prof. Dr. Hans-Werner Griepentrog

Abstract

This dissertation elaborates on the novel sensing approach of multi-wavelength laser line profiling (MWLP). It is a novel sensor concept that expands on the well-known and broadly adopted laser line profile sensing concept for triangulation-based range imaging. Thereby, the MWLP concept does not just use one line laser but multiple line lasers at different wavelengths scanned by a single monochrome imager. Moreover, it collects not only the 3D distance values but also reflection intensity and backscattering of the laser lines are evaluated. The system collects spectrally selective image-based data in an active manner. Thus, it can be geared toward an application-specific wavelength configuration by mounting a set of lasers of the required wavelengths. Consequently, with this system image-based 3D range data can be collected along with reflection intensity and backscattering data at multiple, selectable wavelengths using just a single monochrome image sensor.

Starting from a basic draft of the idea, the approach was realized in terms of hardware and software design and implementation. The approach was shown to be feasible and the prototype performed well as compared with other state-of-the-art sensor systems. The sensor raw data can be visualized and accessed as overlayed distance images, point clouds or mesh. Further, for selected example applications it was demonstrated that the sensor data gathered by the system can serve as descriptive input for real world agricultural classification problems. The sensor data was classified in a pixel-based manner. This allows very flexible, quick and easy adaptation of the classification toward new field situations.

Acknowledgements

Starting with the idea of multi-wavelength laser line profiling (MWLP) in summer 2012, a lot of work was conducted on this thesis and many problems had to be tackled. It would not have been possible without the continuous support and encouragement of many people. I want to express my deepest gratitude to all those who supported me during the time I spent on this dissertation. Only a few of these great people can be mentioned by name here.

First, I want to thank Prof. Dr. Joachim Hertzberg and Prof. Dr. Arno Ruckelshausen. I have come to know Prof. Dr. Arno Ruckelshausen during a Master's course in 2009. In 2011 he supervised my Master's thesis and since end of 2011 I have been part of his working group. His positive and critical way of thinking - always having in mind what might happen for $\lim_{n \to \pm\infty}$ together with being hands-on if values are in a specific order of magnitude - has helped to shift barriers numerous times. Further, I owe many thanks to Prof. Dr. Joachim Hertzberg for supervising this dissertation, thereby giving me the chance to work on this engaging research topic. His continuous support and valuable, constructive feedback during the time working on this dissertation helped to shape ideas into reality.

Second, I had the pleasure of being part of the collaborative research project RemoteFarming.1. Consequently, I owe my gratitude to the institutions that funded and managed the project, the German Ministry of Food and Agriculture (BMEL) and the German Federal Office of Agriculture and Food (BLE). The research project was conducted in collaboration of the project partners Robert Bosch GmbH, Amazonenwerke H. Dreyer GmbH & Co.KG and University of Applied Sciences Osnabrück. The great working atmosphere during this project was thanks to the following team members: Waldemar Bangert, Hannes Becker, Martin Hänsel, Sebastian Haug, Arnd Kielhorn, Maik Kohlbrecher, Frederik Langsenkamp, Alexander Matrosov, Andreas Michaels, Dr. Dejan Pangercic, Dr. Florian Rahe, Dr. Slawomir Sander, Daniel Schmunkamp, Fabian Sellmann and Prof. Dr. Dieter Trautz. Many thanks go to my colleague Christian Scholz for proofreading the final manuscript. Further, I thank the following former and current colleagues of other projects at the Laboratory for Micro- and Optoelectronics of UAS Osnabrück for the enriching discussions and avoiding silence at work: Axel Hoeh, Andreas Linz, Fritz Matthäus, Kim Möller, Andreas Trabhardt and Vadim Tsukor.

Finally, I owe my deepest gratitude to my family for their continuous encouragement and to my dear Ann-Kathrin Fischer for her love and faith in me.

Contents

List of Figures

List of Tables

List of Formulas

List of Code Listings

List of Digressions

Abreviations

API Application Programmers Interface.

BGR Blue, Green, Red.

CAD Computer Aided Design.

CCD Charge Coupled Device.

CMOS Complementary Metal-Oxide Semiconductor.

COG Center of Gravity.

CPU Central Processing Unit.

CT Computer Tomography.

CU Control Unit.

CUDA Compute Unified Device Architecture.

CW Continous Wave.

DMA Direct Memory Access.

DOF Degrees of Freedom.

ECU Electronic Control Unit.

ERM Entity Relationship Model.

EU Execution Unit.

FOV Field of View.

FPFH Fast Point Feature Histograms.

FPGA Field-programmable Gate Array.

fps Frames per Second.

FWHM Full Width at Half Maximum.

GigE Gigabit Ethernet.

GL Gausian-Lorentzian Cross Product Function.

GPS Global Positioning System.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HSV Hue Saturation Value.

I/O Input/Output.

ICR Instantaneous Center of Rotation.

imager Image-Sensor.

IR Infra Red.

LCTF Liquid Crystal Tunable Filter.

LED Light Emitting Diode.

LIDAR Portmanteau of 'light' and 'RADAR'; used as synonym for laser range finders.

LP Line Profiling.

ML Machine Learning.

MR Measurement Range.

mRMR Minimum Redundancy, Maximum Relevancy.

MRT Magnetic Resonance Tomography.

MS Multi-Spectral.

MSB Most Significant Bit.

MVC Model-View-Controller.

MWLP Multi-wavelength laser line profile sensing.

N/C not connected.

NDVI Normalized Difference Vegetation Index.

NIR Near Infra Red.

OS Operating System.

PC Personal Computer.

PCL Point Cloud Library.

pel Pixel.

PFH Point Feature Histograms.

PMD Photonic Mixing Device.

RADAR Radio detection and ranging.

RAM Random Access Memory.

RGB Red Green Blue.

ROI Region of Interest.

ROS Robot Operating System.

RSD Radius-based Surface Descriptor.

SAD Sum of Absolute Differences.

SDM Space Division Multiplexing.

SL Structured Light.

SLERP Spherical Linear Interpolation.

SLS Structured Light 3D Scanner.

SPFH Simplified Point Feature Histograms.

SRAM Static Random Access Memory.

TDM Time Division Multiplexing.

TOF Time of Flight.

UAS University of Applied Sciences.

w. r. t. with respect to.

XML Extensible Markup Language.

Chapter 1

Introduction

Today, the use of sensors and computerized data analysis is mainstream throughout the agricultural research community. Agriculture faces huge challenges when it comes to satisfying global food demand. Human population is expected to reach 9 billion individuals by 2050. Increasing incomes and lifestyle additionally drive the global food demand. Moreover, the environmental footprint of agriculture comes more and more into focus and market forces as well as quality requirements must be met. Thereby, application of sensors and the smart interpretation of their information in cooperation with human interaction and data management is assumed to be a key-enabling technology for optimizing agricultural production and overcoming barriers in terms of increasing yield and reducing environmental impact [162].

During the last 25 years agricultural machines have been more and more equipped with sensors and have become more and more computerized. However, many agricultural applications comprise extremely challenging environmental conditions for sensor operation. Sun light exposure, dust, moisture, dirt, mechanical shocks and vibrations are only a couple of possible distortions. Additionally, ambient conditions change fast and different field situations vary a lot more compared with industrial processes. This causes high requirements to robustness of image processing systems and is the reason for most sensors used and interpreted online for actions on the machines being relatively simple not image-based sensors. Despite having a huge history of research in agricultural applications, image-based sensors - potentially outperforming not image-based ones - are only rarely used in agricultural machines in practice. In spite of its research history, online use of image-based data has so far only been seen for driver assistance systems in agricultural practice [116].

This divergence between research on and practical application of image-based sensor systems is obviously mostly due to the robustness issues. Additionally, economical aspects play a more important role for practical than for research application. Knowing these aspects, the point of flexibility comes into focus. In this regard, flexibility means the ability to tailor an image-based sensor and processing chain toward the application-specific needs and the field-specific environment, hence sensor configuration and model adaptation. Lacking these kinds of flexibility,

sensor systems will either be too expensive or not robust enough for practical agricultural uses, or both.

This dissertation focuses on the novel sensing approach *Multi-wavelength laser line profile sensing (MWLP)* [1]. This approach represents a very flexible concept for sensing range-data and spatial spectral reflectance at multiple, selectable wavelengths in an image-based manner using a single optical sensor. Though it might be of interest for other fields as well, it is particularly geared to and of interest for agricultural applications. Hence, this dissertation mainly elaborates the agricultural domain as an application field.

1.1 Problem statement and approach

This chapter is intended to give an overview of the approach pursued throughout this document. First a description of the problem is given and it is described how it is thought to be solved.

Before the problem statement and approach are elaborated Digression 1.1 on the light section or line profiling [2] method is given here. It is the basic measurement principle of the Multi-wavelength laser line profile sensing (MWLP) system. Thus, knowledge of this method is required for understanding the MWLP approach. Readers familiar with this method may feel free to skip reading the digression.

Digression 1.1: Excursus on laser line profile sensing

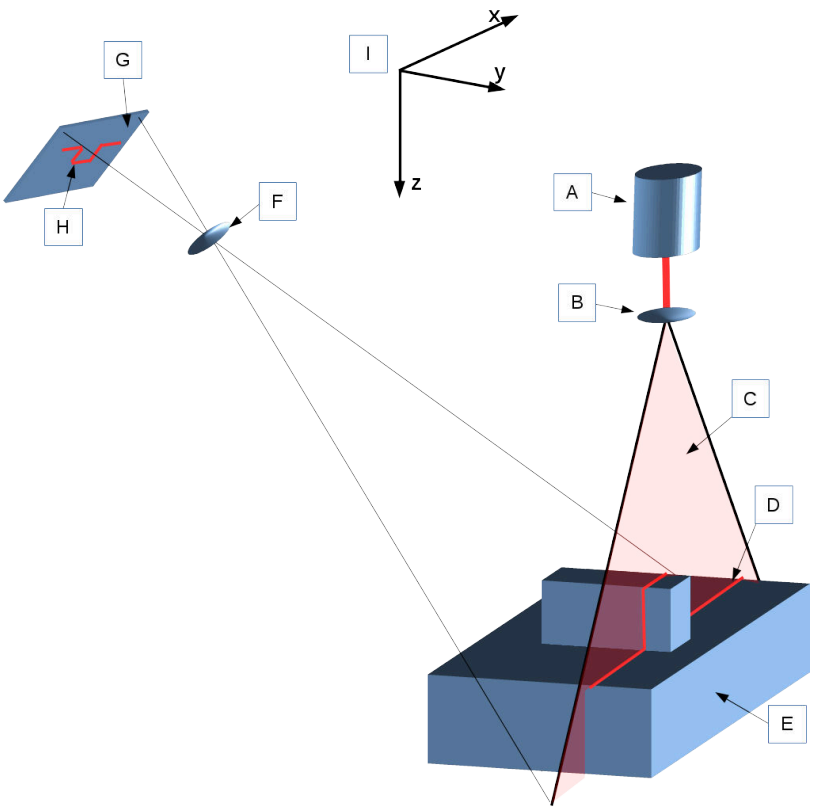
The laser line profile sensing method is a method for precise and fast distance measurements. It is widely used in industrial applications [137]. With light section microscopes known for more than 80 years laser line profiling was first used 50 years ago using the HeNe laser invented in that time. Nowadays Line Profiling (LP) is mainly conducted using semiconductor-based diode lasers [24]. In recent years LP has found its first applications in the agricultural domain, such as plant phenotyping [99] [101].

LP sensors project a laser line onto the measured object. Typically, the ray-like laser beam is expanded by a cylindrical lens to a fan-like laser beam directed to the object of interest for projecting the laser line on the object. The laser line is then monitored using a camera-like imaging system - comprising optics (lens+aperture) and a matrix-like image sensor (CCD or CMOS) - that is mounted angular with respect to the laser line. Thanks to this

[1] The terms 'Multi-wavelength line profiling' and 'Multi-wavelength laser line profile sensing' are used synonymously.

[2] The terms 'laser light section sensor' [71] [104] and 'laser line profile sensors' [73] [56] are both synonymously and frequently used for such sensors. Sometimes also the term '3D laser scanner' in applied to point the sensor type adding a description as 'triangulation-based' in order to distinguish from Time of Flight (TOF)-based laser scanners [100] [102]. Here the terms 'laser line profile sensing' or short Line Profiling (LP) will be used below whenever this method it meant.

angular mounting the laser line projection appears as height profile in the camera image [5]. Knowing the triangulation angle, it allows calculating the distance between laser and the object surface. This is shown in 1.1.



Legend:

A	Diode laser
B	Beam shaping optics
C	Fan-like laser beam
D	Laser line on the object
E	Object to be measured
F	Camera optics
G	Image sensor
H	Object profile projection
I	Coordinate system: • x: along laser line • y: direction of movement • z: object distance

Figure 1.1: Line Profiling (LP) principle. Adapted from [6]

The triangulation angle is spanned by the laser bean (z-direction) and the optical axis of the camera. Depending on the triangulation angle and the camera optics only objects within a setup-defined Measurement Range (MR) are visible. Objects closer to the laser than a minimum z-distance (clearance distance) are not viewable by the camera; the same is valid for objects exceeding a maximum z-distance. The measurement range lies between the minimum and the maximum z-distance, as can be seen in Figure 1.2. Between the minimum and the maximum z-distances object distances can be measured. The number of measurable distance steps depends on the resolution of the image sensor.

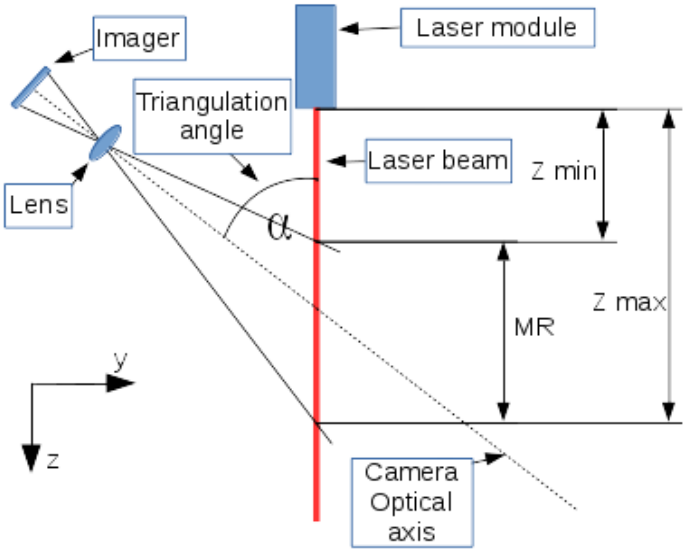


Figure 1.2: LP measurement range. Adapted from [24]

Decreasing the triangulation angle for a given camera + laser system can be done in order to increase the MR of the system. However, assuming a fixed number of pixels available on the camera imager the distance resolution decreases with a decreased triangulation angle. In the opposite case of an increased triangulation angle the measurement range decreases and the distance resolution increases. I.e., for a system with given camera + laser but variable mounting there is a trade-off between distance resolution and measurement range [24]. This is illustrated in Figure 1.3.

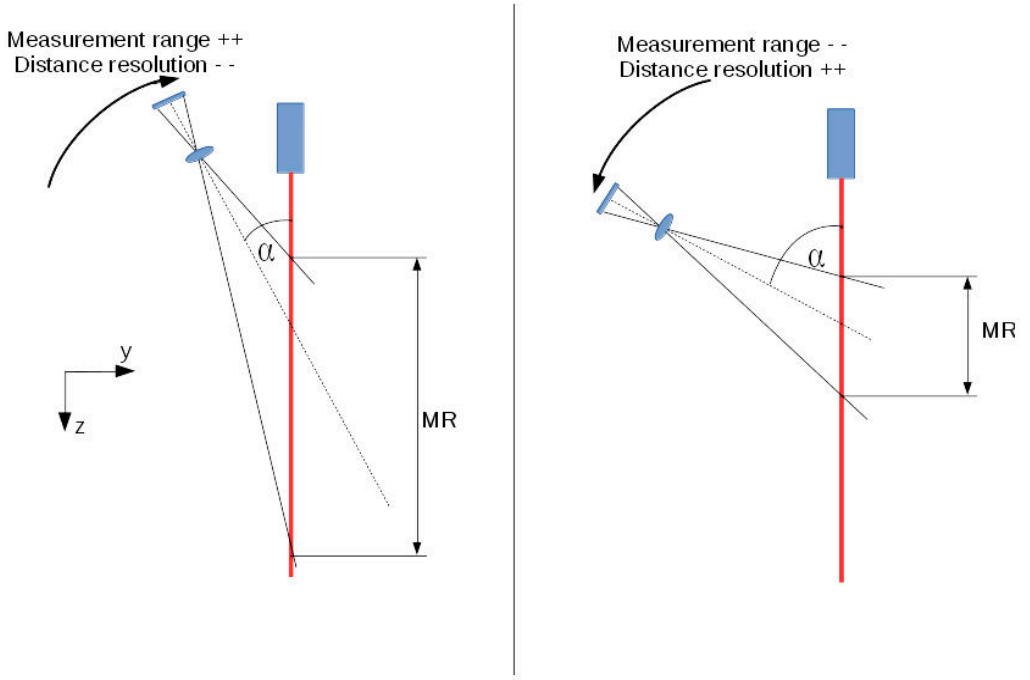


Figure 1.3: Effect of variations of the triangulation angle. Adapted from [24]

The lateral field of view of the LP system depends on the fan angle of the line laser optics and the camera optics as well as the measurement range. A near field of view and a far field of view can be distinguished (cf. Figure 1.4). Hereinafter - unless otherwise stated - Field of View (FOV) of LP systems refers to the far field of view.

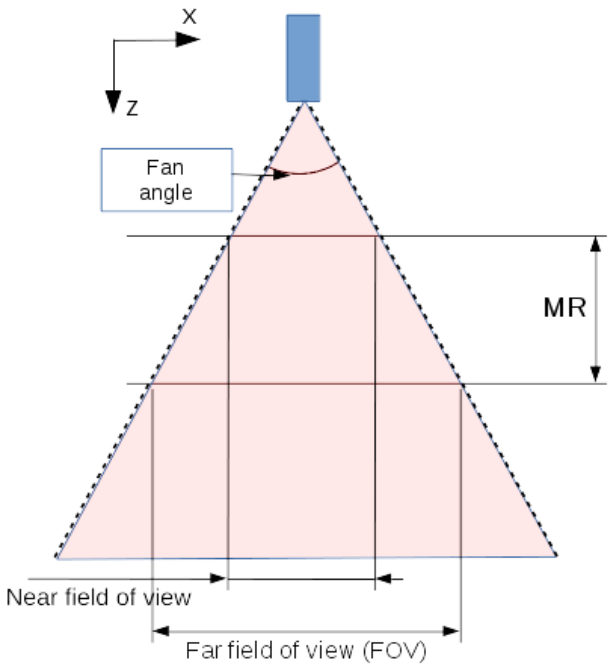


Figure 1.4: LP field of view. Adapted from [72]

Another issue to be kept in mind when applying triangulation-based distance measurement techniques is shading at sharp edges. As drafted in left-most and center part of Figure 1.5 'before' edges and in more-less plain surface regions of the object, the surface point which reflects the laser is in view of the camera imager. The distance measurement is possible here. However, 'after' sharp edges the view of the imager on the relevant surface point might be blocked by the higher part of the object, see right-most part of figure 1.5. A distance measurement is not possible here. Hence, all triangulation-based distance measurement devices create a couple of invalid, i.e., not measurable, pixels along sharp edges of the monitored objects. The amount of invalid pixels an edge of specific height causes hereby again depends on the triangulation angle.

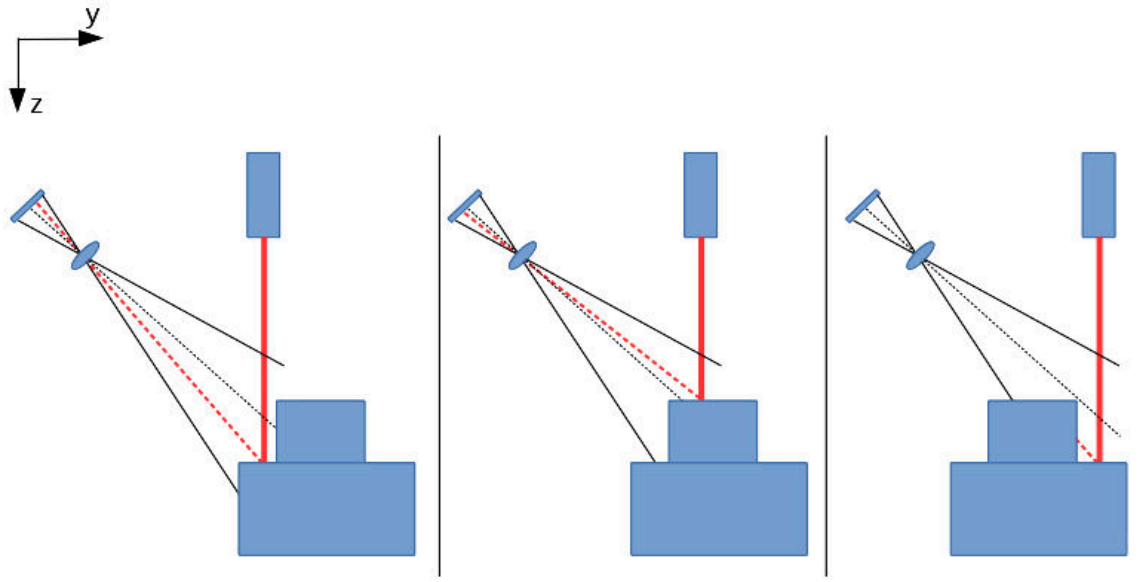


Figure 1.5: Possible missing measurements for triangulation due to shading at sharp edges.

For industrial uses, commercially available line profile sensors usually integrate line laser, camera and electronics for line detection in a single integrated housing (Figure 1.6). Examples for such integrated LP sensors are LMI Gocator 2300 series [73], Pepperl+Fuchs LR 300 series [104], Leuze LPS 36 [71] or MicroEpsilon ScanControl sensor [56].

Figure 1.6: Integrated line profile sensor Gocator 2300. Source: [73].

1.1.1 Drawbacks of multi-sensor systems in agricultural applications

As stated at the beginning of chapter 1, the practical adoption of image-based sensor systems and image processing for agricultural applications is far behind both, their potentials and the level of research conducted in these fields. Not image-based sensors are still far more common in practical use [116]. Image-based sensors potentially outperform other systems. However, they require robust data interpretation and under field conditions there are almost always distortions

weakening the signal or causing the system to fail [116]. Approaches using a single standard issue image-based industrial sensor and applying it without application-specific adjustments to an agricultural application - i.e., only doing application-specific engineering on the image processing chain - have not yet resulted success stories of online sensor use in practical agriculture.

Evidently, a system level engineering of both sensor system and processing chain is required. Moreover, application-specific adjustments of the sensor and its setup are required to provide the level of robustness required in agricultural applications. The approach of multi-sensor data fusion [86] has been identified as promising here [118]. It allows combining multiple sensors and interpreting their data in a combined manner [86]. Probabilistic methods are applied to deal with different levels of reliability of the individual sensors. This approach has become popular and is very promising for agricultural applications, such as phenotyping [18] [118] [156]. Apart from the data processing, hereby, the sensor setup can be geared toward a specific application by correct selection, placement and combination of sensors.

However, while being a straight forward solution for many applications, multi-sensor data fusion still has its drawbacks. Particularly, mounting, adjustment and calibration are tedious and error-prone tasks under varying ambient conditions and under the influence of mechanical shocks and vibrations [137]. Moreover, it still requires a certain amount of work and this effort is not directly targeted for solving the application problem [86]. Finally, the costs of a system do typically not decrease but increase with the number of sensors it comprises.

Consequently, multi-sensor combinations have a high potential for providing practical solutions for agricultural applications because they allow system level engineering of both sensor system and processing chain. However, they still have some drawbacks. I.e., if a single sensor system can be geared to, adjusted and configured in accordance with an application's requirements and thanks to those application-specific engineering delivers sufficient information to solve the problem, this will still be desirable [137].

1.1.2 The idea of Multi-wavelength laser line profile sensing (MWLP)

This is where the approach of Multi-wavelength laser line profile sensing (MWLP) comes in. It represents a novel sensing method for combined acquisition of range data along with spectral information at multiple, selectable wavelengths.

The idea of MWLP initially came up during the project 'RemoteFarming.1' [131]. The goal of this project was to develop a robotic weed control system for intra-row weed treatment in carrot (*Daucus carota*) cultivation in organic farming. For this project an image-based sensor system was required, capable of delivering high resolution spectral reflectance information in the visible red and Near Infra Red (NIR) range. These wavebands are very significant for plant/soil segmentation as basis for plant classification [155]. Additionally, the sensors had to provide high resolution range information in order to control the working depth of the weeding tool. Partly, the range data was further thought as additional input for improving plant classification. An extensive search for commercially available sensors supplying all kinds of required information

did not have results. Therefore, a multi-sensor approach was pursued, including a system of multiple mono cameras and flashed lighting at multiple wavelengths [134]. However, during this the concept of MWLP was born. The MWLP system is able to deliver all required information for this application from a single sensor system.

The MWLP concept expands on the method of triangulation-based laser line profile sensing. It does not sense just a single laser but multiple line lasers at different wavelengths. Moreover, it does not only retrieve the position of the laser lines but also evaluates the amount of reflection and scattering of the individual lines at the different wavelengths. The scan values of the different lines are assembled. This allows improving the robustness of the distance measurement by comparing the measurements of different lasers. Scanning objects with the system results in overlayed distance images or point clouds, in which reflectance and scattering information at multiple, selectable wavelengths are available for each point. [137].

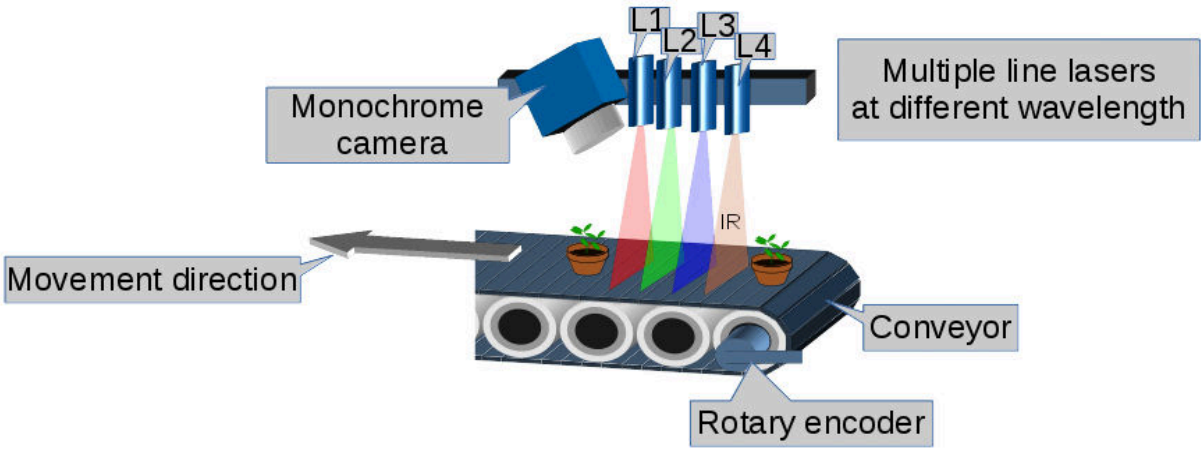


Figure 1.7: MWLP sensing principle mounted on conveyor with four line lasers.

The MWLP system consists of multiple Continous Wave (CW) line lasers. These are simultaneously captured with a single monochrome Image-Sensor (imager). A frequent use case is the application of the system mounted on top of a conveyor for measuring objects that are moved along the sensor. This constellation is sketched in Figure 1.7. However, the system can also be operated mounted on a vehicle or robot for measuring still objects with a moved sensor. As it is a line-wise scanning system, relative movement is required between measured objects and sensor, though [137].

The camera used for the MWLP system has a wide range of sensitivity in the visible and NIR range, e.g., from 400 nm to 1000 nm. This means that line lasers of any wavelength within this range can be combined freely, matching the requirements regarding spectral selectivity of the specific application. The prototype has shown to provide good results with 3 line lasers. However, more than 3 line lasers can be used with it [137].

The mounting of the line lasers is vertically with respect to the conveyor, i.e., direction of

movement (see figure 1.7). The advantage of this mounting is that the distance which the conveyor has to move for a point highlighted by L1 to become highlighted by L3 does not depend on the surface height of the sensed object at this point. I.e., the object height must not be taken into account for the line assembly. This is only possible for vertical mounting [137]. The scans of all wavelengths are assembled based on the monitoring of the conveyor movement by a rotary encoder. Thereby, the data of the rotary encoder can be improved by optical tracking of the movement which in turn improves the accurateness of the line assembly.

- Space Division Multiplexing (SDM) vs. Time Division Multiplexing (TDM) for active spectral selectivity

For devices generating spectrally selective features from objects in an active manner, i.e., using special light sources [3], the selectivity can be provided using SDM or TDM. Time Division Multiplexing (TDM) thereby means that the sensed object stays (almost) in the same place and is spotted by light sources of different wavelengths at different times. For Space Division Multiplexing (SDM) the object is moved and is highlighted by light sources of different wavelengths in different locations for providing different spectral views of it [4].

For the MWLP-concept TDM is also possible. In this case, the line lasers all need to be aligned with each other highlighting the same line on the sensed object. Then, for each camera shot only one line laser can be turned on and it must be turned off for the next shot where another line laser is active. Consequently, the TDM manner for creating spectral selectivity with the MWLP system requires a very good alignment and a high amount of synchronization between camera and lasers. Moreover, the scan rate of the system for providing a scan with multiple wavelengths is reduced. It is divided by the number of wavelengths to be monitored. However, for line-wise scanning systems, such as the MWLP system, the scan rate is always a critical point. Hence, the TDM-based manner was discarded early.

For SDM-based spectral selectivity with the MWLP system there are two approaches possible. Both have in common that - unlike for TDM - the line lasers can be operated in CW mode, i.e., less electronic effort for controlling and synchronizing them, and both reach the maximum scan rate, which is here only limited by the frame rate of the camera.

The first of those approaches is drafted in Figure 1.8. Here a schematic input image of the monochrome camera scanning 3 laser lines is sketched. For this approach the line lasers are mounted such that each of them appears in a dedicated Region of Interest (ROI). In each of those Region of Interest (ROI)s only one line laser may appear - detection of two laser lines in one ROI would cause a discarded pixel[5]. This makes distinguishing the lines caused by different lasers in the monochrome image relatively easy - the line appearing in a dedicated ROI is assumed to be induced by the laser the ROI is assigned to. The drawback of this approach is

[3] More information on this follows in chapter 2.2.

[4] Obviously, for SDM the same object is still spotted with the different wavelengths at different times but, unlike for TDM, for SDM the measurements for different wavelengths can be carried out for different objects simultaneously, i.e., at the same time.

[5] Same like no detectable laser line, e.g., due to shading (cf. Digression 1.1), causes an invalid pixel.

that for each line laser only a reduced ROI is available as dynamic range. I.e., for a system with, e.g., three line lasers in the trade-off between measurement range and distance resolution (cf. Digression 1.1) one of those is reduced (divided by 3). Consequently, a MWLP system following this approach cannot reach the distance resolution or measurement range of a comparable LP sensor.

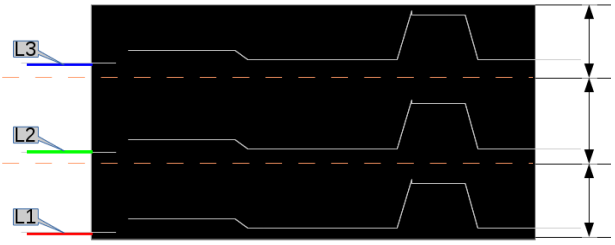


Figure 1.8: SDM approach 1: Dedicated ROI for each laser line.

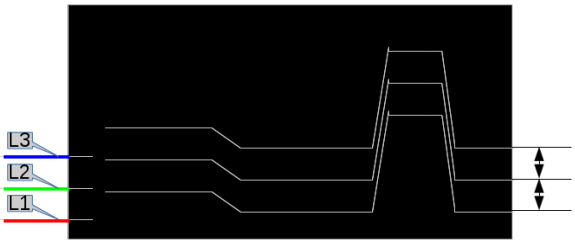


Figure 1.9: SDM approach 2: Defined distances between the laser lines.

The second possible SDM approach is having the line lasers be mounted such that they appear very closed to each other in the image and calibration of their (small) defined distances for distinguishing the lines caused by different lasers in the monochrome image. This would reduce the tension-field in the trade-off between measurement range and distance resolution as each laser line could appear nearly anywhere in the camera image. The approach is shown in figure 1.9. However, for this approach distinguishing the lines and assigning them to the respective lasers is very complex. Even knowing the metric distance between the lasers, the pixel distance in the image also depends on the height of the sensed object. Moreover, for complex geometries laser lines may be shaded or even overtake each other. Hence, a full-fledged tracking of each pixel of all lines with context over multiple images would be unavoidable. Likely, for outdoor use this would frequently be subject to failure. Consequently, the SDM approach 1 was chosen for the MWLP system and applied throughout all work described in this dissertation.

1.2 Objectives and concepts

After section 1.1 has given an initial and crude overview about the MWLP system and its application context this section will point out the targets pursued while working on this dissertation. Further, the structure of the following chapters will be explained and the scientific contributions of this work are itemized.

1.2.1 Goals of this work

The overall target of this work was to show the feasibly of MWLP and potentials of its use for agricultural applications. This implies the structure of this work was sensor-centric. However, as discussed before, for agricultural solutions a system level engineering of the combination of sensor system and processing chain is required. Therefore, a classification of the sensor output data from the MWLP prototype system should be performed for some example applications to

show that the sensor data cannot just be visualized nicely but can also serve as descriptive input for real world classification problems. Hence, the following goals were identified.

Realization of the MWLP prototype: At the very beginning the MWLP concept was drafted. It had to be translated into a hardware design (camera, optics, lasers, processing system) and the software and processing pipeline for image acquisition, calibration, tracking, line detection and assembly had to be designed, implemented and tested.

Characterization for the MWLP prototype: After the prototype was set up it had to be characterized in order to validate its functions and rate its performance in comparison with other state-of-the-art sensor systems.

Classification of MWLP sensor data: As stated, it was intended to show that the outcome of the MWLP system provides descriptive input for agricultural classification problems, i.e., automatic decisions can safely be taken based on this data. Since 'there is no such thing like a free lunch', this cannot be done in a generalized approach but only for example applications. These applications had to be selected and the classification infrastructure and sensor had to be linked and geared toward these applications. Further, system level tests and validation had to be performed for different agricultural application.

1.2.2 Structure of the dissertation

Here, an overview of the contents of each chapter of this dissertation is given:

Chapter 1 gives an overview of the problem and the MWLP approach. Further, goals, structure and contributions of this work are itemized.

Chapter 2 elaborates on the state-of-the-art for range imaging and spectrally selective imaging techniques followed by a brief overview of scatterometry in the agricultural field. A comparison of the MWLP concept and other possible sensing concepts for combined acquisition of range and spectral data outlines differences, advantages and disadvantages of the MWLP approach within the context of related work.

Chapter 3 describes in detail how the MWLP concept was realized and translated into a prototype system with respect to hardware, software, calibration and visualization.

Chapter 4 is entitled to the validation experiments with the MWLP prototype and a quantified comparison for its performance with other state-of-the-art sensors.

Chapter 5 shows the classification of MWLP data for different agricultural applications. The selected example applications are classification of potatoes and stones/soil clods conveyed with them and classification of crop and weed plants. The classification takes place in a pixel-based manner allowing flexible adjustment of the processing chain with In-Field-Labeling.

Chapter 6 summarizes on the previous chapters and gives an outlook on the following work.

1.2.3 Scientific contributions

This section is intended to itemize the scientific contributions of the work described herein. Main contribution is the MWLP system, others are within its uses and context.

MWLP concept and prototype: The MWLP approach was invented, drafted, designed and realized as part of the work on this thesis. This is a novel sensing approach of combined acquisition of image-based range and spectral reflectance data. Such a triangulation-based, scanning system, that combines active ranging with active spectrally selective imaging from laser lines at multiple wavelengths with a single monochrome imager, had - to the best of my knowledge - not been covered by other researchers before. It assembles the lines based on optical tracking and applies differential imaging before line detection. System concept and realization were initially published at 'SPIE Photonics Europe' in 2014 [137].

Image-based scattering evaluation at multiple wavelengths: Besides combined image-based range and spectral reflectance data the image-based evaluation of laser light backscattering is another interesting feature of the MWLP system. As will be mentioned in section 2.3, laser light backscattering is an interesting effect for agricultural applications and there has been work on its use applying point lasers [76] [91]. Moreover, image-based evaluation of line laser backscattering was performed before [126]. However, the combined capturing at multiple wavelengths and pixel-wise assembly along with 3D information in a single sensor is also a novel aspect. This was primarily elaborated by the publication for the workshop 'Computerbildanalyse in der Landwirtschaft' (CBA) in 2015 [138].

Classification of MWLP sensor data for agricultural application: As the sensing approach is novel its use for the applications mentioned in section 1.2.1 could be assumed new, while most of the classification techniques used in this work are known and commonly used techniques. A special aspect hereby is the pixel-based classification approach, which differs from the usually object-based concepts that can be found in the literature for above mentioned applications in vast numbers. The pixel-based concept has obviously many limitations but it provides high flexibility allowing model adaptation by In-Field-Labeling. The **In-Field-Labeling** concept was published at the conference held by the 'Gesellschaft für informatik in der Land- Forst- und Ernährungswirtschaft' (GIL) in 2015 [139]. The pixel-based classification with In-Field-Labeling requires data storage and management of image and label data as well as linking them at pixel level. The **concept for data management** of such image and label data together with metadata was published at the 'European Conference on Precision Agriculture' (ECPA) in 2013 [135].

Chapter 2

State-of-the-art in 3D and spectrally selective imaging

Systems for combined acquisition of image-based range data and spectrally selective image data are a vast topic. This chapter will attempt to give a brief overview of techniques for range imaging, spectral imaging and scatterometry in order to classify the MWLP concept within the context of related scientific work. The last section will compare system concepts for combined acquisition of range data and spectrally selective intensity data to outline the main advantages and drawbacks of the MWLP concept. The application examples of the techniques given in this chapter will - same as this entire dissertation - mainly focus on the agricultural branch.

2.1 Range imaging techniques

Starting from origins in geodesy using triangulation-based principles, during the last 25 years a whole variety of techniques for acquisition of 3D information, i.e., range measurement, have been developed. Some of those techniques still use the triangulation principle, others rely on different physical principles [38, p. 19]. For relating the MWLP concept within the context of principles for distance measurement a classification of those techniques has to be done. It is given by Figure 2.1 showing a brief classification of ranging methods based on Gockel [38, p. 20].

Depending on the physical domain, ranging techniques can be grouped into radiological, optical, acoustical and mechanical ones.

Mechanical methods are 'not contactless'. Such devices touch/feel the object surface or mill the object of interest layer-by-layer[38, p. 20]. These methods typically affect, harm or even destroy the object. Hence, these are not in focus here. Radiological methods, such as Computer Tomography (CT) or Magnetic Resonance Tomography (MRT), have a broad adoption for medical imaging applications. However, they require relatively huge and expensive measurement equipment. Therefore, they are barely suitable for agricultural applications, particularly when it comes to outdoor applications. There has been first research on applying smaller mag-

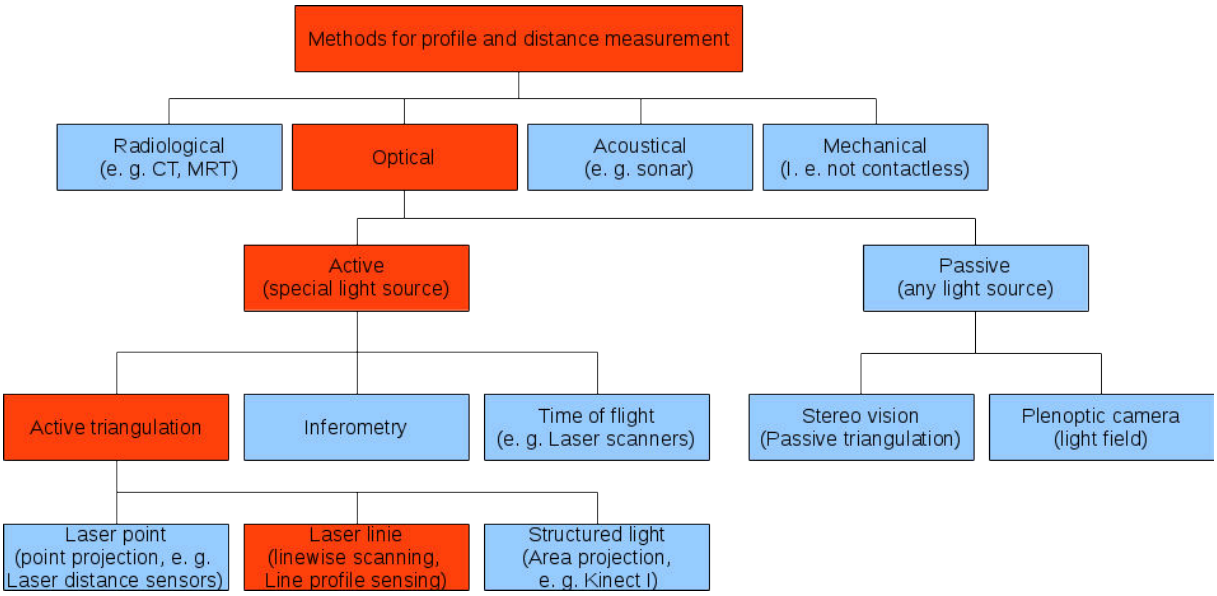


Figure 2.1: Classification of ranging techniques based on Gockel [38, p. 20].

netic resonance devices for determination of moisture on self-propelled forage choppers but these attempts are still in research state. Moreover, these smaller devices are not image-based [65].

Acoustical methods measure the distance by emitting a sound signal - usually ultrasonic - and capture the echo reflected by the object of interest. Knowing the traveling speed of the signal, i.e., speed of sound, the distance of the object can be calculated from the time elapsed between emission and reception of the signal. Variations of the speed of sound due to medium or temperature might be corrected knowing these ambient parameters. The robust and small ultrasonic sensors have found a number of applications in agriculture, such as assessment of canopy biomass [111], height control of sprayer booms [94] or estimation of grass amount[32]. For phenotyping multi-echo sensors are of particular interest because they allow measuring individual leaf levels [79]. However, the so far adopted acoustical systems are not image-based and therefore not in focus of this work.

The optical methods for distance measurement can be grouped into ‘Active’ and ‘Passive’ ones. Active methods in this context require a special light source while passive methods can be used with any light source providing sufficient light for the sensor. For passive systems even the use without a dedicated light source is possible, if enough ambient light is given. Examples for such passive optical devices are stereo vision systems or light field cameras. While having the advantage that no special lighting is needed, the major drawback of these techniques is that they rely on well-structured surfaces for creating dense depth information [137, p. 3]. The correspondence problem has to be solved for image segments captured of the same object surface region from different perspectives [151]. This is not possible if the object of interest has entire regions on its surface without significant textures, i.e., same color, constant reflection. The

correspondence problem can not be solved for such objects. Passive systems fail to create depth maps of such objects [63]. In any case, passive devices cannot provide a depth map with the native camera resolution as for each correspondence some pixels have to be grouped to describe the corresponding points.

The active optical methods mainly consist of TOF-based systems and active triangulation systems. Interferometric methods are very rare [38, p. 23]. TOF-based system measure the time an emitted light signal needs to travel to the object of interest and back, thereby calculating distance using known speed of light. There are scanners having a single beam and provide image-based information using a mirror that is rotated around one or two axes. Further, TOF cameras use e.g. Photonic Mixing Device (PMD) sensors and can create a matrix-like depth image by one shot projection [63]. Both kinds of devices find applications in the agricultural domain, e.g. navigation of vehicles [152] or robots [4] as well as phenotyping [62]. Unlike active triangulation systems with a limited Measurement Range (MR), TOF-based devices can measure quite far distances up to e.g. 80 m. On the other hand active triangulation systems usually outperform TOF-based devices in terms of distance resolution.

The active triangulation methods can be grouped depending whether they scan a single point, a laser line or provide a whole area scan with one shot. Simple laser distance sensors are examples for point projection. There are applications of such sensors in the agricultural domain, such as for phenotyping [18]. Line-wise scanning and active laser triangulation is the basic principle of the laser line profile sensing or light section method. This method is in focus of this work as it is the distance measurement principle of the MWLP concept. Line profiling is a robust and widely used technique in industrial applications for fast and precise 3D measurements [137]. Recently, it has come into the focus of the agricultural research community as well, e.g. for phenotyping applications [99][101][52]. For those line-wise scanning devices a relative movement between the object and the sensor and assembly of multiple scans is required for creating image-based information. This is not necessary for devices with areal projection of structured light. These devices can create full range images by a single shot [38]. Commercial products with this principle are for instance Microsoft Kinect I or David Structured Light 3D Scanner (SLS). However, areal projection of structured light and triangulation again the resolution of a correspondence problem is required. Due to the active illumination weakly textured object surfaces are not problematic for these devices - unlike for stereo systems. However, it still has to be determined which part of the recorded image belongs to a particular point in the projected pattern. Therefore, pixels have to be grouped to describe the pattern point. Consequently, the native camera resolution cannot be reached for the distance map by these devices. Hence, line profile sensors typically reach a slightly higher distance resolution then structured light devices.

2.2 Spectrally selective imaging techniques

Same as the range imaging techniques in the previous section, the methods for deriving spectrally selective reflection data from monitored objects are classified here in order to relate the MWLP approach within the context of related work. Obviously, for measuring light reflection in different

wavebands / for different wavelengths only optical techniques are possible. These are shown in Figure 2.2.

Note that, spectral selectivity is hereby considered in a broader sense. Obviously, the spectral features of a hyperspectral imaging system go far beyond the capabilities of a standard issue color camera. However, both systems measure in the same physical domain even though their spectral resolutions differ by magnitudes.

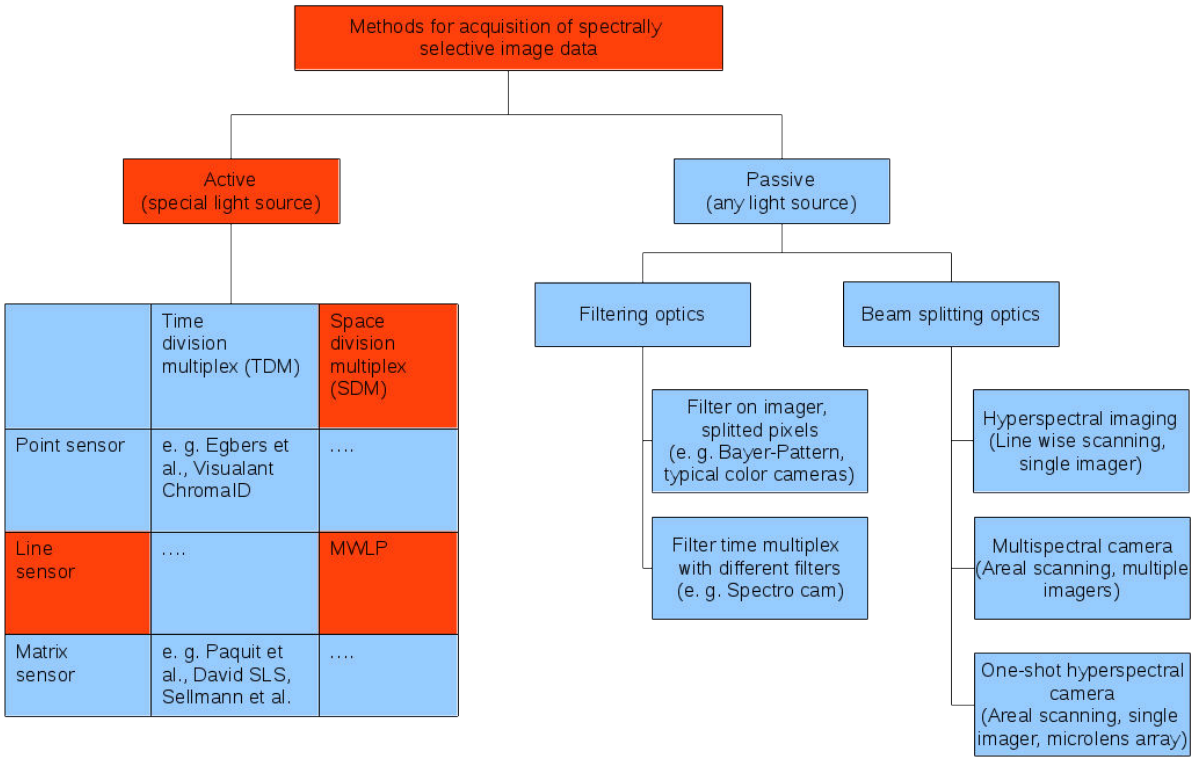


Figure 2.2: Classification of spectrally selective techniques.

Like techniques for optical range measurement, spectrally selective methods can also be grouped into 'Active' and 'Passive' ones, where the active require a 'special' light source and the passive ones can be used with an arbitrary light source or even ambient lighting, if sufficient. For passive methods the spectral selectivity is created by beam-splitting optics and/or filters. The light is spitted/filtered while traveling from the object to the imager, i.e., the selectivity is achieved on the backward path. For active methods the spectral selectivity is provided by a special light source that applies only light with the wavelength(s) of interest onto the monitored object, i.e., the selectivity is achieved on the forward path. In this case, background illumination must be neglected, subtracted or shaded out[137].

- **Passive techniques**

Unlike for optical ranging - where active and passive methods are adopted more or less in the same order of magnitude - for acquisition of spectrally selective data passive methods are far more common [137].

  * Pixel-filter on imager

The most common method for acquisition of spectrally selective image data is using an image sensor with a pixel filter applied directly on the imager. The most common filter is a Bayer-Pattern, which is drafted in Figure 2.3.

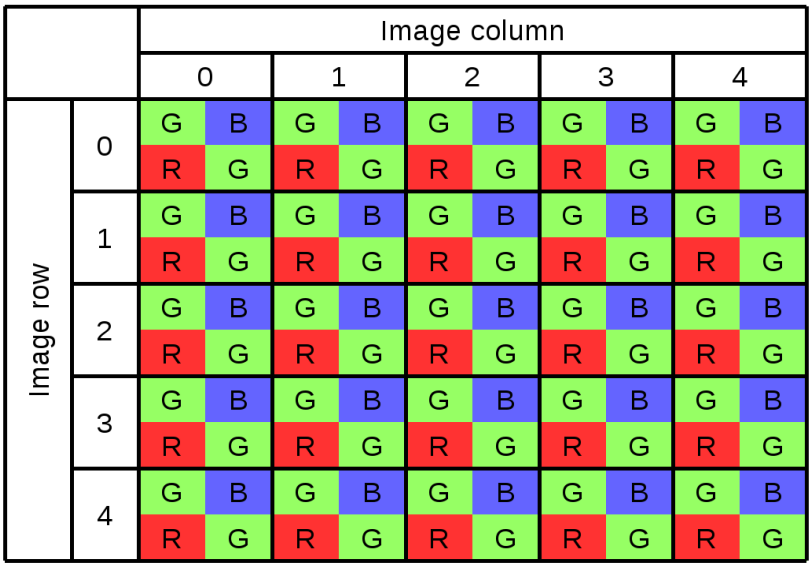


Figure 2.3: Scheme of a Bayer-Pattern.

The concept of a Bayer-Pattern is to have each pixel of the imager subdivided into four parts. Two of these parts have green filters passing only light in the green waveband to those parts of the imager. The other two parts have red and blue filters passing only red and blue light, respectively. Thereby, for each pixel of the imager the different color wavebands are sensed by different pixel regions. Hence, for each color channel selective values are available. The over-representation of the green waveband is intended as the human eye is also more sensitive to light in the green waveband [7].

As said, the Bayer-Pattern is the most broadly adopted technique. However, the method of having a pixel filter applied directly on the imager and dividing the pixel into sub regions is not limited to this configuration. The number of regions and filters per pixel can also be increased creating a single shot, single imager hyperspectral imaging device, such as e.g., provided by IMEC [140][54]. The advantage in both cases is the very low cost per produced unit. However, the main drawback for both is the low flexibility. In order to set up an application-specific

wavelength configuration high initial cost for semiconductor process technology is required, such that application-specific adaptations are only possible for very high numbers of sold units. Another drawback of this technique is the reduced light efficiency due to the filter. E.g. for a Bayer-Pattern only one fourth of the imager is sensitive to red light. Hence, three fourth of the red light applied to the imager are lost.

* Time multiplex filtering with different or tunable filters

Another concept for passive acquisition using optical filters is picking the wavebands of interest in a time division multiplexed manner using different filters in the optical channel at different times. An example of a product applying this concept is the PixelTeq SpectroCam [107]. It consists of a rotating tray with different filters that move in front of the imager at different times. The device is shown in Figure 2.4. Another example of this concept is the setup of a standard issue monochrome camera with a tunable filter, such as CRI VariSpec. This filter product is a Liquid Crystal Tunable Filter (LCTF). The transmitted wavelength bands can be quickly electronically switched. A photo of the setup is depicted in Figure 2.5.

Figure 2.4: Spectro Cam. Source: [107].

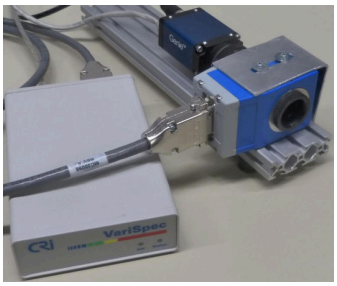


Figure 2.5: Setup of tunable filter (CRI VariSpec [105]) with camera (Dalsa Genie [20]).

The time multiplex filtering concept gains much flexibility as compared with the concept of filters on the imager, particularly as the filters in the rotating tray might be exchangeable in order to address application-specific needs. Using a tunable filter wavelengths of interest can even be configured in software. The drawback of the approach is that for collecting data from moving objects the time multiplex is often inappropriate as the images from different wavebands collected at different times do not necessarily match.

* Hyperspectral imaging

Besides passive acquisition using optical filters also beam-splitting optics can be used to create a passive spectrally selective sensor device. The most broadly adopted technique for this - particularly in the agricultural domain - are typical hyperspectral imaging devices. These sensors work as push broom scanner, i.e., scan the monitored objects line-by-line, and use a prism to split the entering light of all wavelengths into small waveband before it reaches the imager. This sensing principle is drafted in Figure 2.6.

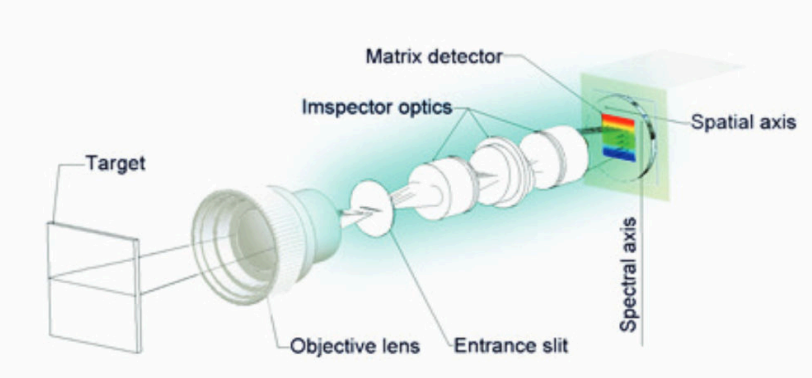


Figure 2.6: Sensing principle of hyperspectral imaging systems. Source: [147].

Hyperspectral imaging devices have found broad adoption in the agricultural research community, for instance for measuring plant moisture [147], plant biomass or other phenotyping purposes [18]. They can deliver very high wavelengths resolutions, thereby splitting the monitored wavelengths range (VIS+NIR or MIR) into very small separate wavebands (100 and more). Their main drawbacks causing a gap between adoption in research and in practice are the relatively low light efficiency requiring a lot of illumination and the high costs per unit.

* Multi-spectral camera

Another concept for passive spectrally selective capturing is the use of a beam-splitter and multiple imagers, where each imager represents a single waveband in the created multi-spectral image [70]. This principle is used by common multi-spectral cameras, such as Quest Condor5 with five imagers [57]. The concept of a beam-splitter is shown in Figure 2.7.

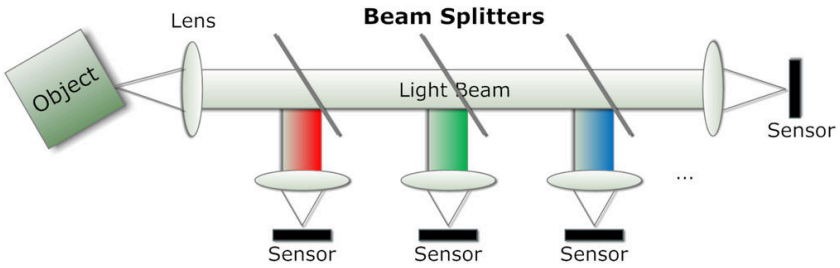


Figure 2.7: Schematic drawing of a beam-splitter with four imagers. Source: [70].

A drawback of the approach of multi-spectral cameras is that the imagers and optics must be matched during manufacturing at sub-pixel tolerance, i.e., manufacturing tolerances during camera assembly are very small. While this is a controllable problem during manufacturing, it becomes even more serious for outdoor use in environments with high temperatures. Thermal movement of the mounting of the different image sensors may cause mismatched parts in the different images.

It is also possible to combine a beam-splitter with one imager having a Bayer-Pattern, such that, e.g., only two rather than four imagers are required for creating an image with RGB channels and one NIR channel. Bispectral or multi-spectral cameras have been used in for agricultural purposes, particularly in the field of crop/weed discrimination [45] [155].

- One-shot hyperspectral camera

A further passive technique one-shot hyperspectral cameras providing 100 and more wavelength channels for a matrix-like image with a single shot and using a single imager are available, e.g. offered by Cubert GmbH [59] [58]. For these cameras different light-splitting techniques are available, such as Wollaston beam-splitting polarizers mapping the different wavelength channels to different regions of the imager or micro-lens array splitting the different channels onto a set of adjacent sensor pixels [42].

- **Active techniques**

As mentioned, for spectrally selective imaging passive techniques are far more common than active ones. Despite Nathan et al. point out that many passive techniques (beam-splitters, filters) could be used to provide defined spectrally encoded lighting [42], examples for practical realization of active techniques are rare.

They include some point sensors, such as implemented by Engbers et al. [23] or Visualant's ChromaID [80] [137]. These sensors use a time division multiplex approach and sequentially apply light of the wavelengths to be analyzed on the object using different Light Emitting Diode (LED)s. The reflected signal is measured using a photo sensor [80] [23].

For image-based active techniques with a matrix sensor concept there are also only very few examples. Same as the mentioned point sensor examples, these sensors capture the different wavelengths in a time division multiplex manner. Paquit et al. use a monochromator to pick the wavelengths of interest [96]. Bangert et al. apply flashed LED lighting in combination with simultaneously triggered monochrome cameras for obtaining spectral selectivity [4] [134]. Another example for such approach are David SLS systems. These devices can obtain 3D measures and RGB color information using active illumination provided by a video projector. The video projector first applies various structured light patters on the object for 3D data acquisition. After that it makes three shots, one with only red lighting, another with only green and a third with blue lighting, thereby obtaining the RGB color information in an active manner.

The MWLP concept can also be found amongst the active techniques for spectrally selective acquisition. As said, passive techniques are far more common. This is mainly because they are cheaper for applications promising high numbers of sold units. However, they have a drawback in terms of flexibility. The monitored wavelengths for passive systems are fixed by either the filter-pattern applied on the imager and/or optics for splitting the light beam into different wavelength bands. I.e., an adjustment of these systems for an application-specific wavelength configuration is only economically feasible, if the respective application will have high numbers

of units sold. In contrast, for active methods the wavelength configuration can be set by selecting the light sources, which can be done even for quite small number of items sold. Hence, both concepts legitimately coexist with focus on different applications.

2.3 Image-based scatterometry

Besides combined acquisition of range image data and spectrally selective reflection data the MWLP concept offers the possibility of monitoring laser light backscattering at different wavelengths in an image-based manner. This is of particular importance for many agricultural applications.

Laser light in the wavelength range between 600 and 1000 nm tends to scatter when being applied to organic tissue with high water content [64]. This can cause problems when using industrial line profiling systems for scanning plants. Such industrial sensors typically use red line lasers around 650 nm. Laser light applied on the epidermal layer is in parts reflected, absorbed, transmitted and scattered back beneath the plant surface. This is critical if the measured signal is caused by light scattered back and not by the light reflected, as Paulus et al. point out [101]. However, this effect is of high relevance for several applications. For instance, in the agricultural field Lorente et al. and Noh et al. have used the backscattering effect for determination of fruit quality using point lasers [75] [91]. Figure 2.8 and 2.9 illustrate their approaches.

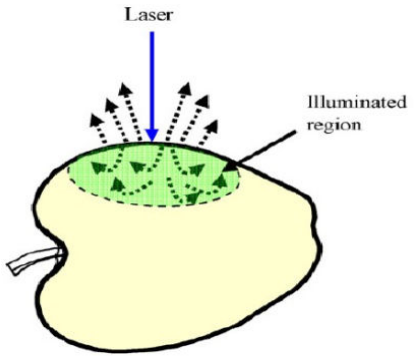


Figure 2.8: Schematic draw of scattering of a point laser by an apple. Source: [91].

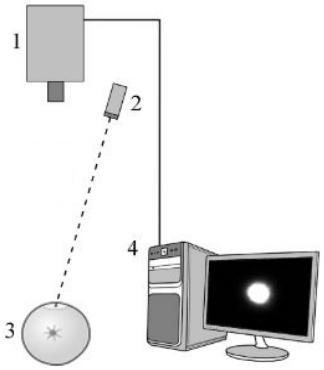


Figure 2.9: Measurement device or measuring backscattering of a point laser by an orange. Source: [76].

Further, there are systems and research projects using line lasers to obtain image-based backscattering data at a single wavelength. E.g. Sakata et al. apply evaluation of scattering for detection of latent flaws in polished glass [126]. However, the MWLP approach here provides new features for examination of fruits and crops because it allows monitoring and rechecking image-based scattering data at multiple wavelengths.

2.4 Comparison of system concepts for combined acquisition

As stated, combined capturing of range data and spectral reflectance data is a vast topic. There is a broad variety of sensor concepts, each with specific advantages and disadvantages. This chapter intents to give a brief overview in order to point out the specific advantages and disadvantages of the MWLP concept in comparison with other concepts.

In Table 2.1 an incomplete list of possible or common sensor concepts is given. The concepts itemized in Table 2.1 are:

A **The MWLP concept**
The MWLP concept is described in this work. It scans multiple laser lines with a single imager and extracts intensity and scattering information. A similar setup for obtaining line laser reflectance at multiple wavelengths was recently installed by Dupuis et al. However, they use two industrial LP sensors scanning in two measurement passes [22]. In contrast, the MWLP system applies only a single sensor and, thus, scans all wavelengths in a single measurement pass.

B **Stereo cameras with flashed lighting at multiple wavelengths**
Sellmann et al. use flashed LED lighting at multiple wavelengths and synchronously triggered stereo cameras to provide high contrast images of plants [131] [134].

C **Color stereo cameras with Bayer-pattern**
The concept of standard issue color stereo vision cameras.

D **Splitted imager with one part for laser line profiling and another part for a color line sensor**
The concept of a splitted imager for ranging and color capturing is implemented by the Sick ColorRangerE [3].

E **Combination of hyperspectral imaging and a line profile sensor**
Such a multi-sensor system delivers very high detail for both, high resolution range data and high resolution spectral data. This system setup is, e.g., used for phenotyping by Behmann et al. [8].

F **Color camera + structured IR laser light scanning**
This concept with a color camera and an additional ranging system consisting of a mono camera and a structured light laser is the sensing principle of Microsoft's Kinect I devices.

G **Plenoptic cameras**
Plenoptic cameras or lightfield cameras capture the light reflected from a point in different optical paths through a microlens array and thereby can estimate the distance of objects based on finding correspondences of the different paths. This principle is, e.g., used by Raytrix devices [37].

H **Multi-spectral camera combined with a time of flight camera**
A combination of two commonly used matrix sensors for both purposes.

I **Structured light video projector with a mono camera and TDM color data acquisition**
This principle is used by David3D SLS-2 structured light scanners [36].

J **Triangulation with a color laser and a beam-splitter**
Blais et al. use triangulation with a color laser to measure the shape of artworks. Their device also involves a beam-splitting prism to separate the different wave parts of the laser beam on the backward path for obtaining color information [12]. The principle of their device is given in Figure 2.10.

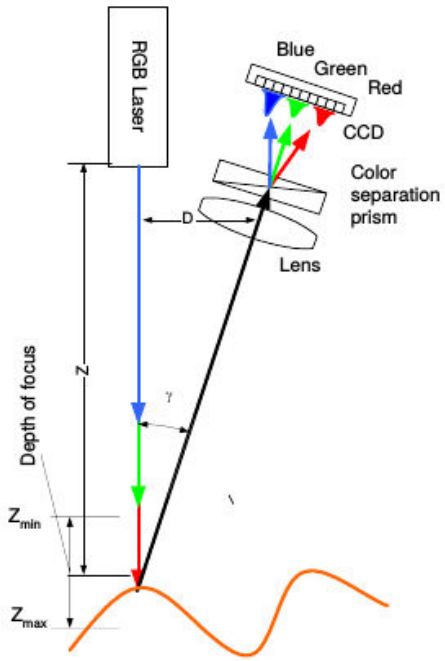


Figure 2.10: Schematic principle the color triangulation device by Blais et al. Source: [12] .

K **Multi-wavelength LIDAR**
Wei et al. have set up a TOF-based LIDAR sensor. It can operate with four different wavelengths in order to obtain 3D spectral information when scanning plant canopy [154].

The mentioned system concepts are classified in Table 2.1 depending on the following criteria:

1. **Scan principle**

   - Point sensing
     One shot of the sensor system creates a single point measurement (image pixel). In order to create image-based data rotating mirrors and/or a relative movement between device and monitored object along one (with mirror) or two axes is required.

- Line sensing
  The device must be used as push-broom scanner. One shot of the sensor system creates a single line measurement (image column or row). In order to create image-based data relative movement between device and monitored object one axis is required.
- Matrix sensing
  One shot of the sensor system creates a 2D matrix measurement (image). In order to create image-based data relative movement between device and monitored object is not needed. If a relative movement is given, sequentially captured images need to be stitched before processing in order to avoid overlapping image parts.

2. **System design**

   - 1-imager systems
     These systems comprise only a single image sensor collecting all the information required. Here, main advantage is the lack of calibration of multiple imagers or sensors with each other. Moreover, thermal movements between different image sensors are not an issue.
   - 2-imager systems
     Systems are classified as 2-imager systems if they comprise two identical image sensors. Obviously, some calibration of the sensors with each other is required for these. However, the effort for this still significantly less than for calibration of different sensor types.
   - Multi-sensor systems
     Multi-sensor systems consist of multiple sensors with different sensing principles that need to be calibrated with each other.

3. **Range imaging**
   The optical distance measurement principles are classified into active and passive ones according to conventions described in Section 2.1.

4. **Spectrally selective imaging**
   The spectrally selective measurement principles are classified into active and passive ones according to conventions described in Section 2.2.

Having in mind these classifications, the sensor concepts are rated under a number of different aspects in Table 2.1. It has to be stated that here only the general sensor concepts are rated, not the illustrating example sensors given for some of them. Moreover, the ratings are intended to be qualitative ones where '+ +' is typically better than 'o' and 'o' is typically better than '- -'. However, this qualitative rating of the sensor concept does not imply that it would be impossible for a specific sensor to outperform another sensor under an aspect in quantitative terms, if the ratings of their concepts were contrary. The rating just states that this would be unusual. The aspects considered for the ratings in Table 2.1 are described as follows:

- General ratings

5. **Robustness with respect to disturbances by uncontrolled ambient light**
   Typically, passive systems are relatively robust against distortions by uncontrolled light, e.g., sun light exposure, whereas active triangulation systems have problems here. Sensor-specific features may help here but generally uncontrolled light is critical point for active triangulation. TOF-based are more robust w. r. t. this.
6. **Light efficiency**
   Typically, active systems have higher light efficiency, as no light is lost due to filters or beam-splitters. Moreover, only light of the wavelengths of interest is applied. This manifests in a reduced energy effort for lighting, if required. This point might be relevant on carrier systems with limited power supply [137].
7. **Effort for calibration of multiple sensors with each other**
   The effort for calibration of multiple sensors is for multi-sensor systems higher than for 2-imager systems. For 1-imager systems it is the lowest.
8. **Effort for calibration of sensor and light source(s)**
   For passive systems a calibration of light source and sensor is not needed whereas active systems might require some calibration here.

- Ratings for the ranging capabilities

9. **Ranging objects without textures**
   Passive range finding techniques require well-structured object surfaces. For object surfaces without textures they fail to create depth maps, as mentioned in Section 2.1.
10. **Reduced depth map resolution due to matching correspondences**
    Ranging techniques relying on finding correspondences can only deliver range maps of resolutions reduced with respect to the native camera resolution. This is the case for passive ranging and active triangulation of structured light patterns.
11. **Maximum distance resolution**
    Though the distance resolution, i.e., minimum quantization step of the provided distance values, mainly depends on the implementation of the specific sensor, for some sensing principles the achievable resolutions are generally higher than for others. In particular, line profiling systems can measure very tiny distance steps - some even in the micrometer scale - thereby delivering very high distance resolution.
12. **Maximum measurement range**
    The measurement range, i.e., difference between the minimum and the maximum of measurable distances, also very much depends on the specific sensors but there are some constraints by the sensing principle. Typically, TOF-based systems outperform all others under this aspect.

- Ratings for the spectral selective features

13. **Maximum resolution**
    For systems applying a color imager the wavelength resolution is fixed to the three RGB wavebands. The stereo system with TDM lighting (B in Table 2.1) presented

by Sellmann et al. [131] provides two wavelengths. However, the approach might not be limited to this. The MWLP system (A) was tested with up to 4 wavelengths [137], same is valid for the multi-wavelength LIDAR (K) [154]. Multi-spectral cameras (H) typically provide data of four or five wavebands. Again, these three approaches are not limited to this number of wavelengths but for all these approaches increasing the number of wavelengths beyond - say - ten wavelengths would increase the system complexity beyond what is feasible.
The highest wavelength resolution with hundreds of wavelengths and more is reachable by using hyperspectral imaging systems (E).

14. **Flexibility of the wavelength configuration**
In case number of wavelengths/bands recorded by a sensor system is low, it will be helpful if the configuration of wavelengths can anyhow be adjusted in order to match some application-specific demands. However, for passive systems, on one hand, this is not possible, as the wavelength configuration is fixed by the filter pattern or the design of the beam-splitter. For active systems, on the other hand, the wavelength configuration can be adjusted by exchanging the light source making them more flexible.

Note that there are two ratings in Table 2.1 that are in contrast to the description in the previous paragraph. One of these exceptions is the SL video projector with mono camera (I). If for this approach a standard issue video projector is used - as done by [36] - there is actually no flexibility for this because the video projector is limited to RGB, same as color cameras. However, by exchanging the video projector for a special projector or involving a monochromator (see Paquit et al. [96]) the approach would be more flexible here. As compromise, an 'o' rating is chosen here.

The other exception for this aspect in Table 2.1 is the hyperspectral imaging (E). Obviously, as a passive system with beam-splitter there is in principle no wavelength configuration adjustment possible. However, the spectral features of this device would be underrated if they were said to be equal to those of a color camera. Typically, the approach when using such systems is the other way around. As usually online processing of the entire hyperspectral data cube is not possible anyway, oftentimes, only the most relevant wavelengths of the spectral imaging system are used. These can then be selected in software by selecting the image sensor rows to readout. This configuration is then even more comfortable than for the mentioned active systems, hence the '+ +' rating.

Table 2.1: Optical sensor concepts for combined acquisition of range and spectrally selective image data

			A	B	C	D	E	F	G	H	I	J	K
			MWLP	**Stereo cam + TDM light**	**Color stereo (Bayer)**	**Splitted imager color + LP**	**Spectral imaging + LP**	**Color camera + structured light**	**Plenoptic camera**	**MS camera + TOF camera**	**SL projector + mono cam**	**Color laser + Beam-splitter**	**Multi-wavelength LIDAR**
		Example	this	[131]		[3]	[8]	Kinect I	[37]		[36]	[12]	[154]
Classification		Scan principle	line	matrix	matrix	line	line	matrix	matrix	matrix	matrix	point	point
		System design	1-imager	2-imager	2-imager	1-imager	multi-sensor	multi-sensor	1-imager	multi-sensor	1-imager	1-imager	1-imager
		Ranging	active	passive	passive	active	active	active	passive	active	active	active	active
		Spectral selectivity	active	active	passive	passive	passive	passive	passive	active	active	passive	active
Rating	General	Robust w.r.t. random ambient light	5	- -	o	+ +	-	-	+ +	+	- -	-	+
		Light efficiency	6	+ +	+	o	- -	o	-	-	+ +	+	+
		Calibration effort multiple sensors	7	+ +	o	o	- -	-	+ +	-	+ +	+ +	+ +
		Calibration effort sensor + light	8	-	o	o	o	-	+ +	o	- -	o	-
	Ranging	Ranging objs. w/o textures	9	yes	no	no	yes	yes	no	yes	yes	yes	yes
		Reduced depth map resolution	10	no	yes	no	no	yes	yes	no	yes	no	no
		Max. distance resolution	11	+ +	o	o	+ +	o	o	-	o	+ +	-
		Max. measurement range	12	- -	-	- -	- -	o	+	+ +	o	-	+ +
	Spectral	Max. spectral resolution	13	+	o	-	-	-	-	+	-	-	+
		Flexibility of wavelength config	14	+	+	- -	- -	- -	- -	- -	o	- -	+

Table 2.2 summarizes the typical boundary conditions for a successful use of the sensor concepts itemized in Table 2.1. Further, in Table 2.2 the numbers of aspects justifying the statements are given behind the each respective statement. These numbers refer to the aspect numbering in Table 2.1.

Table 2.2: Typical conditions for applications of the itemized sensor concepts

#	Concepts	Typical boundary conditions for applications
A	MWLP	• small, ideally shaded measurement space (due to aspects 5 and 12) • energy for lighting might be critical (6) • high requirements regarding distance resolution (11, 12) • moderate requirements regarding spectral resolution (13) • objects with or without textures (9, 10) • objects with relative movement (3, 6)
B	Stereo cam + TDM light	• small, ideally shaded measurement space (5, 12) • energy for lighting might be critical (6) • moderate requirements regarding distance resolution (11, 12) • moderate requirements regarding spectral resolution (13, 14) • objects with textures (9, 10) • objects with relative movement (6) or still objects (3)
C	Color stereo (Bayer)	• huge unshaded measurement space (5, 12) • energy for lighting uncritical (6) • moderate requirements regarding distance resolution (11, 12) • low (only color) requirements regarding spectral resolution (13, 14) • objects with textures (9, 10) • ideally still objects (3, 6)
D	Splitted imager color + LP	• small, ideally shaded measurement space (5, 12) • energy for lighting uncritical (6) • high requirements regarding distance resolution (11, 12) • low (only color) requirements regarding spectral resolution (13, 14) • objects with or without textures (9, 10) • objects with relative movement (3)

Continued on next page

Table 2.2 – continued from previous page

#	Concepts	Typical boundary conditions for applications
E	Spectral imaging + LP	• small, ideally shaded measurement space (5, 12) • sufficient light for spectral imaging; low light distortion of LP • energy for lighting might not critical (6) • high requirements regarding distance resolution (11, 12) • high requirements regarding spectral resolution (13) • objects with or without textures (9, 10) • objects with relative but ideally slow and highly controlled movement (3, 6)
F	Color camera + SL	• medium size measurement space (5, 12) • energy for lighting not critical (6) • moderate requirements regarding distance resolution (11, 12) • low (only color) requirements regarding spectral resolution (13, 14) • objects with or without textures (9, 10) • ideally still objects (3, 6)
G	Plenoptic camera	• huge unshaded measurement space (5, 12) • energy for lighting not critical (6) • moderate requirements regarding distance resolution (11, 12) • low (only color) requirements regarding spectral resolution (13, 14) • objects with textures (9, 10) • still objects highly preferred (3, 6)
H	MS camera + TOF camera	• very huge unshaded measurement space (5, 12) • energy for lighting not critical (6) • low requirements regarding distance resolution (11, 12) • moderate requirements regarding spectral resolution (13) • objects with or without textures (9, 10) • ideally still objects (3, 6)

Continued on next page

Table 2.2 – continued from previous page

#	Concepts	Typical boundary conditions for applications
I	SL projector + mono camera	• medium-size, ideally shaded measurement space (5, 12) • energy for lighting might be critical (6) • moderate requirements regarding distance resolution (11, 12) • moderate requirements regarding spectral resolution (13) • objects with or without textures (9, 10) • still objects (3)
J	Color laser + beam-splitter	• small, ideally shaded measurement space (due to aspects 5 and 12) • energy for lighting might be critical (6) • high requirements regarding distance resolution (11, 12) • low (only color) requirements regarding spectral resolution (13, 14) • objects with or without textures (9, 10) • objects with controlled relative movement (2, 3, 6)
K	Multi-wavelength LIDAR	• very huge unshaded measurement space (5, 12) • energy for lighting might be critical (6) • low requirements regarding distance resolution (11, 12) • moderate requirements regarding spectral resolution (13) • objects with or without textures (9, 10) • objects with controlled relative movement (2, 3, 6)

As Table 2.1 shows, all mentioned sensor concepts differ from each other in their classifications as well as the ratings of their specific advantages and disadvantages. However, as Table 2.2 states, all have certain boundary conditions of applications particularly favoring their use. Consequently, all the systems coexist with each other partly - of course - competing but also partly complementing each other. All have advantages and drawbacks. The particular advantages and disadvantages of the MWLP concept are listed as follows.

- Advantages of the MWLP concept
  - ⋆ The first advantage of the MWLP is that it allows combined acquisition of range and spectral image data using a single sensor system. Hence, this single 1-imager system can replace multiple sensor systems. As seen before, this is not a unique advantage but still it remains being an advantage [137].
  - ⋆ The second advantage is that it can replace high-cost passive spectral imaging systems in case for an application only a few specific wavelengths are of interest. The

MWLP prototype was build with lower (device) costs than a commercial hyperspectral imaging would cause. The prototype delivers scans of three wavelengths. The concept does not imply that but increasing the number beyond - say - ten wavelengths is hardly feasible, while hyperspectral imaging provides 100+ wavebands. However, the MWLP might be an economically reasonable choice, if a specific application only required a couple of wavelengths, these were known and the others, if obtained, would be discarded from the spectral data cube anyway. As the monitored wavelengths of the MWLP system can be selected by mounting a specific laser of that wavelength, it can be tailored to provide exactly those (and only those) wavelengths, which are of interest for the application [137].

⋆ Thirdly, the system has high light efficiency. This reduces the energy effort required for lighting, which might favor its use on carrier systems with limited power supply [137].

⋆ Fourth, as stated before, it allows monitoring laser light back scattering at multiple wavelengths in an image-based manner [137]. This is of particular importance for applications sensing objects with high water content and wavelengths in the range between 600 and 1000 nm. In this range laser light tends to scatter when applied to organic tissue with high water content [64]. Here, it is not only reflected on the surface but also partly entering the tissue and scattered back beneath the surface [101]. Hence, scattering analysis - with limitations - allows assessing object properties manifesting beneath the surface. This is not possible with any passive optical system [138]. Hence, this technique has a high potential.

- Disadvantages of the MWLP concept

  ⋆ The main drawback of the MWLP approach as pursued here is that high distance resolution is only possible in a relatively small measurement range and that this measurement range is even more reduced with respect to conventional LP sensors (see section 1.1.2) [137].

  ⋆ Another disadvantage is that as an active/active system it is relatively sensitive toward distortions by uncontrolled ambient lighting. However, this drawback of the concept is partly diminished for the realized MWLP system by the differential imaging technique. It will be described in Chapter 3.

Chapter 3

Realization of the MWLP system

After an introduction in Chapter 1 and the classification within the context of related work in Chapter 2 this chapter will describe the translation of the MWLP concept to a working prototype in detail.

The prototype is able to capture 3 line lasers at its maximum resolution with a scan rate of 100 Hz. The MR in this configuration is approx. 15 cm ranging between a minimum object distance of 55 cm and a maximum distance of 70 cm.

3.1 Hardware system design

This section describes the hardware design of the MWLP system. The sensing head comprises sensor and lighting. The processing and control hardware are mounted into the control cabinet.

3.1.1 Sensing head

The sensing head of the MWLP prototype comprises a monochrome camera, line laser modules at different wavelengths and LED clusters for controlled background illumination. The sensing head is shown in Figure 3.1.

The camera is a Baumer HXG20NIR [34] (Figure 3.2). At maximum resolution of 2048x1088 pel it provides up to 105 fps. The image sensor of the camera is of type CMOSIS CMV 200. It has an enhanced NIR sensitivity, thereby assuring a good spectral response in a broad waveband from 400 nm to 1000 nm. Having a Complementary Metal-Oxide Semiconductor (CMOS) imager rather than Charge Coupled Device (CCD) the cameras frame rate can be increased beyond 105 Frames per Second (fps) in partial scan modes. This can be achieved by specifying a ROI to readout from the imager [34]. However, for use in the MWLP system the ROI mode reduces MR or FOV of the system. This can be overcome using binning or subsampling modes supported by imager and camera. In binning mode the frame rate of the camera can be increased to 210 fps, in subsampling mode 420 fps are possible. These modes provide images with reduced resolution but identical FOV of the camera, i.e., MR and/or FOV of the MWLP system are not reduced. Explanations on the scan modes can be found in Table 3.1 and Figure 3.3. Other features

of the camera that are very relevant for this application are the global shutter for minimizing motion blur and the optional overlapped readout, i.e., readout of one image from the imager may happen while exposure for the next image is already active.

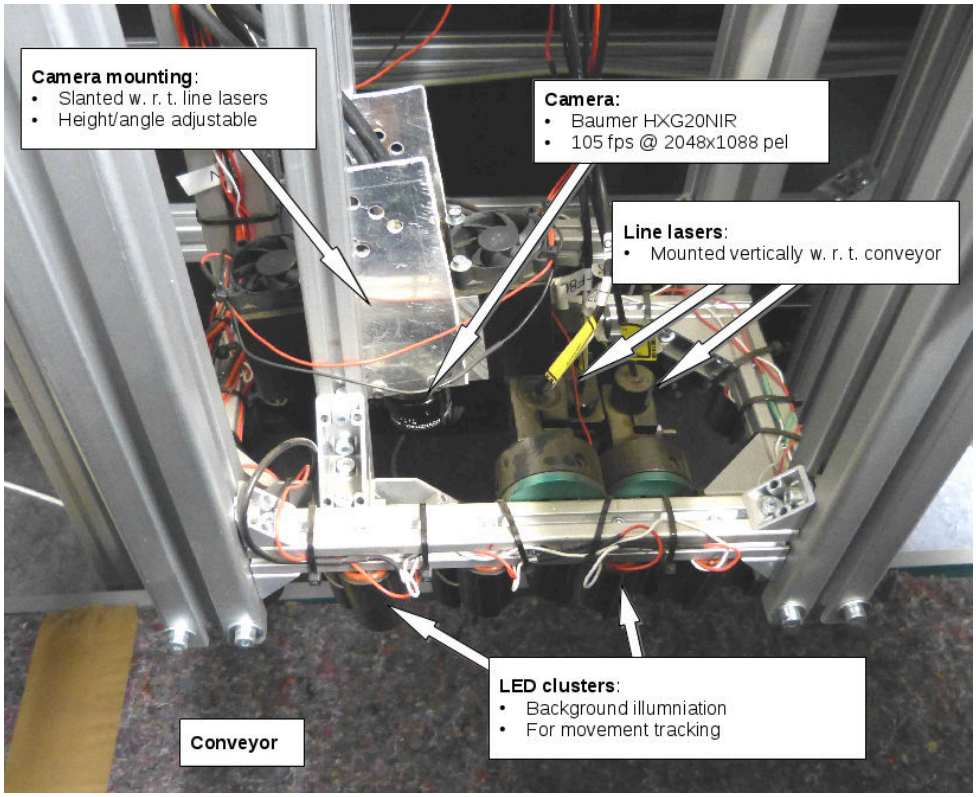


Figure 3.1: Sensing head of the MWLP prototype.

Figure 3.2: Camera of the MWLP prototype.

Table 3.1: Description of the scan modes supported by the used camera

Scan mode	Explanation	Resolution / Frame rate
Full frame	• Each native sensor pixel is readout and results in a respective pixel in the image. • See left-most drawing in Figure 3.3	2048x1088 / 105 fps
Binning 2x1	• The values of groups of each two vertically adjacent pixels are added on the imager before readout and only the sum of both pixel values is read out and results in a pixel in the outcome image. • Results in reduced resolution (higher frame rate) and higher light usage (reduced exposure time) • See center-left drawing in Figure 3.3	1024x1088 / 210 fps
Binning 1x2	• The values of groups of each two horizontally adjacent pixels are added on the imager before readout and only the sum of both pixel values is read out and results in a pixel in the outcome image. • Results in reduced resolution (higher frame rate) and higher light usage (reduced exposure time) • See center-right drawing in Figure 3.3	2048x544 / 210 fps
Subsampling 2x2	• In each group of four native camera pixels only one pixel is read out and in a pixel in the outcome image. • Results in reduced resolution (higher frame rate) • See right-most drawing in Figure 3.3	1024x544 / 420 fps

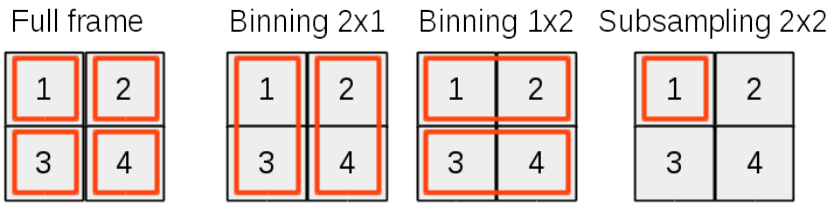


Figure 3.3: Scan modes supported by the camera.

Second main component of the sensing head of the MWLP prototype are the line lasers. These are broadly used laser modules containing semiconductor-based diode laser and the beam-shaping optics in a compact common housing. As explained in section 1.1.2, the lasers are mounted vertically with respect to the movement direction. A list of lasers tested for the MWLP prototype is given in Table 3.2. A photo of one of the line laser modules is given by

Figure 3.4. The power supply to the lasers can be switched programmatically using a transistor array and a microcontroller. I.e., the lasers can be turned on and off automatically depending on whether the system is scanning or not [137].

Table 3.2: Line lasers used with the MWLP prototype so far. [137]

#	Product	Wavelength	Optical Power	Laser class
1	Laser Components FP-MVnano-405-5M-45-F	405 nm	5 mW	2M
2	Picotronic LD405-10-24(20x80)-F800	405 nm	10 mW	1
3	Laser Components FP-MVnano-532-5M-45-F	532 nm	5 mW	2M
4	Picotronic LD532-10-24(20x80)	532 nm	10 mW	1
5	Laser Components FP-65/5LOF-Ö45-HOM	650 nm	5 mW	2M
6	Picotronic LH650-16-24(20x80)-F800	650 nm	16 mW	2
7	Laser Components FP-MVnano-780-45-F	780 nm	5 mW	2M
8	Picotronic LH850-30-3(16x45)	850 nm	30 mW	1M

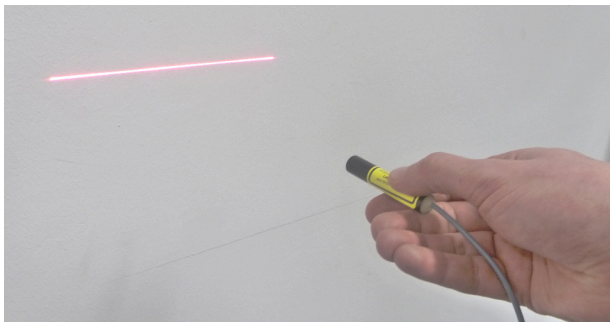


Figure 3.4: Photo of laser line module #5 (Laser Components Laser Components FP-65/5LOF-Ö45-HOM) with laser turned on.

Further, the sensing head is equipped with LED clusters for background illumination. The background illumination is required for the optical movement tracking, if ambient lighting is shaded out or insufficient. The LED clusters are of type Kingbright BL0106-15-28 [130]. A single LED cluster of this type is shown in Figure 3.5. The central wavelength of the light emitted by these clusters is at 940 nm, i.e., it is in the sensitive range of the camera but does not interfere with the wavelengths of the lasers used (cf. Table 3.2). There are 24 clusters mounted. They are connected to the power supply in groups of 8. Same as the lasers the power supply of these groups can be turned on and off programmatically using microcontroller and transistor array. This allows programmatically adjusting the background illumination depending on ambient lighting, such that the background is illuminated sufficiently for tracking, but not too much, such that it would disturb the line detection [137].

Figure 3.5: Single LED cluster Kingbright BL0106-15-28 (Diameter approx. 26 mm). Source: [130]

3.1.2 Control cabinet

The processing and control hardware required for operating the sensing head and processing the collected data through the image pipeline to a point cloud is mounted into a 50 cm x 40 cm x 21 cm control cabinet. The control cabinet is shown in Figure 3.6.

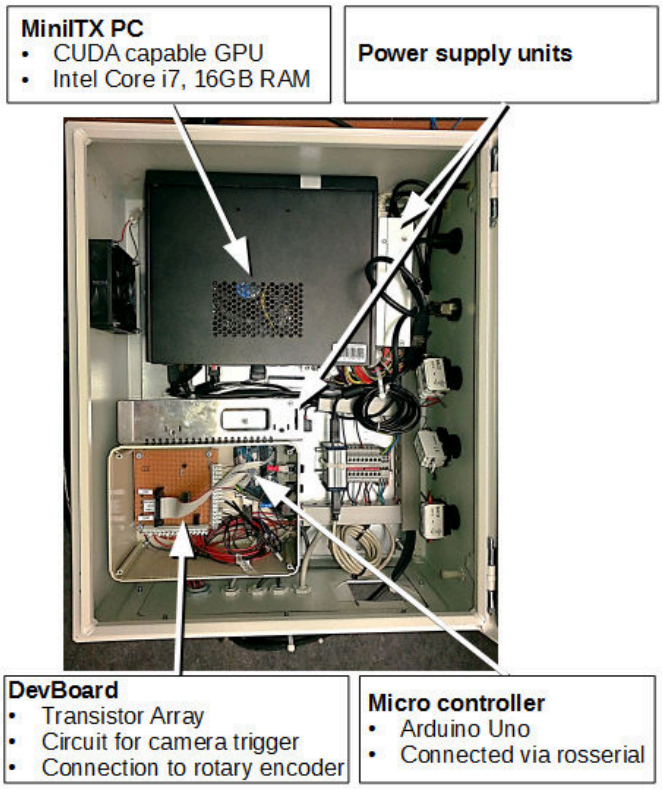


Figure 3.6: The control cabinet of the MWLP prototype.

The control cabinet contains a microcontroller of type Arduino Uno and a DevBoard connected to it. The DevBoard is equipped with a transistor array of type Darlington ULN2803AN (50 V 0.5 A, Octal, PDIP 18-pin). The transistor array connected to output pins of the Arduino and is used for switching the power supply of lasers and LED clusters. Further, there is a circuit with transistors and resistors for controlling the camera hardware trigger by the Arduino. Finally, the DevBoard sets up a connection between Arduino and rotary encoder. This design assures there

is a time-deterministic connection, i.e., 'hard real-time', from camera trigger to rotary encoder position and time stamp assigned to each image.

Further, the control cabinet contains a PC in MiniITX format. The camera is connected to the PC via Dual GigE Vision interface. The microcontroller Arduino Uno is connected via USB using rosserial [25]. The PC contains a CUDA-capable (CUDA) graphics card allowing performing parts of the image processing on the Graphics Processing Unit (GPU) using NVIDIA's CUDA framework as well as a Central Processing Unit (CPU) with four cores [137].

Table 3.3: Key features of the MiniITX PC mounted in the control cabinet [137]

Component	Type	Remarks
CPU	Intel Core i7 2600k	4 Cores @ 3.4 GHz
RAM	Kingston DDR 3	16 GB @ 2133 MHz Front Side Bus
Graphics Card	NVIDIA GeForce GTX 750 Ti	640 Stream Processing Units @ 1033 MHz 2048 MB GDDR5 RAM
Interfaces	2 x GigaBit Ethernet	Bonded for connection to Dual GigE camera
	1 x GigaBit Ethernet	General purpose communication via ROS with client PC
	USB 2.0	Connection to Arduino Uno via rosserial
	USB 3.0 / WLAN	General purpose communication

3.2 Software design and used tools

A variety of software tools and libraries have been used while conducting the work described in this thesis, particularly for setting up the MWLP prototype. These tools are explained in Section 3.2.1. A main commitment of all following developments was the use of Robot Operating System (ROS) as communication and processing backend. This allows splitting the process and control tasks into many small and reusable 'nodes' that communicate with each other while running in different processes and optionally on different machines. The nodes that are used for the MWLP prototype are listed in Section 3.2.2. Special issues for the use rosserial with Arduino and CUDA with OpenCV are explained the further subsections.

3.2.1 Used tools

The following tool have been used throughout the work on this dissertation.

Robot Operating System (ROS): ROS (www.ros.org) is an open-source robotic middleware and collection of other tools and libraries for use in robot development [108]. It was used here as backend for communication, thereby abstracting interprocess communication and optionally multiple machines. Further, several of its packages have been used, such as

the build system catkin (wiki.ros.org/catkin), the tool roslaunch (wiki.ros.org/roslaunch) and the visualization tools rviz (wiki.ros.org/rviz) and rqt (wiki.ros.org/rqt).

OpenCV: OpenCV (opencv.org) is an open-source library for image processing [15]. It was used as backbone for image processing tasks, including its GPU module which allows off-boarding processing tasks to a capable GPU device, if present.

NVIDIA CUDA framework: The Compute Unified Device Architecture (CUDA) framework (developer.nvidia.com/cuda-zone) is a proprietary platform for parallel computing on NIVIDIA GPUs. It provides a runtime API for accessing the GPU memory and scheduling as well as launching custom implemented processing tasks - so called 'kernels' - on the GPU [89]. It was used here directly with own implemented kernel methods as well as indirectly using the GPU module of OpenCV.

Point Cloud Library (PCL): The PCL (pointclouds.org) is an open-source library for processing and visualization of 3D point clouds [123]. It was used here for processing and visualization of the data created by the MWLP system.

Arduino: Arduino is an open-source platform for microcontroller hardware and software [66]. Same as the Arduino Uno hardware (cf. section 3.1), the software from the project has been used as well, in particular libraries and IDE for implementing the control code on the device. Further, for topic-based ROS communication with the Arduino Uno device the ROS package rosserial [25] has been applied (wiki.ros.org/rosserial).

3.2.2 Itemization of nodes

This chapter lists the nodes that are launched for the MWLP prototype to run in 'live' mode, i.e., capturing and online line detection, including visualization. Unless otherwise stated, the nodes have been implemented as part of this work.

`/serial_node`: The `/serial_node` is provided by the ROS package rosserial [25]. It sets up the topic-based communication with the Arduino Uno over serial USB communication.

`/sensor_module`: The `/sensor_module` includes the ROS driver for the Baumer HXG20NIR camera. Its implementation uses the Baumer GAPI [35]. It reads out the camera image buffers and publishes them as ROS `sensor_msgs/Image`. Moreover, it listens to messages published by the Arduino Uno that triggers the camera (cf. 3.1.2). Having both, image buffers and the trigger message from Arduino, that contains time stamp and position of the rotary encoder while triggering, it assembles the correct trigger message with the image. Hence, nodes listening the images always have both combined, image data together with real-time rotary encoder position and time stamp. Finally, the `/sensor_module` configures the camera trigger of Arduino and the camera in accordance with the relevant parameters set to the ROS parameter server.

`/line_detection_node`: The `/line_detection_node` subscribes to the image and pos/time stamp data published by the `/sensor_module`, performs matching, line detection and assembly and publishes `mwlp_scan` data, i.e., single scan lines assembled of all mounted

and active lasers with distance information, reflectance and backscattering data for all monitored wavelengths. Additionally, it loads the line calibration on start up and makes it available to all nodes that require it. It is most complex node that was implemented for the MWLP prototype. Details on its implementation are given in Sections 3.3 and 3.4. Its calibration will be described in Section 3.6.

`/lighting_controller`: This node retrieves the calibration from the `/line_detection_node` and configures the power supply to lasers and LEDs via Ardunio in accordance with the calibration, e.g. turning on the used line lasers.

`/scan_buffer_node`: The `/scan_buffer_node` listens to the `mwlp_scans` published by the `/line_detection_node`. As the processing in the `/line_detection_node` takes place multi-threaded and widely asynchronously the `mwlp_scans` do not safely arrive in correct order. Hence, this node sorts them, in accordance with the calibration retrieved from `/line_detection_node`. Further, it aggregates a configurable number of `mwlp_scans` (e.g. 200) and republishes them as `mwlp_chunks`. This is done in order to assure that Graphical User Interface (GUI) updates are not issued for each and every scan of the MWLP system, i.e., GUI updates at a lower refresh rate than the scan rate of the system. Further, aggregation of chunks is separated from the `/line_detection_node` because in this manner if line detection and visualization run on different machines the `/scan_buffer_node` runs on the machine with the visualization, i.e., the scans are transferred via the network. This means (single) scans might get lost during transfer over network but loss of (whole) chunks is unlikely. The `mwlp_chunks` are basis for both main visualization sinks (image and point cloud visualization) and is (later on) basis for the `/mwlp_classification_pipeline`. Finally, the `/scan_buffer_node` offers a ROS service that can be used to save the scans currently in buffer as image files to the file system for inspection purposes.

`/mwlp_point_cloud_publisher`: The `/mwlp_point_cloud_publisher` listens to the `mwlp_chunks` published by the `/scan_buffer_node` and transforms them into point couds, as will be described in Section 3.5.4. The point cloud is of type XYZRGB and is colored from the MWLP data using a configurable visualization model (cf. Section 3.5.3). Then, the derived point cloud is published for visualization in /rviz.

`/mwlp_piped_image_publisher`: The `/mwlp_piped_image_publisher` listens to the `mwlp_chunks` published by the `/scan_buffer_node` and transforms them into RGB images. The transformation from the overlayed MWLP images with distance information and spectral features to RGB images using a configurable model, which will be topic of Section 3.5.3. The images are then published as a standard ROS message in order to show them in the /rqt_image_view.

`/rviz`: The node `/rviz` is provided by the ROS package of same name [47]. It is used for 3D visualization of the MWLP sensor data as colored point clouds. However, mostly for online point cloud visualization the high resolution data of the MWLP system must be down sampled for fluid rendering in `/rviz`. This is also done by the `/mwlp_point_cloud_publisher`.

/rqt_image_view: In order to provide online views of the MWLP sensor data at full resolution the (2D) image visualization of the data published by the `/mwlp_piped_image_publisher` is possible. It can be shown in any image viewer, such as the rqt plugin `/rqt_image_view` with is provided by the ROS package of the same name [149].

/overlay_viz_controller: The `/overlay_viz_controller` is an own implemented rqt plugin that provides a GUI for adjustment of the visualization model used by the `/mwlp_point_cloud_publisher` and `/mwlp_piped_image_publisher` as well as later on the `/mwlp_classification_gui`.

3.2.3 PROGMEM usage on Arduino with rosserial

For setting up the graph-based topic communication on the Arduino many constant strings are required. Each publisher or subscriber must have one string with the message type name, one string with the md5sum of the message and one string containing the topic name to publish to. I.e., constant strings are required per connection. Moreover, log messages passed on to the standard log method of rosserial are also typically passed as constant strings.

Ardunio microcontrollers have very restricted resources. Particularly, the Arduino Uno has only 2048 KB of SRAM. This SRAM can be easily used up by usage of long or many constant strings of standard C type, i.e., as constant char arrays. SRAM overflow results in undefined behavior of the device.

Now that rosserial uses many constant strings it can easily use up SRAM, if many subscribers are used, descriptive log messages should be posted or topic names within namespaces are intended. To overcome this the rosserial package was modified in order to allow the usage of flash memory for such information. The modified rosserial version then only reads out the strings from the flash memory of the device (PROGMEM) when they are used and releases the memory as soon as they are no longer required. I.e., the strings must not stay in SRAM the entire application life-time, which reduces the SRAM usage significantly.

3.2.4 Distributed memory management with asynchronous CUDA streams

Using CUDA kernels in a process with a distributed, multi-threaded software architecture with many methods calling into CUDA kernels and without centralized memory management turned out to be an issue. In order to understand the problem that was to be solved here, some knowledge of the streaming and parallelizing techniques provided by CUDA is required. Therefore, a brief overview of CUDA streams is given in Digression 3.1. Readers familiar with CUDA streams may feel free to continue reading at the end of the digression box.

Digression 3.1: Brief overview of CUDA streams

The CUDA framework allows executing custom code on the GPU, thereby highly speeding up its execution for simple repeated operations. The high number of stream processing units, e.g., 640 for the graphics card of the PC of the MWLP prototype (cf. table 3.3), can perform simple operations on many objects in parallel. GPUs typically have groups combining a single Control Unit (CU) with multiple Execution Units (EUs). In contrast, CPUs typically map one CU with one EU. Therefore, simple operations, which have to be performed on many objects in the same manner, can be executed in parallel on GPUs using these sets of multiple EUs sharing a single CU. This is a common task in image processing, e.g., if - on CPU - an operation would be performed on all pixels of an image in a loop [89]. Introductory readings on the CUDA techniques are, e.g., [61] or [127].

The parallel execution in many stream processing units takes place by launching so-called kernels on the GPU. However, using the GPU-based techniques a new bottleneck occurs. This is copying the data from CPU memory (host memory) to the memory of the graphics card (device memory) for processing and back for accessing the results [78]. In order to reduce the impact of this bottleneck, there are so-called streams for asynchronously copying and processing data. Typically, CUDA-capable graphics cards have two or more copy-engines running parallel to GPU execution and independently from CPU. This enables another level of parallelisms, thereby speeding up the processing performance. Informations on the CUDA streams can, e.g., be found in [127] and [78]. For introducing into the streaming concept - first - a sequential example of pseudo code using CUDA's C API for calling into a CUDA kernel is given. It can be found in Listing 3.1.

Listing 3.1: A sequential launch into a CUDA kernel.

```
....
// pre-condition1: host_data is a uchar* pointer pointing memory filled with
//                 data to be processed of size_t data_size

// pre-condition2: host_result is a uchar* pointer pointing memory pre-
//                 allocated for the processing result of size_t result_size

// allocate memory for data on device
uchar* device_data = NULL;
HANDLE_ERROR( cudaMalloc((void **)&device_data, data_size) );

// copy the data to the device
HANDLE_ERROR( cudaMemcpy(device_data, host_data, data_size,
                         cudaMemcpyHostToDevice ) );

// allocate memory for the result on device
uchar* device_result = NULL;
```

```
HANDLE_ERROR( cudaMalloc((void **)&device_result, result_size) );

// launch into kernel
my_kernel<<<blocks, threads>>>( device_data, data_size,
                                device_result, result_size );

// copy the result from the device
HANDLE_ERROR( cudaMemcpy(host_result, device_result, result_size,
                         cudaMemcpyDeviceToHost ) );

// free device memory
HANDLE_ERROR( cudaFree(device_data) );
HANDLE_ERROR( cudaFree(device_result) );

// post-condition: the memory pointed to by host_result now contains
//                 the result calculated by the GPU
....
```

The code given in Listing 3.1 allocates memory on the device for input data and result. It copies the input data to the device then launches the kernel. The kernel will process the input data using a number of parallel executing stream processors (blocks, threads) and will write the processing results to the device memory pointed by `result_host`. This result data must then be copied back from the device to host memory to be accessible by the CPU.

All calls are sequentially performed in Listing 3.1. The function calls are synchronous. This means all function calls fall back when all required actions are fully completed. E.g. `cudaMemcpy` will only fall back if the copy action is completely finished (assuming no error occurred) [92]. The CPU thread is blocked until then. The timing diagram for Listing 3.1 is shown in Figure 3.7.

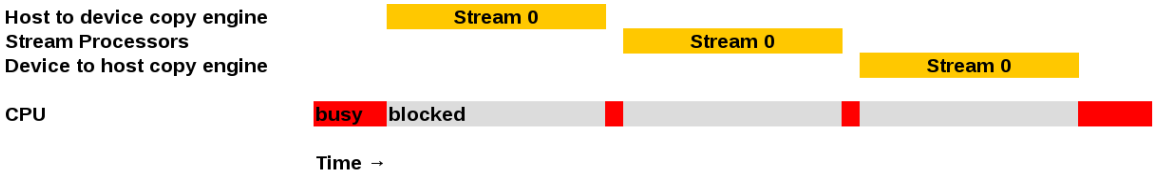


Figure 3.7: Timing diagram of the sequential calls.

In this case, the launch on stream processors is executed in parallel. This may cause a notable speed up with respect to simple operations on many elements that are executed in a loop on the CPU. However, copying data from and to the GPU memory is another bottleneck in this case. Moreover, the CPU thread is blocked the entire processing and copying time the graphics card requires, i.e., the CPU core cannot perform other tasks

while the graphics card is busy with the calls. To overcome this, `cudaStreams` and the asynchronous functions of the CUDA runtime API can be used. An example code that performs multiple asynchronous kernel launches for chunk-wise processing of a data set is given in Listing 3.2.

Listing 3.2: Asynchronous kernel launches, chunk-wise processing. Adopted from [127, p. 192-204] .

```
////  process data asychronously in N chunks
#define N (8)

////// at start up: create number of stream streams and allocate
//     both device and pinned host memory

// allocate pinned memory for data on host
uchar* host_data = NULL;
HANDLE_ERROR( cudaMallocHost((void **)&host_data, data_size) );
// allocate pinned memory for result on host
uchar* host_result = NULL;
HANDLE_ERROR( cudaMallocHost((void **)&host_result, result_size) );
// allocate memory for data on device
uchar* device_data = NULL;
HANDLE_ERROR( cudaMalloc((void **)&device_data, data_size) );
// allocate memory for the result on device
uchar* device_result = NULL;
HANDLE_ERROR( cudaMalloc((void **)&device_result, result_size) );

// create cuda streams
cudaStream_t streams[ N ];
for ( int i = 0; i < N ; i++ ) HANDLE_ERROR( cudaStreamCreate( &(streams[ i ]) ) );

....
////// fill host_data
....

// condition: host_data is a uchar* pointer pointing memory filled with
//            data to be processed of size_t data_size

////// trigger asychronous chunkwise processing
//  assume data_size and result_size can be divided by N without rest
size_t data_chunk_size = data_size / N;
size_t result_chunk_size = result_size / N;

for ( int i = 0; i < N; i++ )
{
  // copy the data to the device
  HANDLE_ERROR( cudaMemcpyAsync(
         device_data + (data_chunk_size*i), host_data + (data_chunk_size*i),
```

```
        data_chunk_size, cudaMemcpyHostToDevice, streams[i] ) );

  // launch into kernel
  my_kernel<<<blocks, threads, 0, streams[i]>>>(
        device_data + (data_chunk_size*i), data_chunk_size,
        device_result + (result_chunk_size*i)  );

  // copy the result from the device
  HANDLE_ERROR( cudaMemcpyAsync(
        host_result + (result_chunk_size*i), device_result + (result_chunk_size*i),
        result_chunk_size, cudaMemcpyDeviceToHost, stream[i] ) );
}

....
////// the CPU thread can now do other tasks while the GPU is working on the streams
....

////// synchonize with streams before accessing result_host
for ( int i = 0; i < N; i++ )
{
  HANDLE_ERROR( cudaStreamSynchronize( streams[i] ) )

  // chunk i in result_host can now safely be accessed
  ....
}

....

////// clean up before closing

// free device memory
cudaFree(device_data);
cudaFree(device_result);
// free pinned host memory
cudaFreeHost(host_data);
cudaFreeHost(host_result);
// remove streams
for ( int i = 0; i < N ; i++ ) HANDLE_ERROR( cudaStreamDestroy( streams[ i ] ) );
```

The code given in Listing 3.2 first allocates pinned or page-locked host memory. This is Random Access Memory (RAM) memory used by the CPU which is locked for paging, i.e., the Operating System (OS) is not allowed to move this data to swap areas on hard discs. The asynchronous copy actions to and from the GPU memory happen using Direct Memory Access (DMA) without interference of the CPU. This is only possible if the data is safely placed in physical RAM rather than virtual swap RAM [127, p. 186].

Next, in Listing 3.2 the device memory is allocated and a number of streams are created. All this happens during start up. After this, the pinned host memory is filled with the data to be processed. It is followed by the asynchronous copies and launches. In a loop for each stream calls into `cudaMemcpyAsync` from host to device are followed by a kernel launch with the respective stream specified and a call into `cudaMemcpyAsync` copying from device to host. Note that these calls do not block the CPU until they are done but only pop another action on the respective stream that can be executed as soon as the scheduler of the GPU finds it appropriate. This means after the asynchronous chunk-wise processing is triggered the CPU thread is free to perform other tasks.

In order to safely access results derived on the GPU with the CPU it is required that the CPU thread is synchronized with the respective GPU stream before. This can further be seen in Listing 3.2. Finally, before closing the application the allocated memory has to be freed and the created streams have to be destroyed.

The timing diagram for Listing 3.2 could look like Figure 3.8. It can be seen that the streams (S1 - S8) processing the different data chunks are using the different resources of the graphics card in parallel. This results in an overall processing time reduced with respect to the timing shown in Figure 3.7. Moreover, while the GPU is busy with processing the data, the CPU is not blocked, i.e., can perform other tasks.

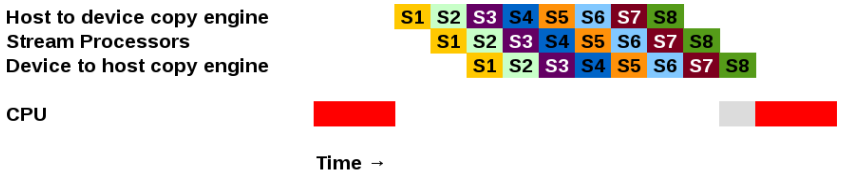


Figure 3.8: Timing diagram of the asynchronous calls.

When using the CUDA framework is important to note that a proper asynchronous timing as shown in Figure 3.8 is only been achieved if a stream different from stream zero is assigned to all asynchronous calls, the memory to be copied is page-locked as shown above and no implicit synchronization happens. Implicit synchronization occurs if a synchronous CUDA runtime API function, such as e.g. `cudaMalloc`, `cudaFree` or `cudaStreamCreate`, is called while streams are active on the GPU [78]. It does not depend whether this is done from the thread that invoked the stream action before or another thread of the application. A synchronous CUDA function called while the GPU is processing calls `cudaDeviceSynchronize`, i.e., blocks until all invoked asynchronous calls are done, before performing. This implies that for real asynchronous behavior of an application using the CUDA framework it is necessary to preallocate all device and page-locked memory and streams at start up, then reuse the buffered memory for all asynchronous launches and free it only at the end before closing.

This is a serious design problem, when using the CUDA framework in a multi-threaded application with distributed memory management and many functions invoking CUDA kernels

for different purposes. To illustrate this at the example of a function from the gpu module of OpenCV, the behavior of `cv::gpu::add` is analyzed here. Its function signature is:

```
void gpu::add(const GpuMat& a, const GpuMat& b, GpuMat& c, const GpuMat& mask =
              GpuMat(), int dtype=-1, Stream& stream=Stream::Null() )}
```
[141]

Its parameters are:

- "**a** - First source matrix.
- **b** - Second source matrix to be added to a . Matrix should have the same size and type as a .
- [...]
- **c** - Destination matrix that has the same size and number of channels as the input array(s). The depth is defined by dtype or a depth.
- **mask** - Optional operation mask, 8-bit single channel array, that specifies elements of the destination array to be changed.
- **dtype** - Optional depth of the output array.
- **stream** - Stream for the asynchronous version."[141]

OpenCV also features a decentralized memory management. The `cv::gpu::GpuMat` class, same as its CPU counterpart `cv::Mat` implement reference counting [142]. It means that multiple headers share the same data field. Copying the class instance only copies the header, i.e., flat-copy, rather than copying the entire data, i.e., deep copy. The copy constructor thereby increments the reference counter, while the destructor decrements it. As the reference counter reaches zero, i.e., the last header pointing the data field is destructed, the data field is freed.

Using reference counting in general unbinds the application programmer from the burden of memory management. This is particularly a favor in a distributed architecture with many classes. In particular, it frees objects immediately as soon as they are no longer accessible [11]. However, its use by OpenCV in combination with CUDA may cause unintended implicit synchronization because the constructors and destructors of `cv::gpu::GpuMat` may call into `cudaMalloc` or `cudaFree` at points were this is not intended by the programmer and hard to trace. For instance, let the above mentioned function `cv::gpu::add` be called in an application while other CUDA streams are being processed by the graphics card asynchronously. In this case, `cv::gpu::add` will only behave fully asynchronous, i.e., does not wait for its own completion nor for completion of the other streams that have been launched anywhere else in the application, if and only if

- the destination matrix **c** is a preallocated matrix with same size like **a** and **b**
  - ⋆ *and*
- **dtype** is -1 and **c** has same type like **a** and **b** or **dtype** corresponds with the type of **c**
  - ⋆ *and*
- **stream** is a proper pre-created stream, i.e., not the default value.

For not passing a stream, synchronous behavior can be assumed as intended by the programmer. However, if the passed destination matrix **c** is passed as empty header or does not match the

type and or size of the processing result `cv::gpu::add` creates a new matrix data field into which the result is written and which is assigned to the header reference passed as argument **c**. I.e., it allocates (and possibly frees!) data for the GPU using `cudaMalloc`, i.e., it causes implicit synchronization with the streams that have been launched before. This synchronization can be assumed to be not intended by the application programmer. Moreover, he can only avoid it if and only if

- he assures that proper result matrices of correct size and type are passed on to all functions of the gpu module
  - ⋆ *and*
- he creates all needed GPU matrices at program start up
  - ⋆ *and*
- he makes sure that at least one header for all GPU matrices stays in scope during the entire program production time, such that the all data is freed only at the end of program execution.

In the end, using the version 2.4 of the OpenCV gpu module's API this means that for obtaining fully asynchronous behavior you cannot take advantage of the distributed memory management by the included reference counting but have to do your own resource pooling on top of it. This was a particular problem for the development of the `/line_detection_node` with its distributed architecture containing many classes that occasionally create matrices, including own implemented kernels (matching and line detection), usage of different functions from OpenCV gpu module's API and different wait states and processing steps in the multi-threaded image pipeline. A common resource pooling affecting all the modules of the architecture should be avoided.

Therefore, a memory pool of GPU memory, page-locked memory and CUDA streams was implemented. It replicates most of the CUDA runtime API in a separable namespace, thereby it was possible to bind it beneath OpenCV. The design is shown in Figure 3.9. At start up, the application - here the processing pipeline of the `/line_detection_node` - calls `initialize` on the memory pool. This triggers it to allocate an amount of GPU memory as well as of page-locked memory and create a number of CUDA streams using the synchronous CUDA methods `cudaMalloc`, `cudaMallocHost` and `cudaStreamCreate`. After initialization, it replicates the CUDA runtime API directly passing asynchronous calls, such as kernels or `cudaMemcpyAsync`. However, the malloc and free methods as well as `cudaStreamCreate` and `cudaStreamDestroy` are not passed on to the CUDA runtime API but handled by the memory pool itself, e.g. by passing a pointer to chunk of respective size with in the preallocated memory to calls on `cudaMalloc`. Consequently, OpenCV gpu module's functions, such as `cv::gpu::subtract` or `cv::gpu::warpPerspective` as well as the constructors and destructors of `cv::gpu::GpuMat`, can now be called by the MWLP image pipeline without worrying about implicit synchronization. Further, no memory preallocation is required for modules of the pipeline that implement own CUDA kernels, such as the matching and the line detection. After the productive life-time of the application, the program must call finalize on the memory pool. This causes it to free its memory and release all pre-created resources.

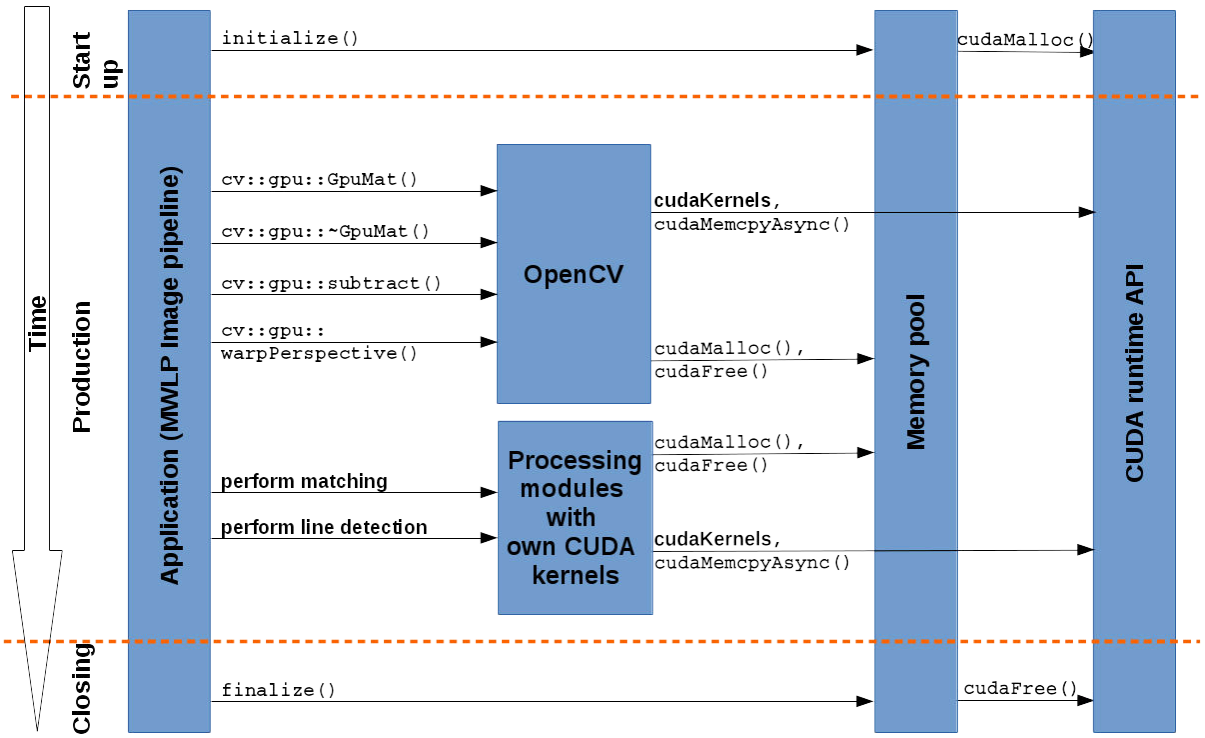


Figure 3.9: Design of the Memory Pool.

The memory pool can also be used without calling `initialize` (and `finalize`, consequently). In this case, it just passes all calls to CUDA runtime API including `cudaMalloc` and so on, i.e., having the same synchronization behavior like the CUDA runtime API.

Consequently, the memory pool implemented here allows using distributed memory management by reference counting on CUDA GPU memory without causing implicit, unintended synchronization of the applied CUDA streams.

3.3 Image processing steps for line detection and line assembly

This section focuses on the image processing pipeline implemented for line detection and line assembly in the `/line_detection_node`. This image processing pipeline processes the incoming camera images down to assembled `mwlp_scans`. The images, thereby, can fall into different processing and wait states. An overview of the image pipeline is given in Figure 3.10. The pipeline is further explained in [137].

The processing of the images in the queue is performed by multiple worker threads executed in parallel. Apart from the line filtering and the line assembly all processing steps are implemented on CPU and on GPU. It can be selected using preprocessor directives whether each step may be

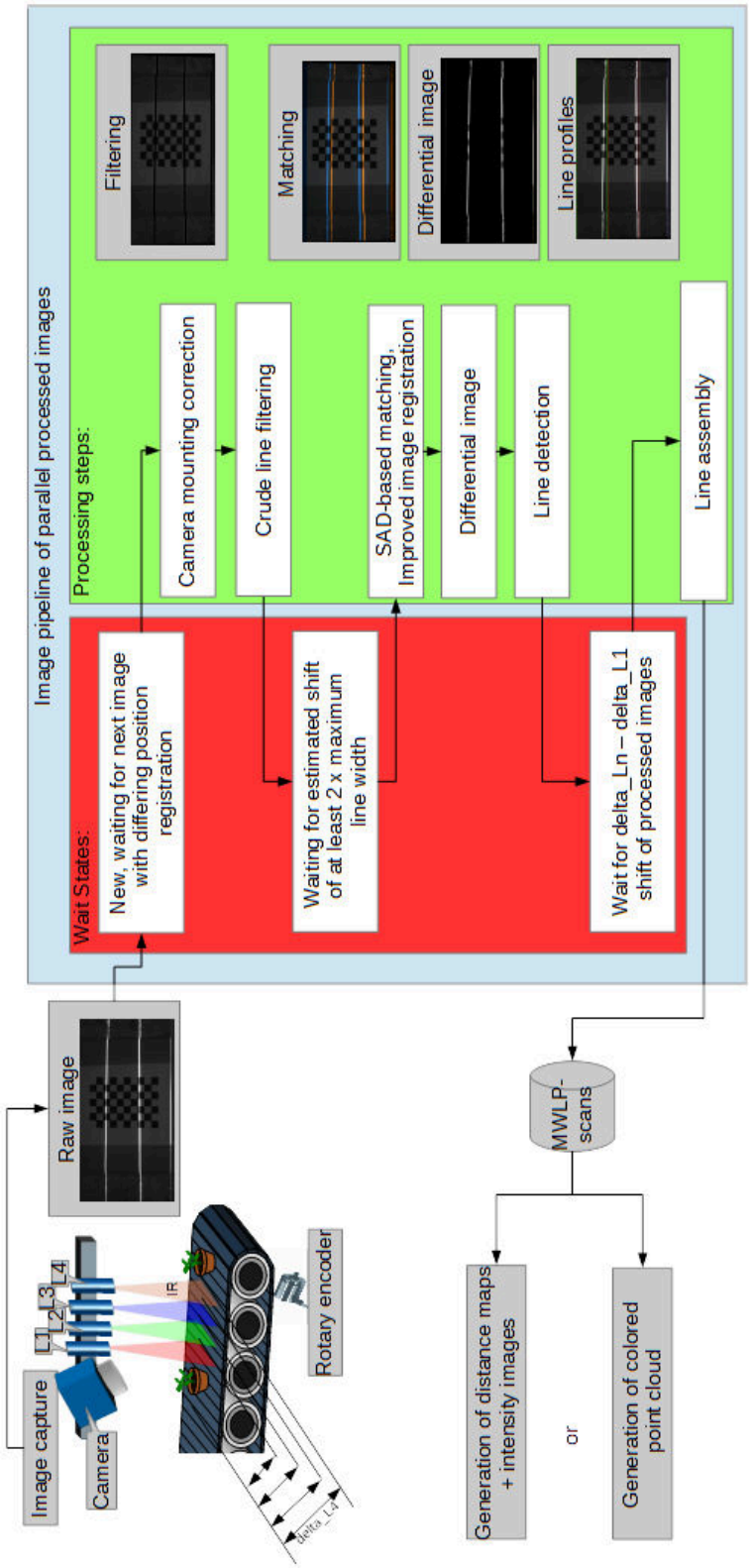


Figure 3.10: Overview of the multi-threaded image pipeline of the `/line_detection_node`.

performed on CPU or GPU. However, the GPU version has shown to have favorable performance for all those steps. Further, the `/line_detection_node` can be built with a flag such that it saves the image after each processing step to the file system for debugging and inspection purposes. It this case only approx. 5 image per second (compared to 100 full frames without image logging) can be processed, though.

The following subsections will now give in depth information on all pipeline steps seen in Figure 3.10.

3.3.1 Image acquisition

- **Processing step 0: Raw image**

The image acquisition is done by the ROS node `/sensor_module`. It reads out the camera while the required lasers are turned on, assembles the image information with rotary encoder based position stamp and time stamp from Arduino and publishes it to a ROS topic, as mentioned in Section 3.2.2. The `/line_detection_node` then subscribes to the image topic. For improved performance it is also possible to run the relevant classes from `/sensor_module` and `/line_detection_node` in ROS nodelets [30]. In this case, both modules run in the same process and the image data is passed using shared pointers, i.e., there is no added computational cost for copying the image data between multiple processes.

An example for an original image as it could occur when running the MWLP system is given in Figure 3.11.

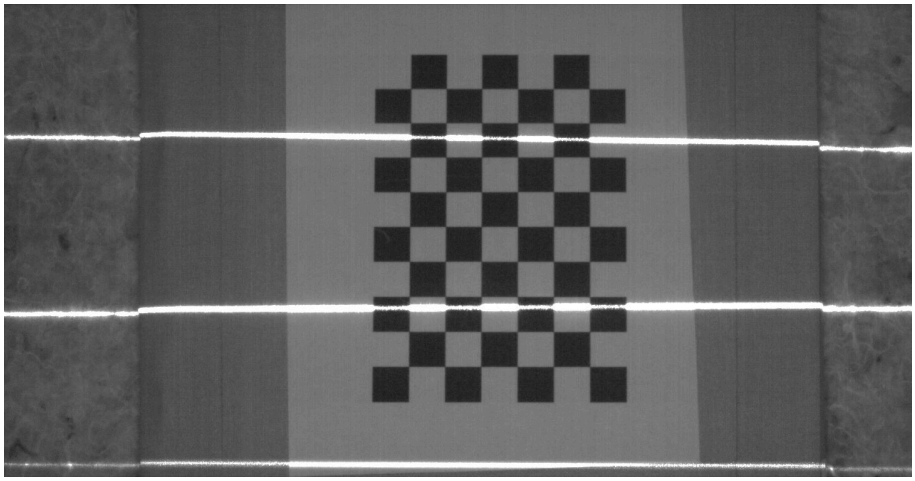


Figure 3.11: Raw image of three laser lines captured by the camera.

- ◇ **Wait state 0: Waiting for images to get appended**

Immediately after being pushed to the image pipeline the image gets into the first wait state (cf. Figure 3.10). As mentioned, the image has already a (crude) position based on the rotary encoder. For the MWLP system processing images does only make sense if the objects are in relative movement. A single image captured by the camera is insufficient for gathering MWLP data. Therefore, an incoming image always wait for a next image with different pose registration before its processing starts. Further, the images are sorted based on the pose registration before processing. I.e., if there is no relative movement between sensor and objects detected, only one image will be pushed to the pipeline and remain in this wait state. All subsequently following images are discarded and no processing takes place.

3.3.2 Mounting correction

- **Processing step 1: Correction of angular camera mounting (conveyor skewness)**

After images are pushed to the pipeline and a movement of the objects is provided, the processing starts with the correction of the angular camera mounting. As required for triangulation, the camera is mounted angular with respect to the laser lines. As the lasers for complying with the MWLP concept (cf. 1.1.2) have to be mounted vertically, the camera mounting is slanted w. r. t. the conveyor. This means that the projected image plane of the camera is tilted compared to the object plane of the conveyor. This is depicted in Figure 3.12.

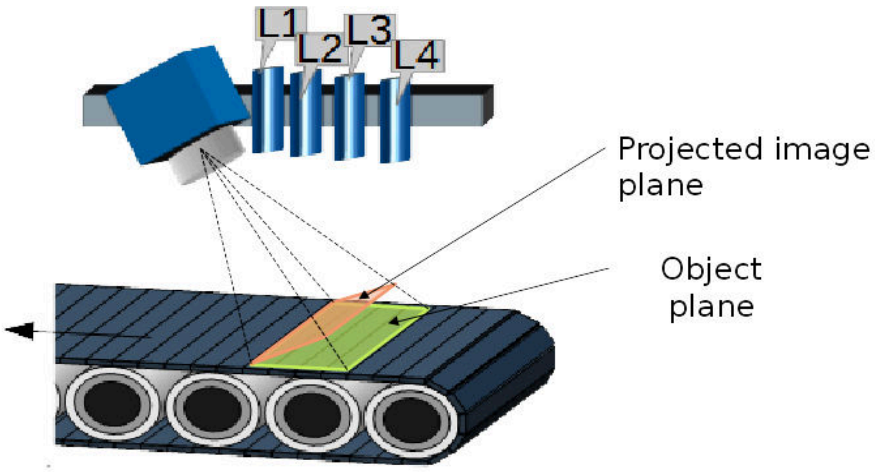


Figure 3.12: Scheme of tilted image plane.

This difference between the respective planes results in a FOV of the camera that is wider at the top of the image than it is at the image bottom. Consequently, pixel sizes are unequal in the different parts of the image. This is illustrated by Figure 3.13. There a raw camera image of a chessboard is shown that is noticeably tilted with respect to the projection plane of the camera. Hence, the chessboard fields at image bottom in this image appear to be smaller than the chessboard fields at image top. In Figure 3.14 this effect is corrected. Here, the chessboard corners are detected and the perspective of the image is transformed such that the projection

plane now matches with the chessboard plane. Consequently, all chessboard fields in Figure 3.14 have the same size. In Figure 3.13 this effect is exaggerated, though. For the real configuration of the MWLP system the differences cause by this correction are hardly notable from the images (compare Figure 3.15 with Figure 3.16). However, the improvements induced by this processing steps are measurable, as will be shown in Section 4.2.

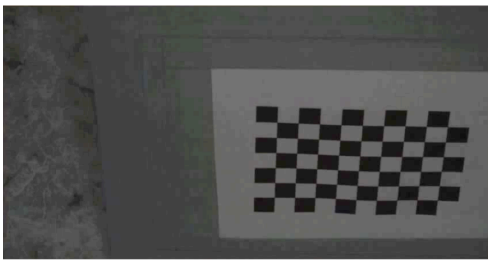


Figure 3.13: Uncorrected view of a noticeably tilted chessboard.

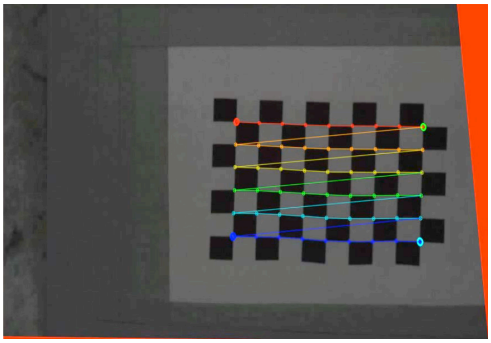


Figure 3.14: Corrected view for tilted image in Fig. 3.13.

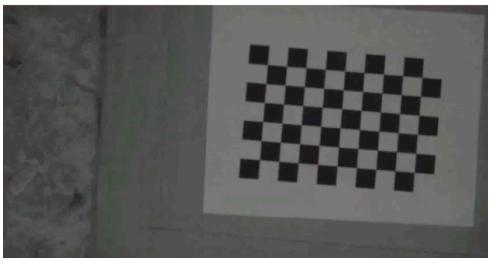


Figure 3.15: Tilted chessboard with tilt due to slanted mounting of camera of MWLP prototype.

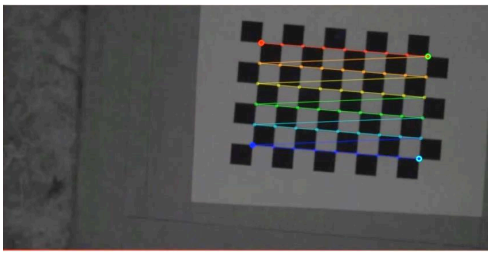


Figure 3.16: Corrected view for tilted image in Fig. 3.15

The mounting correction is performed by using the OpenCV methods `cv::warpPerspective` or `cv::gpu::warpPerspective` if performed on CPU or GPU, respectively. The required homography is calibrated ones during the calibration procedure of the MWLP system (cf. 3.6). It is then applied to each incoming image using the mentioned methods. Some figures on the runtime of the mounting calibration on the PC of the MWLP system (cf. Table 3.3) are given in Table 3.4 for both CPU and GPU. This analysis took place on the same data (processing from rosbag) and with the same processing parameters. Clearly the GPU outperforms here.

Table 3.4: Runtime of the mounting correction on CPU and GPU.

	Number of images	Average time per image	Standard deviation
CPU	1170	60.0 ms	24.6 ms
GPU	1171	2.96 ms	0.98 ms

3.3.3 Image matching

After the mounting correction was performed, the image matching of sequentially captured images is the next major task. It is needed to enhance the position registration for the images in order to improve the line assembly as the different wavelengths components in a single MWLP scan line originate from different camera images. The image matching is based on Sum of Absolute Differences (SAD).

An advantage of having the image matching to run after the mounting calibration is that (in ideal case) the conveyor movement can be modeled as shift in y-direction in the image as the image's projection plane is identical with the object plane of the conveyor. If the conveyor and camera vertical are not perfectly justified (non-ideal case), an additional x-shift may also be taken into account, but rotations and z-shifts must not be modeled in this constellation.

- **Processing step 2: Crude line filtering before matching**

Before the image matching can be performed, a crude line filtering takes place. This step is needed because - in order to track movement - the matching may only take into account pixels of the background of the images. The positions of the laser lines are (partly) caused by the laser mounting, which is not moved w. r. t. camera. I.e., the lines must be filtered out to assure that they do not disturb the matching. Note that a line detection is here not yet intended nor conducted. This step is basically a very crude filtering creating a mask for filtering out everything that could possibly be a line.

The result of line filtering is given in Figure 3.17. It is the processed version of the camera image given in Figure 3.11 with mounting correction and line filtering performed. As the figure shows, the lines are filtered out in Figure 3.17 and only the background remains structured. Moreover, the (slight) effects of the mounting correction can also be noted in Figure 3.17. Tiny black regions appear at bottom right and bottom left as the camera FOV is smaller at bottom, i.e., the perspective transformation results in undefined regions here.

The line filtering is done by adaptive thresholding followed by a dilation operation. The OpenCV functions `cv::adaptiveThreshold` and `cv::dilate` are used here. Unfortunately, there is no GPU counterpart for `cv::adaptiveThreshold` provided by OpenCV. Fortunately, using SSE acceleration the OpenCV implementation of `cv::adaptiveThreshold` is already relatively fast. The performance for processing the above mentioned data is given in Table 3.5. The average processing time per image with full camera resolution on the PC of the MWLP prototype is around 12.3 ms. This appears relatively long as the 100 images are to be processed per second,

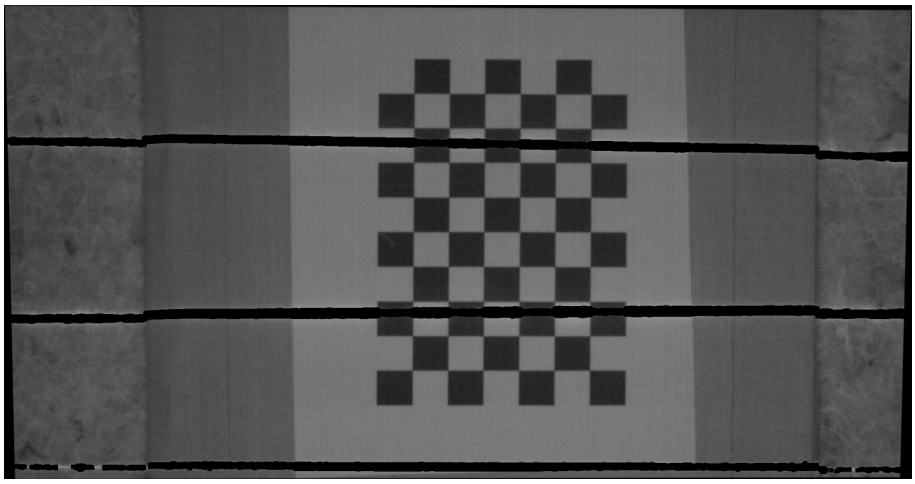


Figure 3.17: Camera image after mounting correction with filtered laser lines. Values of pixels that are masked by the filter derived mask are set zero, i.e., appear black.

Table 3.5: Runtime of the line filtering.

	Number of images	**Average time per image**	**Standard deviation**
CPU	1171	12.3 ms	3.41 ms

i.e., one image can (theoretically) block the (entire) processing resources for 10 ms. However, using hyperthreading the CPU can process eight threads in parallel meaning that only one out of eight virtual cores is blocked for the 12.3 ms with this task. Hence, processing full resolution images through the line filter in this constellation is still possible at up to approx. 640 Hz - theoretically and assuming there are no other tasks to be performed on the CPU. Therefore, an own implementation of the adaptive thresholding on the GPU was not required and for all tests described here the CPU implementation was used for line filtering.

◇ **Wait state 1: Waiting for movement of at least two times maximum laser line width**

After line filtering, the image falls into another wait state. Here, the image has to wait for a movement of the conveyor of at least two times the maximum laser line width (cf. Figure 3.10). This is needed due to the later following step of background subtraction when creating the differential image. For this, the shift between the subtracted images needs to be safely more than the laser line width. If the shift would be less than that the laser lines would be subtracted with each other, i.e., they would cancel out in the differential image rather than being enhanced.

• **Processing step 3: SAD-based image matching**

As sufficient conveyor movement for matching happened and both, the image and its counterpart to be matched with, are processed including processing step 2, the image matching is ready to

be performed. As mentioned, the matching is based on Sum of Absolute Differences (SAD), the Sum of Absolute Differences (SAD) of all pixels of both images are obtained and applied to represent the matching error.

Such methods convolving templates over the entire image are typically much slower than key-point based methods. Key-point based methods, such as SIFT [143] [77], ORB [144] [115] or FAST [145] [114], and key-point based homography estimation have also been tested for this step. The major favor of key-point based methods is that they can model arbitrary camera movements, i.e., movements in six Degrees of Freedom (DOF). Further, with huge search space they are usually much faster than convolving methods. However, they cannot be accelerated much by restricting the search space as key-point detection and feature extraction cause the major part of the required processing time. These steps do not depend on the search space, i.e., cannot be accelerated by restricting the search space. In contrast, convolution-based methods, such as the SAD-based matching, can be accelerated much by restricting the search space [137].

In this case, the search space is restricted. As mentioned at the beginning of the Section 3.3.3, the conveyor movements can be modeled thanks to the mounting correction by a combination of x- and y-shift, i.e., two DOF rather than six DOF. Further, a robust estimation for the shift is given by the rotary encoder position at the time the image was triggered. It can be used to further speed up the matching process. The SAD-based matching takes the rotary encoder position as initial guess. Then, using gradient method it optimizes the shifts such that the matching error is minimized to the next local minimum. As the rotary encoder delivers a stable and quite good estimation as input (just lacking a bit of resolution) this matching procedure provides very good results. As it is SAD-based, the matching error to be minimized is thereby represented as SAD of the images divided (normalized) by the sum of both images. Pixels where an image contains a zero, i.e., likely pixel of a laser line filtered out, are not evaluated here. The notation of this is given in Formula 3.1.

Formula 3.1: Minimizing logic of the SAD-based matching [137].

$$[x^*|y^*] = arg \min_{[x|y]} \left(\frac{\sum_{i,j} \begin{cases} abs(I1(i,j) - I2(i+x, j+y)) & , if I1(i,j) \neq 0 \wedge I2(i+x, j+y) \neq 0 \\ 0 & , otherwise \end{cases}}{\sum_{i,j} \begin{cases} I1(i,j) + I2(i+x, j+y) & , if\ I1(i,j) \neq 0 \wedge I2(i+x, j+y) \neq 0 \\ 0 & , otherwise \end{cases}} \right)$$

$$, where\ I1(i,j)\ and\ I2(i,j)\ are\ pixel\ values\ of\ the\ first\ and\ second\ image\ at\ position\ i,j$$

For the matching both CPU-based and GPU-based versions have been implemented as part of this work. The CPU-based version basically convolves the images in nested loops and calls into the OpenCV functions `cv::absdiff` and `cv::sum` with differently shifted ROIs for each convolution step. However, it is too slow for practical use. In the performance test mentioned for the previous processing steps, on average it took 468 milliseconds per image to run on CPU (cf. Table 3.6).

Table 3.6: Runtime of the SAD-based matching on CPU and GPU.

	Number of images	Average time per image	Standard deviation
CPU	1045	468 ms	194 ms
GPU	1051	1.88 ms	1.13 ms

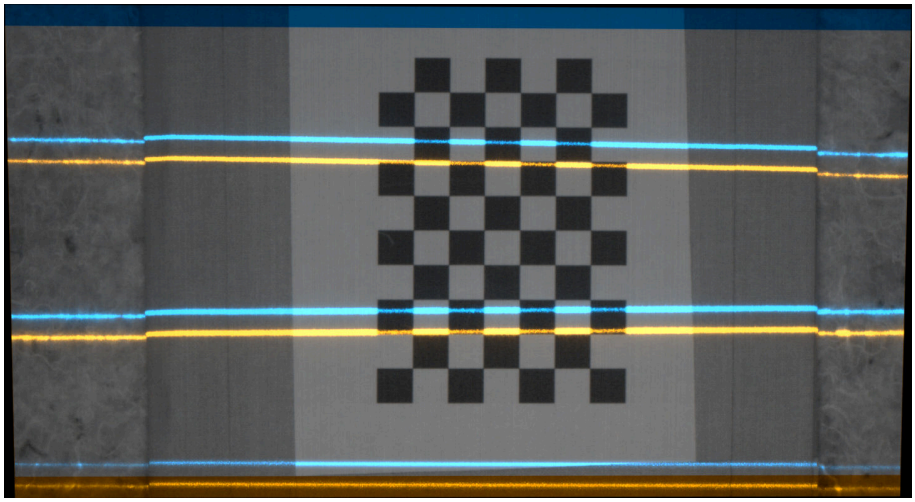


Figure 3.18: Overlay view of two matched images.

Now, it would be possible to just exchange `cv::absdiff` and `cv::sum` for `cv::gpu::absdiff` and `cv::gpu::sum` to obtain a GPU based version. In this case, during summing up the images in each convolution step, the pixels of the images would be treated in parallel. However, this would cause again a lot of interaction between the CPU and GPU as the convolution loops would still run on the CPU. As mentioned, the interaction between CPU and GPU is a major bottleneck for GPU-based processing. Therefore, an own CUDA kernel was implemented for the GPU-based matching version as part for this work. It not just treats the pixel in parallel but (mostly) even unrolls the convolution loops, such that entire convolution steps can run in parallel on the different stream processors of the GPU. This takes advantage of the parallel processing on the GPU even more than just using `cv::gpu::absdiff` and `cv::gpu::sum` in the loop, thus drastically reducing the runtime of the matching on GPU (cf. Table 3.6).

The result of the SAD-based matching is shown in Figure 3.18. The image shown in this figure is an overlay image created from two images that have been matched using this method. Both images are shifted with respect to each other. The applied shift was determined using the matching method. Thereby, one image is drawn orange, the other in a light blue. This means that everywhere, where the pixel values of the shifted images are identical, they sum up to a gray value. Only where the pixel values differ, orange or light blue regions appear.

As Figure 3.18 depicts, the overlay of the images is gray throughout the background information.

I.e., matching based on the background information worked out. The laser lines did not move with the conveyor movement and are not taken into account for matching. Hence, the laser lines appear colored.

3.3.4 Line detection

After the image matching is done, the line detection takes place. It is improved by a prior subtraction of the matched images.

- **Processing step 4: Differential image**

As Figure 3.18 shows, the laser lines are the only parts in the overlayed image that appear colored. This effect can also be used for the line detection. By subtracting two matched images the laser lines can be highly emphasized. Further, the background subtraction also partly cancels out ambient lighting. This improves the robustness of the system against disturbances by uncontrolled light, thereby reducing this critical point of the MWLP sensor concept (cf. Table 2.1 in Section 2.4). The subtraction result for the example image is depicted in Figure 3.19.

This technique for enhancing and improving the laser lines before detection is also e.g. used by DAVID laser scanners. There a scene with fixed constellation of camera and objects is monitored using a moved laser line. Thereby, a pre-captured image without laser is subtracted from the following images to enhance the laser line [95, p. 28]. This technique could also be used by LP sensors. However, for a conventional LP sensor with laser fixed w. r. t. camera, it requires optical tracking, which is a computationally expensive operation.

In contrast to conventional LP systems, for the MWLP approach the optical tracking with the SAD-based matching is performed anyway. It is required for the line assembly because otherwise the line assembly would only be possible based on the rotary encoder information, i.e., with significantly reduced resolution. However, since the matching is done anyway, the step of creating a differential image can be performed with practically negligible computational costs.

The creation of the differential image is also implemented on both, CPU and GPU. The versions many apply `cv::subtract` or `cv::gpu::subtract` with properly shifted ROIs of the respective images. As stated in Table 3.7, the processing times are relatively low, particularly for the GPU-based version.

Table 3.7: Runtime of the image subtraction on CPU and GPU.

	Number of images	Average time per image	Standard deviation
CPU	1049	1.89 ms	7.12 ms
GPU	1051	0.757 ms	0.340 ms

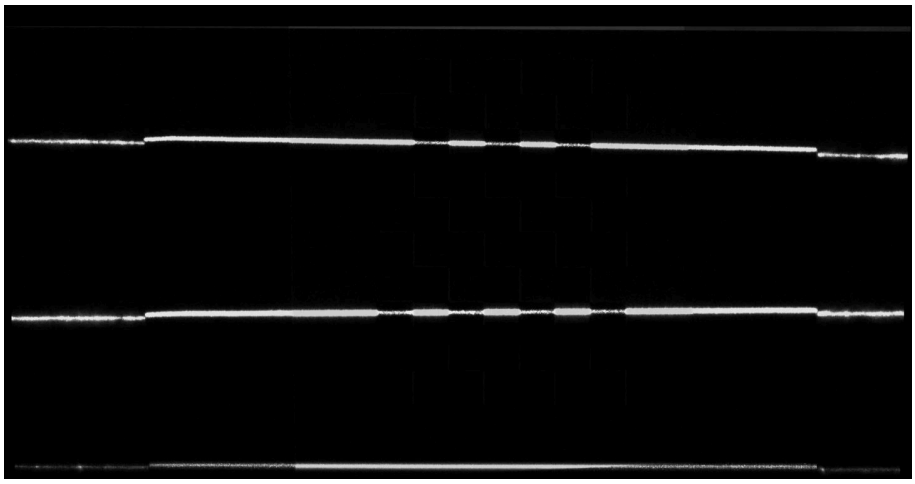


Figure 3.19: Result of subtracting two matched images

- **Processing step 5: Line detection**

The image subtraction result gives a very significant, highly normalized input for the line detection. The line detection under these conditions provides good results using a simple threshold. It searches for line segments in each image column. Thereby, as mentioned, the different laser lines are searched in different ROIs or the image. Only one detected segment is allowed for each ROI, otherwise nothing is detected. More details on the line detection procedure and particularly on the feature extraction from the laser lines, that happens along with this step, will be given Section 3.4.

Figure 3.20 depicts the example image with the detected laser lines drawn into it. The straight horizontal lines drawn into it are the boundaries of the ROIs in which the laser lines are searched. The line detection is also implemented on both, CPU and GPU. Here, the CPU version is also optimized taking advantage of the 64 wide word-width on common devices for comparing 8-Bit pixels in 8 adjacent image columns at once using a bit-mask. However, the GPU version is still faster even though the speed up is not so significant like for some operations mentioned before. The respective runtimes are given in Table 3.8.

Table 3.8: Runtime of the line detection on CPU and GPU.

	Number of images	**Average time per image**	**Standard deviation**
CPU	1049	4.46 ms	4.30 ms
GPU	1051	2.33 ms	0.80 ms

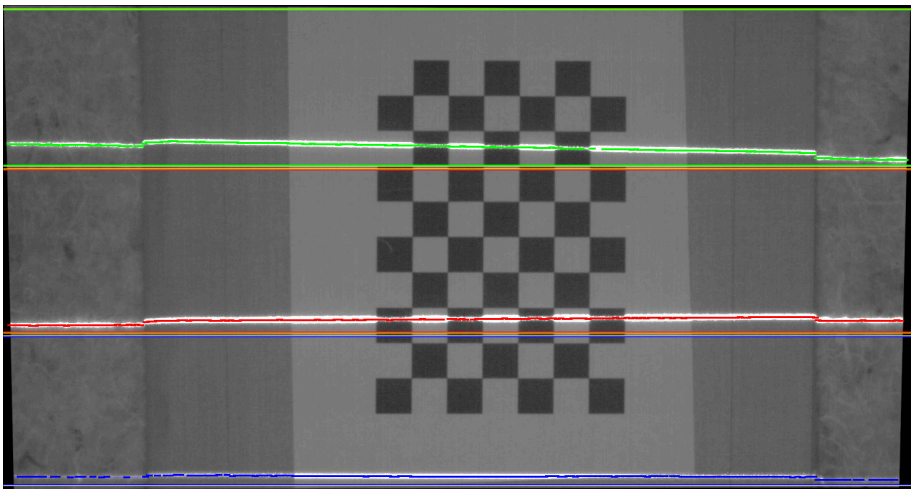


Figure 3.20: Image with multiple detected laser lines of different wavelengths.

3.3.5 Line assembly

After the line detection is done, the line assembly follows as last step before `mwlp_scans` can be published out from the `/line_detection_node`.

◇ **Wait state 2: Waiting for movement of calibrated distance between first and last laser line**

Before the line assembly can be undertaken, the image enters the last wait state of the image pipeline. Here, it has to wait for until the conveyor has moved the calibrated distance from the first to the last laser line (cf. Figure 3.10). A scan of the MWLP system is only possible after the occurring objects have been scanned by all mounted lasers. Further, all images captured over the mentioned distance have to be processed including processing step 5, i.e., line detection before a single image can leave this state for line assembly.

• **Processing step 6: Line assembly**

The line assembly works on the detected laser lines including the extracted features in multiple images. It assembles the features extracted for lasers of different wavelengths in different images taking into account the pre-calibrated distances between the laser lines and assuming a vertical mounting of the line lasers as mentioned in Section 1.1.2. I.e., it assembles the scans of the same object position which is captured at different times with the different lasers.

It is possible to perform an additional scan check on the assembled data. This check assures that for a constellation with e.g. 3 line lasers at least 2 or all 3 lasers have been detected. If fewer lasers have been detected, even the values of the detected laser lines are not trusted and discarded. Further, it can be adjusted to assure that the z-values gathered from the different lines comply with each other. In this case, scan points are rejected if any of the obtained z-values

differs from any other more than some value, e.g. 5 mm. This improves the reliability of the obtained 3D scan data.

The line assembly is only implemented on CPU. As it works on the detected lines, i.e., on scan data rather than image data, it is less computationally expensive. The runtime for processing an image, i.e., assembly of a single scan line, is given in Table 3.9. After line assembly, the assembled scan data is published. The image objects, which passed this step, stay in the queue until it is safe that they will not be required for line assembly of other image objects. Finally, the respective object in the image queue is erased.

Table 3.9: Runtime of the line assembly.

	Number of images	**Average time per image**	**Standard deviation**
CPU	775	1.73 ms	1.56 ms

3.4 Feature extraction from the detected laser lines

This section is dedicated to the line detection and feature extraction from the laser lines as it is done during processing step 5 of the MWLP image pipeline (cf. Section 3.3.4). Understanding the feature extraction is prerequisite for understanding visualization and analysis of the MWLP data in the following parts. Due to this an in depth explanation of line detection and feature extraction is given here in addition to the brief overview during the description of the image pipeline in Section 3.3.4.

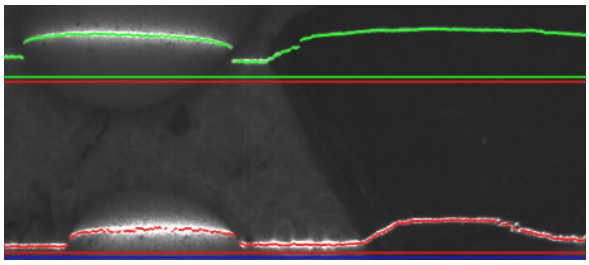


Figure 3.21: Image excerpt of camera image scanning potato (left) and stone (right) with two lasers including detected laser lines [138].

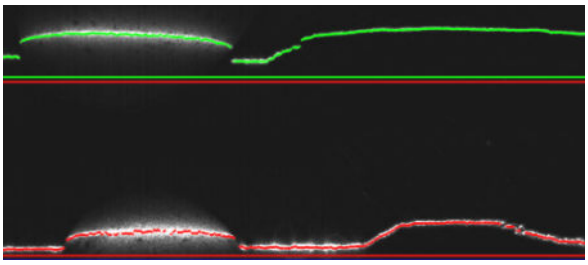


Figure 3.22: Differential image corresponding to image in Figure 3.21 including the detected laser lines [138].

The feature extraction is described at the example of the camera image excerpt shown in Figure 3.21. Here, potatoes (left-hand side), i.e., objects with high water content, and a stone, i.e., and optically dense object are scanned by two line lasers. The respective differential image is given by Figure 3.22. As expected, the laser lines are emphasized in the differential image while the background information is canceled out. Moreover, the scattering of the laser lines by the potatoes can clearly be seen in the differential image because the scattering is also caused by the

lasers which do not move with the matched positions. As expected, the laser lines are scattered by the potatoes because the laser light partly enters them and is scattered back beneath the object surface. In contrast, the stone on the right-hand side of the image does not scatter the laser lines and only reflects a small line (cf. Figure 3.23).

3.4.1 Line detection

The line detection is performed on the differential images. The images are searched column-by-column for laser lines in the respective ROIs. This is illustrated by the following images. Figure 3.24 shows the image data of the differential image in Figure 3.23 as heat map. A blue colored pixel thereby represents the minimum gray value, red color the maximum value. The image resolution is down sampled for visualization purposes [138].

Figures 3.25 and 3.26 contain the same data like Figure 3.23 and 3.24. They represent the data as morphological 3D views from different perspectives, where the z-values correspond with the gray values of the respective pixels. Particularly in Figure 3.26 the effects of scattering can be noted. In the front part of Figure 3.26 (corresponds to the left-hand side of Figure 3.23 where the lines are significantly scattered) the peaks of the lines have quite smooth endings at bottom. In contrast, at rear (corresponding to the left-hand side of Figure 3.23 without much scattering) the peaks caused by the laser lines have relatively sharp edges at bottom.

For the line detection the algorithm runs along the image columns (Figure 3.26). Figure 3.27 depicts two examples for profile cuts through the morphological view in Figure 3.26 along two different image columns. The left-hand side chart in Figure 3.27 is derived by cutting through a column in the rear part of Figure 3.26, i.e., right-hand side in Figure 3.23, i.e., low scattering. The right-hand side chart in Figure 3.27 is obtained from a cut along a sample column in the front of Figure 3.26, i.e., left-hand side in Figure 3.23, i.e., high scattering. In these charts the x-axis corresponds with the image row, the y-axis represents the gray value of the pixels in the differential image. It becomes apparent that the heights of laser caused peaks in both cases do not differ very much. However, the differences between situations with scattering and without scattering can still clearly be noted taking into account the behavior of the curves at the foot of the laser peaks [138].

Profile cuts, such as shown in Figure 3.27, are the basis for the line detection. The lines are searched within their respective ROIs. Only one line segment is allowed in each ROI[1]. The line segments are searched by applying a threshold. Thanks to the high grade of normalization of the differential image because of the background subtraction the line detection worked out very good applying a fixed threshold. Further, Mueller et al. have shown that a segmentation applied prior to the common Center of Gravity (COG) analysis for line detection in LP can improve the robustness of the system, particularly in case there is a lot of scattering [88]. For the MWLP prototype a COG analysis after the segmentation by thresholding was skipped in order to improve the performance of the algorithm, i.e., reducing runtime. The center of the detected line segment is assumed to be the line position and is used for calculation of the distance values of

[1]Note that both sides of Figure 3.27 each show 2 adjacent ROIs of 2 lasers.

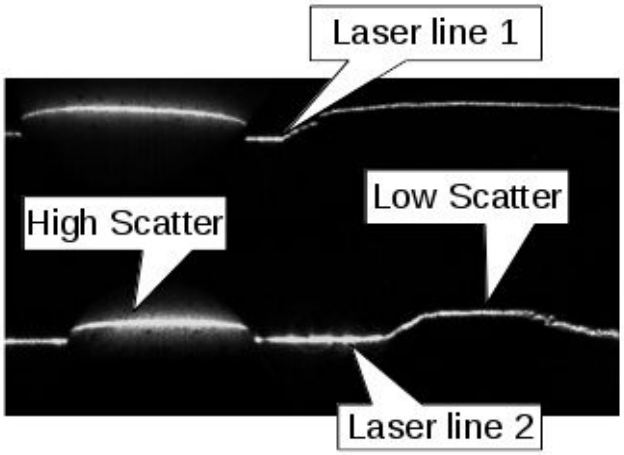


Figure 3.23: Differential image with difference in scattering highlighted [138].

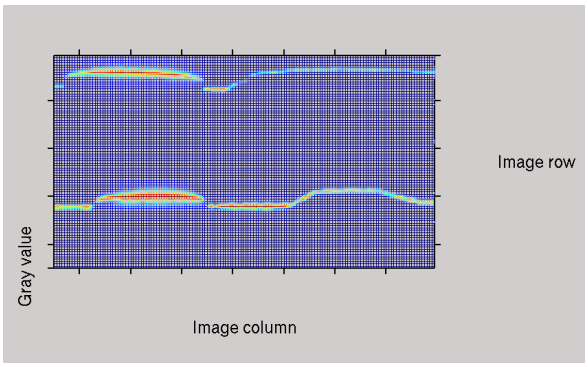


Figure 3.24: Differential image of data from excerpt shown in Figures 3.21 and 3.22 visualized as heat map. Resolution is down sampled for visualization purposes [138].

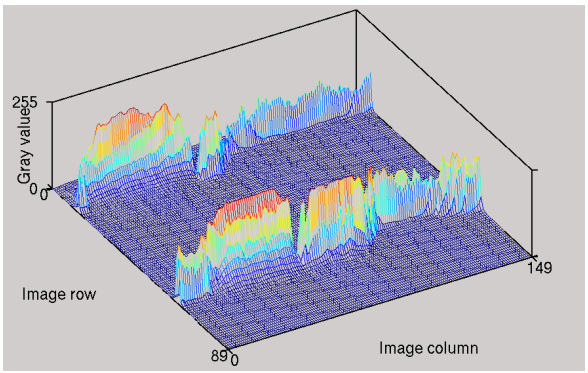


Figure 3.25: Visualization of Figure 3.25 as 3D view. The z-coordinate in the morphological view corresponds with the gray value of the respective image pixel [138].

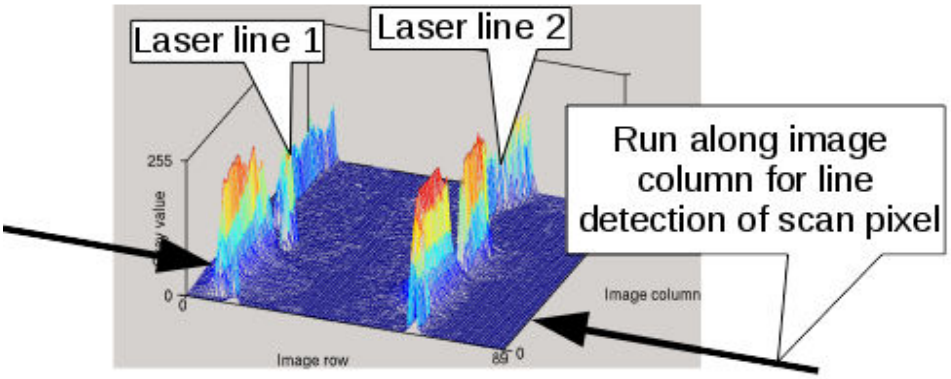


Figure 3.26: 3D view of the data with annotations. The line detection takes place by analyzing the line profiles along the image columns [138].

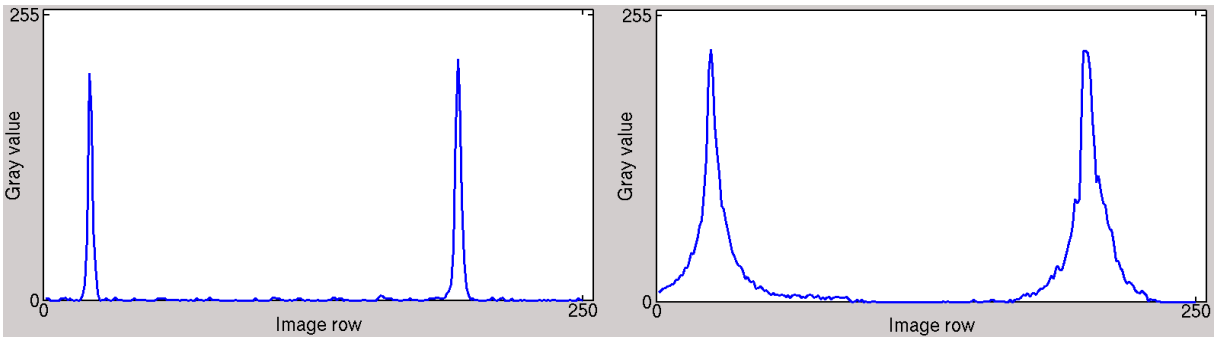


Figure 3.27: Sample cuts of the data along image columns (Left-hand side: low scattering; right-hand side: high scattering) [138].

the pixels by triangulation. As the profile cuts are more-less symmetric (cf. Figure 3.27) the differences that would be induced by an additional COG evaluation after the segmentation are negligible [138].

3.4.2 Feature extraction

In order to obtain image-based information on laser reflexion and backscattering for all attached lasers it is necessary to extract numerical features from the profile cuts along the image columns for each scan pixel of the MWLP data. These numerical features can then later on be scaled and visualized in an image-based manner.

Hereby, the high extraction rate has to be taken into account. If the MWLP prototype is operating at 100 fps camera frame rate with 3 line lasers and full camera resolution of 2048 image columns, up to 614400 of such feature extractions will have to be performed per second (parallel to laser/camera control, image acquisition and all other image processing steps). It was refrained from fitting a model for describing laser light backscattering, such as the Gausian-Lorentzian Cross Product Function (GL) which is used for analyzing scattering profiles of point lasers [76]. Due to the high extraction rate model fitting is not possible during online processing. To allow online processing only simple, primarily summing-based features have been selected for extraction. Moreover, only features are extracted that can be calculated by looping through the profile cut only once. Unfortunately, this also rules out Full Width at Half Maximum (FWHM) as it would require running along the image column twice (Once for determining the threshold and a second time for applying the threshold). The line detection and determination of the line width are conducted using a fixed threshold. As mentioned above, this is possible thanks to the good normalization of the image by the background extraction, which cuts out influences of ambient light [138].

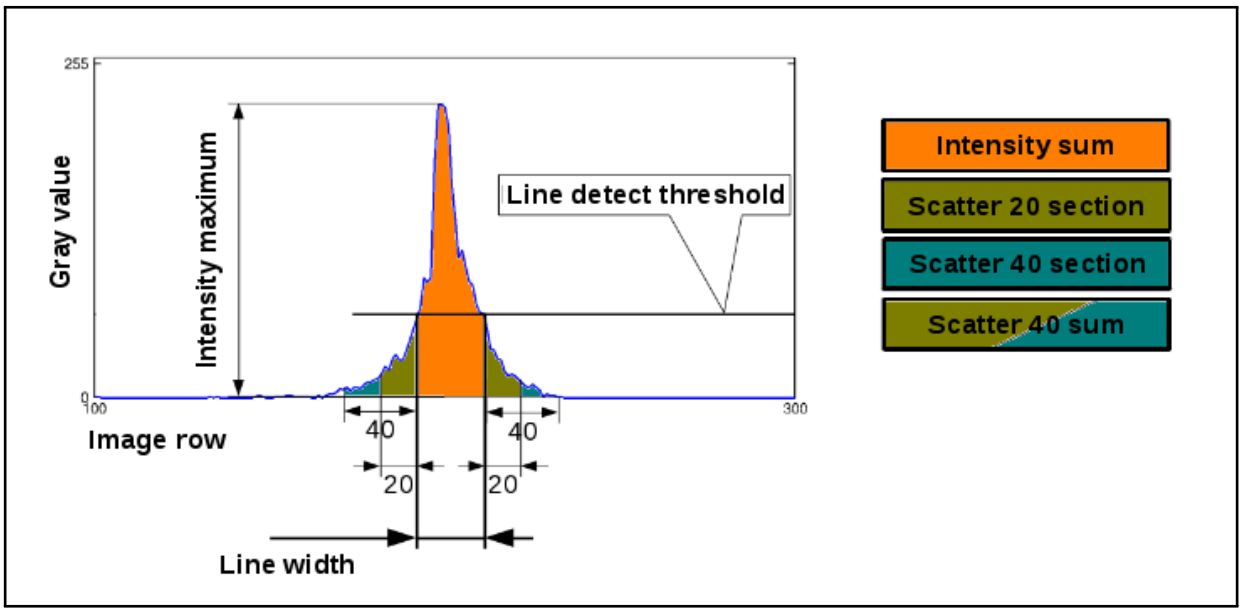


Figure 3.28: Descriptions of the features extracted from the laser lines [138].

The feature extraction is shown in Figure 3.28. It is performed by the system on every laser line and each image column of all incoming images. Figure 3.28 shows profile cut for a single laser line. It is analog to those profile cuts plotted in 3.27. The extracted features are noted.

As mentioned, during the line detection first a threshold is applied. The crossing points of the gray value curve with the line detection threshold are the line beginning and line end. As stated, their center is assumed to be the line position for triangulation. Now, the feature extraction follows. The descriptions of the extracted features are given in Table 3.10. The *Line-Width* is defined as difference between the line beginning and line end. Further, the laser reflexion can be described by the *Intensity-Maximum* (maximum gray value of the line) and the *Intensity-Sum* (sum of pixel's gray values over the line width). For estimation of the scattering of the laser lines the *Scatter-...* features are intended. They sum up pixels values in different distances before line beginning and after line end rather than between them. All described features can be obtained by a single loop run. In principle - if required - a fitting of a GL could then be done later on only on very relevant parts of the captured and stored scan data in offline processing. However, during online processing the computational effort for feature extraction is, hereby, reduced.

Table 3.10: Numerical features extracted by the MWLP system for each laser line pixel [138].

Feature	Description
Line-Width	Distance between the detected line beginning and line end (Crossing points of the gray value curve along the image column with the line detection threshold)
Intensity-Maximum	Maximum gray value of the laser line in the differential image
Intensity-Sum	Sum of pixel's values summed up between line beginning and line end, i.e., over the *Line-Width*.
Scatter-20-Section	Sum of pixel's values summed up in the column of the differential image between the 1st to the 20th pixel before the line beginning and the 1st to the 20th pixel after the line beginning.
Scatter-40-Section	Sum analog to *Scatter-20-Section* with distances between 21 and 40 Pixel (pel)
Scatter-40-Sum	Sum analog to *Scatter-20-Section* with distances between 1 and 40 pel, i.e., equals *Scatter-20-Section* + *Scatter-40-Section*
Scatter-60-Section	Sum analog to *Scatter-20-Section* with distances between 41 and 60 pel
Scatter-60-Sum	Sum analog to *Scatter-20-Section* with distances between 1 and 60 pel, i.e., equals *Scatter-40-Sum* + *Scatter-60-Section*
Scatter-80-Section	Sum analog to *Scatter-20-Section* with distances between 61 and 80 pel
Scatter-80-Sum	Sum analog to *Scatter-20-Section* with distances between 1 and 80 pel, i.e., equals *Scatter-60-Sum* + *Scatter-80-Section*

3.5 Visualization of MWLP sensor data

This section shows how the numeric sensor data obtained by the MWLP system and described in Section 3.4 is visualized to provide the user meaningful representations of it. Thereby, at this point a generic visualization is intended, not geared toward a specific application but only optimized for the sensor system.

3.5.1 Value scaling

For image-based visualization of the MWLP sensor data the extracted feature values have to be scaled. Image visualization is typically performed with 8-Bit pixel values, i.e., the allowable dynamic range of the pixel values is from 0 to 255. However, most of the extracted features mentioned in Table 3.10 have larger dynamic ranges. E.g., the *Intensity-Sum* and the *Scatter-...* values are stored in `uint16_t` variables, i.e., can theoretically range from 0 to 65535. Often times not this entire range is required for them. But still the used dynamic ranges are significantly greater than 256. Hence, it is necessary to scale them for image-based visualization. The scaling

is performed linearly between selectable lower and upper borders. Values situated beyond those borders are saturated. This is denoted in Formula 3.2.

Formula 3.2: Linear value scaling to 8 Bit range with saturation.

$$ScaledValue = \begin{cases} 0 & , if\ OriginalValue < LowerBorder \\ 255 & , if\ OriginalValue > UpperBorder \\ (\frac{OriginalValue - LowerBorder}{UpperBorder - LowerBorder}) \cdot 255 & , otherwise \end{cases}$$

To provide meaningful views of the data, thereby the scaling borders have to be selected properly depending on the real - not theoretic - dynamic range of the respective feature. This can be done manually or automatically. The automatic version analyzes the data to be visualized. In order to avoid the scaling borders to be specified depending on crude outliers a configurable amount of samples is saturated on both ends. I.e., if e.g. 5 % are to be saturated the 5 % percentile will be selected as lower border and the 95 % percentile will be selected as upper border for the scaling.

3.5.2 Pixel colorization

For colorizing pixels using one or more features extracted by the MWLP system three modes are possible. A single feature can be visualized as gray scale or as heat map and multiple features can be colored together overlaying different colors assigned to them.

- **Single feature channel as grayscale**

For gray scale visualization of a single feature of the MWLP data, simply the *ScaledValue* of the respective channel and the respective pixel is written into the all channels of the pixel of the out coming image in Red Green Blue (RGB) color space, as Formula 3.3 states. However, the MWLP data can contain invalid pixels, e.g. due if the laser line was shaded for triangulation. These are represented in the data using magic values. In order to distinguish invalid pixels in the out coming images these are drawn in color rather than gray, e.g. red pixels signal that the line detection failed to detect a line at their position.

Formula 3.3: Gray scale colorization.

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} ScaledValue \\ ScaledValue \\ ScaledValue \end{pmatrix}$$

- **Single feature channel as heat map**

For visualization of single feature channels as heat map, the Hue Saturation Value (HSV) color space was used. The HSV color space encodes color information of a pixel using cylindrical coordinates by the three components Hue, Saturation and Value. The Hue represents the rotation angle and specifies the color.

The conventions used here are the same like the OpenCV function `cv::cvtColor` specifies for 8 Bit images. It assumes the range of the S and V components is from 0 to 255. The H component is measured in degrees, i.e., goes from 0° to 360°. However, to fit into the 8 Bit range the degree value is divided by 2. Hence, the actual H range is from 0 to 180 [146].

For visualization as heat map the minimum *ScaledValue* is mapped to a blue color. Blue is represented by the hue angle of 240°, i.e., 120 following OpenCV conventions. The maximum *ScaledValue* is mapped to red color. The hue angle here is 0°, remains 0 following OpenCV conventions. The *ScaledValue* in between are mapped to the H component linearly between this. S and V components of the HSV color space are set to 255 for heat map visualization. This is denoted in Formula 3.4. Invalid pixels of the system are drawn white into the heat map image to distinguish clearly from the saturated colors of the valid heat map pixels.

Formula 3.4: Heat map colorization.

$$\begin{pmatrix} H \\ S \\ V \end{pmatrix} = \begin{pmatrix} (-1 \cdot ScaledValue \cdot \frac{120}{255}) + 120 \\ 255 \\ 255 \end{pmatrix}$$

- **Multiple feature channels overlayed**

Apart from visualizing single channels the system also offers the possibility to colorize multiple channels overlayed in different colors. This, e.g., allows generating natural RGB views in case lasers at wavelengths 405 nm (i.e., blue), 532 nm (i.e., green) and 650 nm (i.e., red) are mounted to the MWLP system.

In this case, the colorization outcomes of all selected feature channels are summed up. The colorization outcome of a single channel is thereby determined by multiplying an optional weight factor with the configured color of the channel and the *ScaledValue*. This is divided by 255 to fit the 8 Bit range and summed up for all selected feature channels as noted by Formula 3.5. Invalid pixels are drawn in black color for this colorization mode.

Table 3.11: Example for colorization: Configuration that creates an RGB view.

Feature	**Laser**	***WeightFactor***	**Color**
Intensity-Sum	405 nm (e.g. #2)	1.0	$\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 255 \end{pmatrix}$
Intensity-Sum	532 nm (e.g. #4)	1.0	$\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} = \begin{pmatrix} 0 \\ 255 \\ 0 \end{pmatrix}$
Intensity-Sum	650 nm (e.g. #6)	1.0	$\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} = \begin{pmatrix} 255 \\ 0 \\ 0 \end{pmatrix}$

Table 3.11 gives an example for such a configuration[2]. The configuration could be used to created an RGB view of the MWLP data if the above mentioned lasers are mounted. However, in spite of likely being the most useful configuration this is not the only configuration possible here. Arbitrary combinations of any number of feature channels provided by the MWLP system are possible here.

Formula 3.5: Colorization overlaying multiple features.

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \sum_{selected\ features} WeightFactor_i \cdot \begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} \cdot \frac{ScaledValue_i}{255}$$

$, with$

$WeightFactor_i$	$Value\ to\ weight\ the\ selected\ feature\ i\ with\ [0.0, 1.0]$
$R_i,\ B_i,\ C_i$	$Components\ of\ the\ color\ to\ draw\ feature\ i\ with\ [0, 255]$
$ScaledValue_i$	$ScaledValue\ of\ feature\ i\ for\ respective\ pixel\ [0, 255]$

3.5.3 Configurable MWLP image visualization model

For visualization of image-based MWLP data a flexible visualization model was set up that encapsulates the mentioned scaling and pixel colorization techniques. It contains scalers for arbitrary numeric data types for scaling to eight bit range as described before. Further, it

[2]The numbers with the # mentioned in the Laser column of Table 3.11 refer to the numbers for the lasers given in Table 3.2.

allows configuring which channels should be used for the pixel colorization. The model can be used for pixel colorization of all visualization outcomes, i.e., point clouds, overlayed images and mesh visualization. Later on, it was also be used for visualization of intermediary results of the classification pipeline.

To configure the visualization model the rqt_plugin `/overlay_viz_controller` is used. Its GUI is shown in Figure 3.29. For each type of image with configurable visualization a tab is opened in this GUI. In Figure 3.29 just a single tab is shown as only the visualization of image-based MWLP data is configured. On the left-hand side a list of channels is given. Each pixel of the data contains numeric values for all features listed there. On the right-hand side at the top the contribution of the currently selected channel can be configured. The currently selected channel appears in a white box on the left-hand side. A check box has to be set in order to use this channel for visualization. All channels used for visualization appear in the colors used for their respective visualization on the left-hand side. The color for the visualization of the currently selected channel can be specified using the color picker under the use channel check box. Further, its scaling borders can be set there as well as the mentioned weight factor. For the auto scaling drop box three selections are possible:

- **No**: In this case, the manually (or default) specified borders are used for scaling.
- **Once**: The scaling borders will be specified depending on the next data chunk and then stay untouched.
- **Always**: The scaling borders will be refreshed for each data chunk.

Below the configuration of the selected channel in Figure 3.29 the configuration of the entire image can be specified. Most important here is the selection whether a single channel shall be colorized as gray scale or heat map. However, this selection does only matter if just one channel is used for visualization.

Finally, there are two buttons in Figure 3.29. The right button reset discards all changes that have been made through this GUI. The left button sends the modified configuration to all visualization models in the connected ROS nodes. Thereby, all visualization outcomes are effected, even those created in different ROS nodes. This common configuration can be seen in Figures 3.30, 3.31 and 3.32. A visualization of MWLP data gathered from scanning cookies is configured in these screen shots. As the figure shows, image visualization and point cloud representation in rviz are colorized in the same manner.

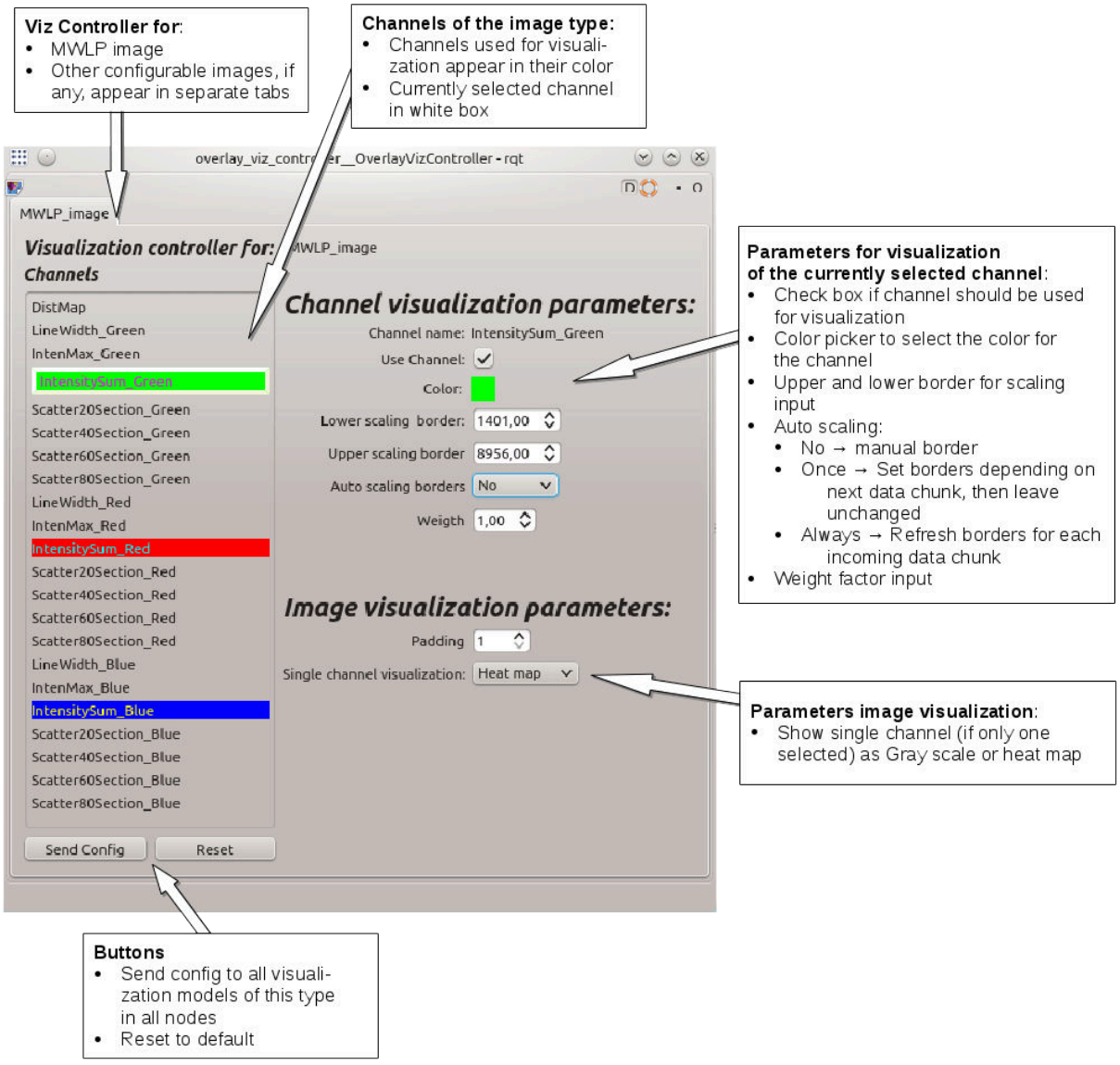


Figure 3.29: GUI of the `/overlay_viz_controller`.

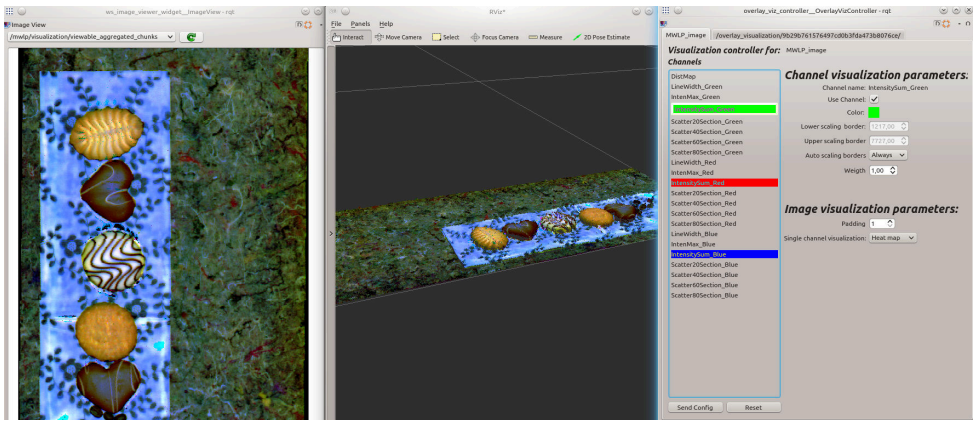


Figure 3.30: Image and point cloud colorization of scanned cookies in RGB (cf. Table 3.11).

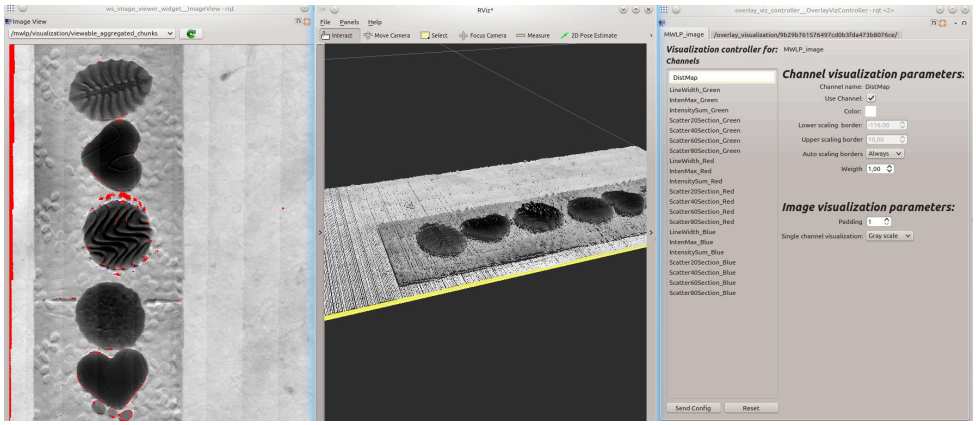


Figure 3.31: Image and point cloud colorization of scanned cookies with gray-scaled distance data.

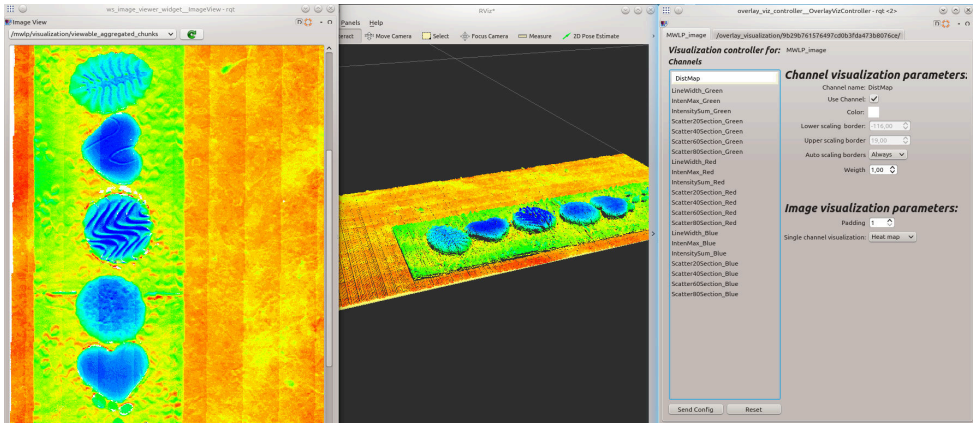


Figure 3.32: Image and point cloud colorization of scanned cookies with heatmap-like distance data.

3.5.4 Visualization outcomes

For actually showing the colorized, image-based data with overlayed distance information three possibilities are available, visualization as overlayed images, visualization as point cloud or as mesh shape. The visualization options as overlayed images and 3D point clouds have already been shown in Figures 3.30, 3.31 and 3.32. However, both of these options have drawbacks. Images for visualization can only show spectral features or distance features as depth map. Combining both in a single image view is not possible in an intuitive manner. Moreover, as stated the MWLP data may contain invalid pixels due to the triangulation principle. Representing these pixels in image visualization is only possible by stuffing the images with pixels of special colors showing invalid data, such as the red pixels in Figure 3.31. In contrast, for point clouds such pixels can be skipped, i.e., they are not drawn at all. However, when zooming into a point cloud even between neighboring points there might be an uncolored distance. I.e., by having a point cloud visualization it can hardly be seen whether there is a point missing between two adjacent points or the distance between them is their 'natural distance', due to the sensor configuration.

To overcome these drawbacks a mesh-based visualization is provided, too. The mesh-based visualization connects neighboring pixels/points in the organized point cloud using triangle faces. However, only those neighboring points where no points are missing in between are connected. I.e., the length of the triangle faces projected to the x-y-plane cannot exceed the 'natural pixel-distance', due to the sensor configuration. Visualization in this manner allows showing the distance information along with colorization to the user - as a point cloud view does - and further allows him exactly identifying poses where points are missing, e.g. due to shading.

In order to derive a mesh-based view of the captured point cloud data surface reconstruction was necessary. For surface reconstruction the Marching Cubes [74] algorithm "is the de-facto standard" [160]. It was considered and initially tested in the version implemented by the Las Vegas Reconstruction Toolkit [161]. However, the algorithm does yet approximate the surface, thereby partly interpreting the data. This interpretation of the captured sensor raw data may suit - depending on the parameters - for one application but may not suit in a different situation. Hence, interpretation of the raw sensor data is not intended at this point because in some situation it may let the sensor data appear less valuable than they are. It is intended to show the sensor data to the user as raw as possible but as descriptive as possible. This is achieved by a mesh generator that just connects the adjacent pixels of the organized cloud data without approximation of the surface. The reconstructor `pcl::OrganizedFastMesh` [49] of the `surface` module of the Point Cloud Library PCL [123] was found to do so by creating triangle meshes. Further, it is relatively fast [49] and does not require an estimation of normals for generating the mesh.

The generated mesh data can be colorized with the same visualization model like image data and Point Clouds before. In order to show it to the user, it can be rendered with the PCLVisualizer. However, `rviz` was the main tool for visualization used throughout this work as it is deeper integrated into ROS and supports listening to `tf` [28] for referencing the positions of different

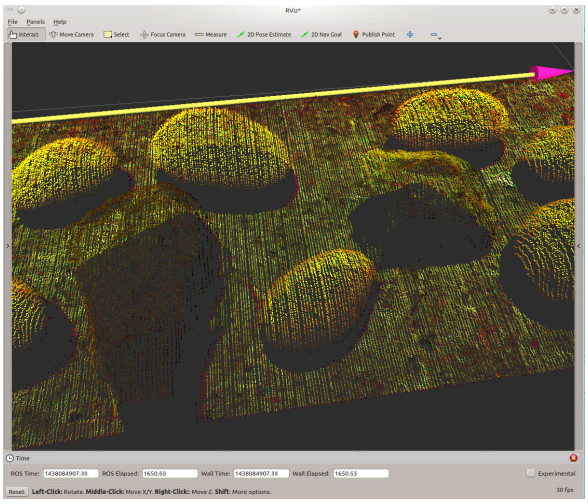


Figure 3.33: Zoom into a Point Cloud visualization of scan data from potatoes and stones.

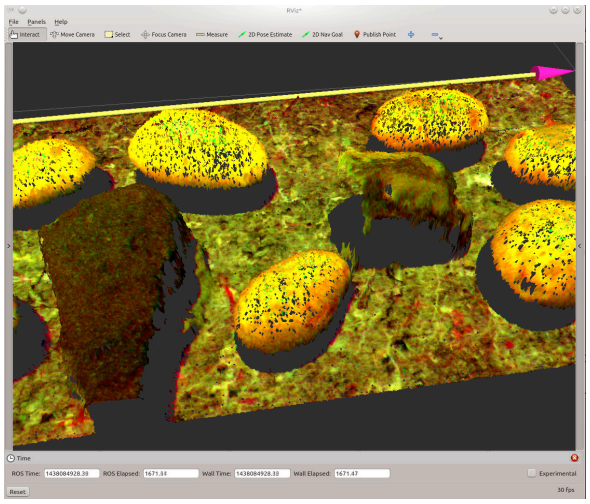


Figure 3.34: Visualization of the same data as in Figure 3.33 at the same resolution as mesh shape in `rviz`.

3D objects [47]. To provide a mesh view also in `rviz` an `rviz`-plugin was implemented for showing the mesh based data. Figure 3.33 shows a zoom into point cloud visualization of data from scanning potatoes and stones. For comparison, Figure 3.34 shows the same data rendered with the same resolution from almost the same view point but visualized as mesh shape generated with `pcl::OrganizedFastMesh`. It is clear that many textures, which can be seen in the mesh-based representation, cannot be identified in the Point Cloud view. Moreover, in the mesh view some missing points at the smooth surface of the potatoes can be identified. I.e., likely some minor adjustments of the sensor parameters could still improve the result. This issue cannot be noted when relying on the point cloud representation.

3.6 Calibration of the MWLP system

In order to use the MWLP system its image pipeline as described in Section 3.3 a number of different calibration steps is required, previously. These steps are described in this chapter as well as the concept for persistence and use of the created calibration set of the MWLP system in a distributed system architecture.

3.6.1 Calibration steps

This section describes the individual calibration steps of the MWLP system. A couple of GUI screen shots of the different calibration steps are shown in order to illustrate the different steps. Note that the system still lacks a common calibration GUI and the GUI views are still in flow. The screen shots show very basic implementations created using OpenCV's highgui module. Nevertheless, the calibration logic, data set and procedure are stable and have shown to work well. Hence, a refactoring of the GUI and creation of a common calibration GUI would cause

the views in the screen shots cause changing slightly. However, the underlying logic being in focus here would stay as it is.

- **Calibration step 0: Camera calibration (skipped)**

When using image data from cameras for image processing purposes, it is common practice and usually required to calibrate the camera at the beginning using a chessboard pattern and then rectify all incoming images during operation in order to correct lens distortions using the pin-hole camera model [43]. This is particularly indispensable when using relatively cheap USB web cams of consumer sector. With their cheap optics and small imagers the images are otherwise unsuitable for automated image processing. Unfortunately, this step of image rectification and correction of lens distortion is requires quite some processing power, if the images are coming in at high frame rates.

However, in this case, industrial grade lenses are used and the camera comprises a relatively huge 2/3" imager. Therefore, and because the processing capacities at the high image frame rate are a critical point for online processing with the MWLP system, it was tested which impact the conventional camera calibration really has for this configuration.

For testing this the camera was first calibrated using package `camera_calibration` provided along with ROS [84] and a chessboard pattern. The lens mounted to the camera for this test was of type Pentax RICOH FL-CC3516-2M (Focal length 35 mm; Aperture F1,6). Same as always used for the MWLP system during this work, the camera of type Baumer HXG20NIR (cf. 3.1) was used for this test. The object distance was approx. 60 cm.

After the camera was calibrated, the derived calibration model was applied to correct a set of images captured with the camera. Samples of the collected raw images are shown in Figures 3.35 and 3.38. Figure 3.35 contains a shot of a chessboard pattern in front of the camera, Figure 3.38 shows a shot of potatoes with the slight background illumination of the MWLP system applied for matching (lasers are turned off). Figures 3.36 and 3.39 depict the images after rectification. There are hardly any differences notable between the image before and after rectification. Figures 3.37 and 3.40 show the difference images obtained by subtracting raw image and rectified image from each other using `cv::absdiff`. In the case of the chessboard there are only some minor fragments visible among the edges of the chessboard field. In the more realistic case with the potatoes (Figures 3.38, 3.39 and 3.40) and the background illumination of the MWLP system, there is next of nothing visible in the difference image. This means in both cases the raw image and the rectified image are practically equal.

This was further observed and quantified for a set of captured images. In order to measure the impact of the image rectification the Sum of Absolute Differences (SAD) method was applied again, similarly to what has been described for image matching (cf. 3.3.3). But here no pixels where skipped for evaluation, the resulting SAD value is just Sum of Absolute Differences of both images normalized by the sum of both images. This was applied for the comparison the raw image with the rectified image. The average of the obtained SADs for a set of 1886 images

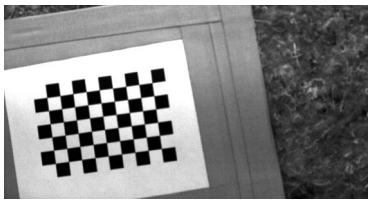


Figure 3.35: Collected raw image of chessboard.

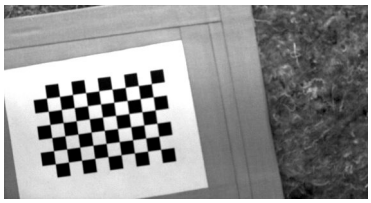


Figure 3.36: Image of Figure 3.35 after rectification.

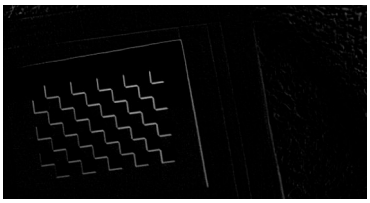


Figure 3.37: Difference image of Figures 3.35 and 3.36.

Figure 3.38: Collected raw image of potatoes with MWLP background illumination.

Figure 3.39: Image of Figure 3.38 after rectification.

Figure 3.40: Difference image of Figures 3.38 and 3.39.

Table 3.12: Impact of the image rectification.

Number of images	Mean SAD	Standard deviation	Min SAD	Max SAD
1886	0.0653	0.0227	0.0315	0.119

was 0.0653, as Table 3.12 shows. This means the images do not differ relevantly. To make this value comparable it can be stated, that for the SAD-based matching as described in Section 3.3.3 minima with values less than 0.12 can typically be trusted to be good matches.

Further, the runtime of the image rectification was measured during this test. The results are given in Table 3.13. The average runtime for the set of 1886 images was 11.8 ms per image. This means the step is not absolutely infeasible for processing with the MWLP system. It has similar computation cost like the processing step line filtering, which also runs on the CPU of the MWLP system. However, as the impact of the camera calibration is very limited as shown and the computational costs are notable, it was skipped for most tests described throughout this dissertation (unless otherwise stated in the following chapters). The reduced computational effort was preferred here. The design of the ROS `image_pipeline` would still allow to plug the image rectification in, if this was required for in another lens configuration, though.

Table 3.13: Runtime of the image rectification.

	Number of images	Average time per image	Standard deviation
CPU	1886	11.8 ms	1.49 ms

- **Calibration step 1: Mounting calibration**

As the description of the previous calibration step was just a description of the circumstances that led to the decision to skip it, this is the first calibration step that actually has to be done by the user in order to calibrate the MWLP system. It is required for running the processing step of mounting correction (cf. 3.3.2). In order to perform this calibration step, a chessboard pattern has to be laid flat onto the conveyor, i.e., in the object plane of the conveyor as described in Section 3.3.2.

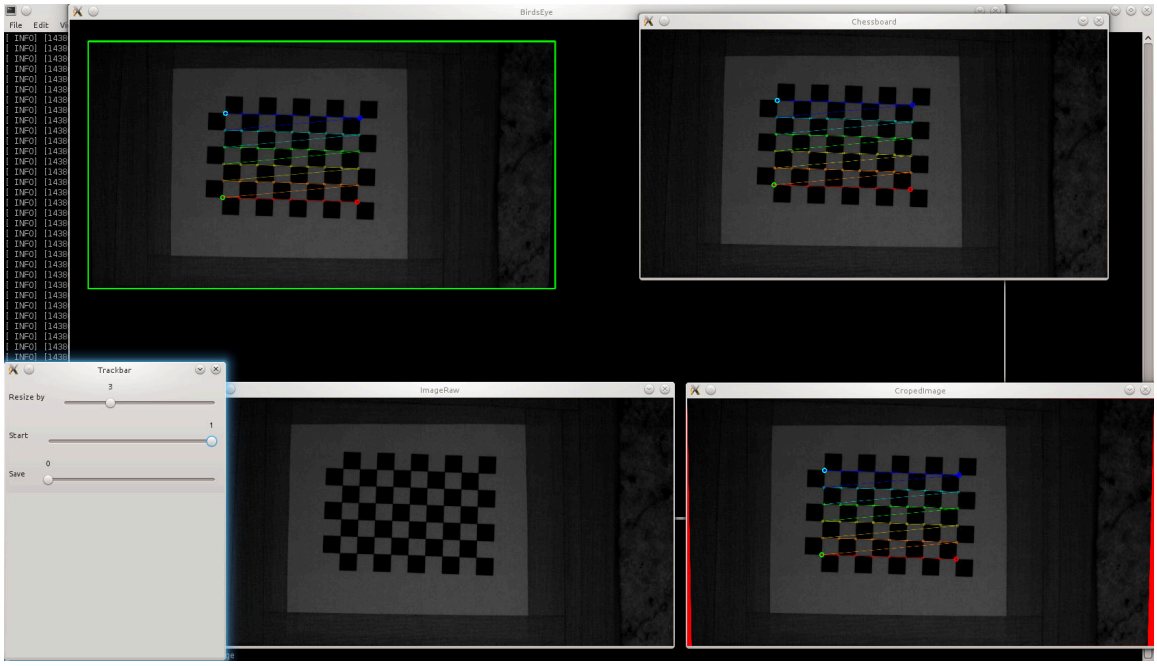


Figure 3.41: Screen shot during mounting calibration.

For running the mounting correction the camera driver and the lighting controller have to be started. However, the camera runs at a reduced frame rate of e.g. 10 fps. Then, the mounting calibration process can be started. It configures the lighting controller such that the LEDs for background illumination are turned on and the lasers are turned off. Further, it subscribes to the camera raw image. It opens a number of windows as shown in Figure 3.41. The incoming raw camera image can be seen in the bottom left window of Figure 3.41. As soon as the chessboard is properly shown in this window, the user can pull the 'Start' trackbar to 1 and the processing starts. First the chessboard corners are detected in the image, which leads to the image in the top right window of Figure 3.41. Based on the detected chessboard corners the homography for the perspective transformation from the projected image plane to the object plane (cf. Figure 3.12) can be estimated. The obtained homography is modified, such that no rotation around the z-axis is modeled. Further, the homography is optimized, such that least scaling is applied. This is done in order to prevent loss of data as well as avoiding unneeded processing of interpolated data. Finally, the image is cropped (Top left window in Figure 3.41). The result of processing

the image with the derived homography is shown in the bottom right window of Figure 3.41. In this window the real transformation is shown, as it would be conducted on the incoming images in the processing step of mounting correction during online processing. For inspection and clearer visualization, the chessboard corners are also drawn into this transformed image and the stuffed pixels that cannot be mapped during the perspective transformation are colored red (During online processing the images are stuffed with zeros, i.e., black pixels). As the user sees the result is appropriate, he can pull the 'Save' trackbar to 1 and the current homography and crop window are saved to the calibration.

- **Calibration step 2: Rotary encoder calibration**

The next step after the mounting calibration has been performed is the calibration of the rotary encoder. For using the rotary encoder data as initial guess for the matching it has to be calibrated how differences in the count of the rotary encoder typically correlate to the image shifts as outcome of the matching.

To calibrate the rotary encoder the camera and light controller stay active. Same as for the previous step, background illumination is turned on, while the lasers are switched off. Now the user has to record a number of images with rotary encoder pose stamps while the conveyor is moving - forward, backward and different speeds.

These images are used for calibration. The previously calibrated mounting correction is applied to the images. Next, images captured sequentially are matched with each other using the same matching procedure as described in Section 3.3.3. However, here a previous line filtering is not necessary here as the lasers are turned off. Further, unlike in Section 3.3.3 an initial guess is not known for the matching, i.e., the matching must happen globally rather than locally. This means this calibration step may take a couple of minutes. But this is not as critical as during online processing. The matched image shifts are then correlated to the ticks counter of the rotary encoder by least-squares fitting of a linear regression. However, a difference in the ticks counter of zeros, i.e., no movement, should during calibration safely cause an image shift of zero. Therefore, the intercept of the linear model can be skipped, i.e., the fitted function is like $y = ax$ rather than $y = ax + b$. After all images are matched, the derived factor is saved to the calibration.

- **Calibration step 3: Line laser ROI setting**

The next following calibration is the specification of the laser's ROIs. In accordance with the description in Section 1.1.2 (cf. Figure 1.8) a dedicated ROI must be known for each laser active on the MWLP system. These ROI are specified here.

For setting of the ROIs camera driver and lighting controller nodes are running. Now the process for setting the ROIs can be started. There are different versions of this executable, of which each can be used for setting the ROIs for a different wavelength configuration, e.g. the RGB-setter is used for setting the ROIs in a configuration of the MWLP system with lasers #2, #4 and

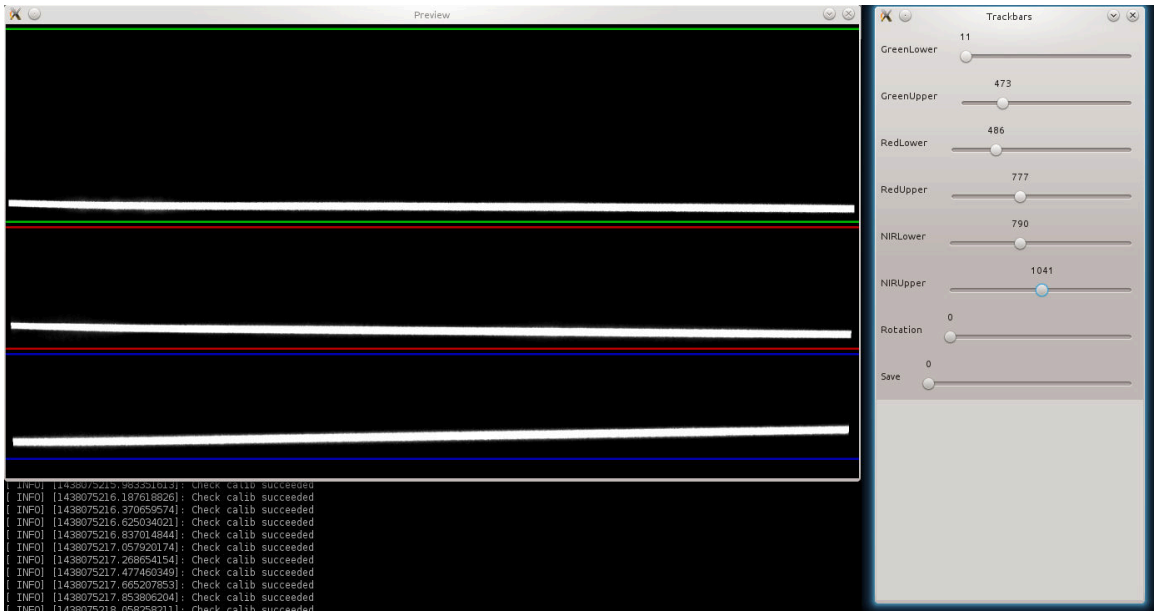


Figure 3.42: Screen shot of the laser ROI setter.

#6 (cf. Table 3.2). The process of setting the ROIs configures the lighting controller, such that the LEDs for background illumination are turned off and the respective lasers, the ROIs shall be configured for, are turned on.

As the process is started and the view is configured the user may place a blank calibration target beneath the MWLP system flat on the conveyor such that it fills the entire FOV of the camera. This target must be flat and have a constant Lambertian reflection behavior throughout its camera facing surface, ideally with same level of reflection for all monitored wavelengths. Ideally, calibration targets such as Spectralon panels provide such characteristics [39] and would be used here. However, if lacking a Spectralon calibration target of proper size using a blank sheet of paper laying flat on the conveyor has also provided good results.

Figure 3.42 shows a screen shot of the ROI of the camera. In the 'Preview' window at top left the subscribed camera image of the laser lines (here config with green (#2), red (#6) and NIR (#8) lasers) on the blank calibration target (here sheet of paper) with LEDs turned off can be seen. The Mounting correction (cf. calibration step 1) is performed before the image is shown. Further the boundaries of the laser ROIs are drawn into the image view (colored horizontal lines in top left window of Figure 3.42). The user can now modify the ROIs by moving the respective trackbars on the right-hand side. The ROIs must be specified by the user such that they do not overlap, upper and lower boundary of each ROI do differ and each of the lasers is completely inside a single ROI. Further, no high pixel values (exceeding the line detection threshold, cf. Section 3.4) may be found in the image parts not belonging to a ROI. If any of these conditions is not met, the image view will be overlayed red (cf. Figure 3.43), thereby indicating that the current ROI settings are improper.

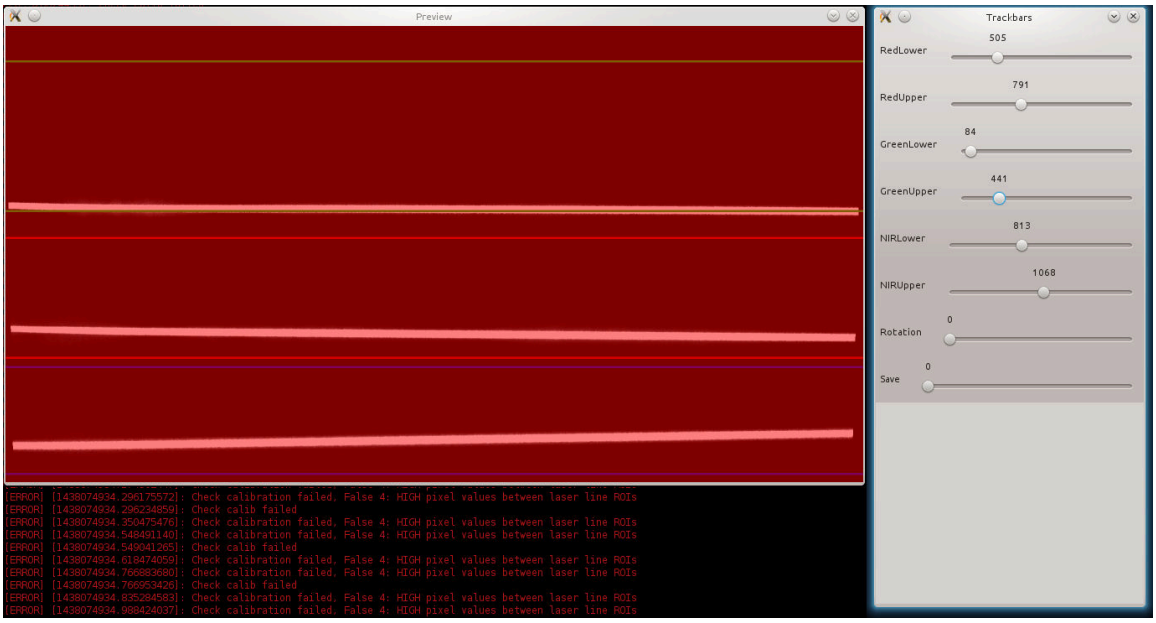


Figure 3.43: Screen shot of the laser ROI setter if the specified ROIs are obviously not correct.

Figure 3.42 shows an ROI selection as it should be. The line lasers are here reflected by a blank sheet of paper that lays flat on the conveyor. Therefore, it can be assumed, that the object distances will not grow much more, objects on the conveyor will only cause the measured distances to decrease. This means that all laser lines during practical measurements will not move much in the bottom direction of the image. Therefore, the bottom ROI boundaries can be relatively close to the respective laser lines. If there are objects on the conveyor, meaning smaller measured distances the laser lines will move closer to the top in the image. Hence, the top ROI boundaries are relatively far away from the laser lines. Further, there are small parts of the image at top and bottom and between the ROIs not belonging to any ROI. These can be used for additional trust indication during online processing, as there may not appear pixels exceeding the line detection threshold in these regions. They can be used to detect that the system's MR is exceeded. Distances of e.g. 10 pel between the ROIs have approved to be a good choice here. Finally, in Figure 3.42 can be noted, that the top laser line has the hugest ROI while bottom laser line has the smallest. This is due to the configuration. The laser inducing the top laser line has the hugest mounting distance from the camera and, thus, the triangulation angle of the thought optical axis centering its ROI is the hugest. I.e., an object of constant high measured by all these lasers will cause the top laser line to differ by the more pixel than the laser line in the center, which in turn differs more than the bottom laser line if measured in pixels. Consequently, targeting a similar measurement range for all lasers the ROI of the top laser line must be the hugest. In turn, the top laser has also the best distance resolution. Therefore, during online processing, if after the lines are assembled and the scan check is passed, the distance value of the top laser line will be preferably used as distance value of the MWLP scan pixel. Further, the mounting order was picked due to this issue. Here, the top laser is

the green one (@532nm). This is because, as stated before, laser light mainly in the wavebands between 600 nm and 1000 nm is scattered much by organic tissue [64]. The scattering can cause problems when using industrial LP systems at 650 mn for scanning plants [101]. However, the green laser is scattered less and therefore expected to provide the best results with distance measurement. Hence, it is mounted as top laser with the best theoretical distance resolution and is preferably used for the distance measurement. The distance values derived by the other lasers during online processing will be used for the scan check (cf. Section 3.3.5) and all value will be compared with each other, but if a pixel with the green laser passes, the value of the green laser will be used as distance value.

After the user has selected the ROIs like shown in Figure 3.42 using the respective trackbars on the right-hand side, he can pull the 'Save' trackbar to 1 which causes the process to save the current ROI setting to the calibration file and then terminate itself.

- **Calibration step 4: Prepare calibration of line detection and assembly**

After the laser's ROIs are specified for calibration step 4 the user may record a small number (e.g. 10) of images with the same setup as before during step 3, i.e., background illumination turned off and the selected laser turned on with camera FOV facing a blank reflection target.

The recorded set of images is then partly processed. The mounting correction is applied (cf. calibration step 1). Next, the line detection in the previously specified ROIs takes place. However, unlike during online processing it is assumed that thanks to the setup with blank target and background illumination turned off the line detection does work without any preprocessing, i.e., the matching and differential image parts of the online processing are skipped. Finally, the detected lines are drawn into the images with applied mounting correction as well as the laser's ROIs and the resulting images are saved.

An example for a resulting image of this step for preparing the line calibration is given in Figure 3.44. The user may now inspect these image and approve that the laser lines are detected correctly. The detected laser lines are the input for calibration step 5. In principle, the calibration steps 4 and 5 could be merged to a single step. However, the steps are separated to allow the user inspecting and approving the detected lines in order to assure a correct calibration. If he can approve the detected lines and ROIs, he may proceed with calibration step 5.

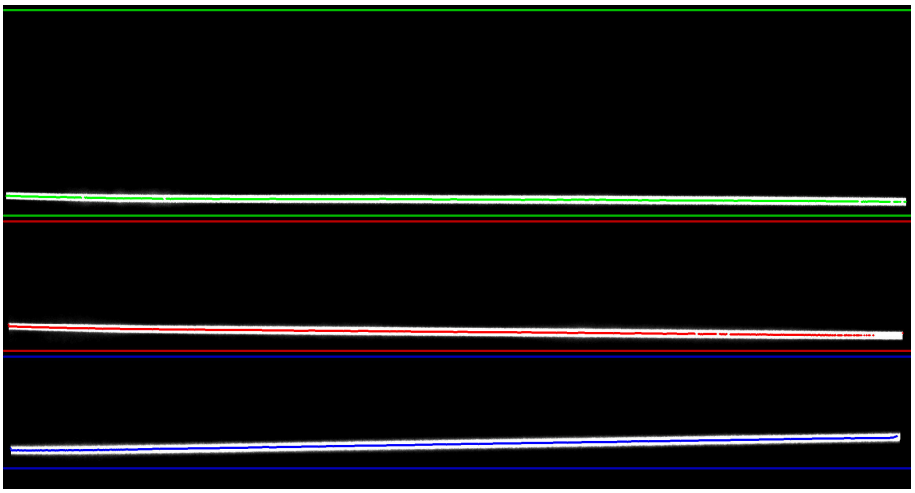


Figure 3.44: Example for a result image of calibration step 4.

- **Calibration step 5: Make calibration of line detection and assembly**

This calibration step uses the lines detected in the previous step to actually modify the calibration file. The images may be derived from a flat calibration target laying flat onto the conveyor. Therefore, it is assumed that the distance between the object and the sensing head is equal throughout everything seen in the view of the camera. Now all lines detected in the previous step are fitted to a linear model by least-squares regression. Unlike during calibration step 2 the intercept is not forced to zero here, i.e., fitted model is function is like $y = ax + b$. Thereby, x represents the image column and y represents the image row where a pixel of the respective laser line approximately appears. The result of this approximation is shown in Figure 3.45. Here the fitted linear regression function is drawn in the respective colors on top of the laser lines. Note that - unlike in Figure 3.44 - the colored lines drawn on top of the laser lines are really straight lines here and not drawn pixel-by-pixel for each detected pixel.

The linear regressions derived hereby are used during online processing for two different purposes. First, they are used for correction of a possible imprecise laser mounting. With a flat calibration target, i.e., distance assumed equal throughout the camera's FOV, theoretically all laser lines should be exactly horizontal lines in these images. However, they are not as Figure 3.45 shows. Particularly the bottom-most laser line is sloped a bit. This is due to minor inaccuracies in the laser mounting. The lasers are rotated a bit around the z-axis, i.e., not perfectly aligned with the camera. This is corrected during online processing by always subtracting the values of the linear regression functions calibrated here before evaluating the object distances. Second, the linear regression functions derived here are the main calibration input for the line assembly. The line assembly uses the distances between the laser lines obtained from these regression functions and compares them to the image shifts obtained by the SAD-based matching in order to assemble multiple detected laser lines to an MWLP scan. After all detected lines in all processed images are fitted the obtained regressions are saved to the calibration file and the user may proceed with calibration step 6.

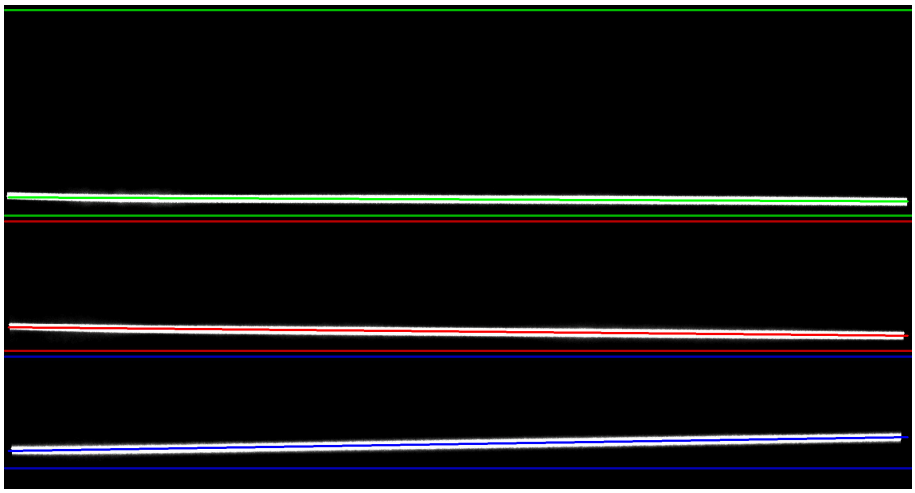


Figure 3.45: Example for a result image of calibration step 4.

- **Calibration step 6: 3D-Calibration**

After calibration step 5 is finished, all parts of the image processing pipeline of the `/line_detection_node` can be used in principle. Only the scan check of the line assembly (cf. Section 3.3.5) cannot be used. This is because - e.g. for a constellation with three lasers mounted - there are three distance values available for each pixel. But it is not known, how these values correlate with each other. Further, the values are pixel difference in the positions of the detected laser lines and it is not known, how these values correlate to metric dimensions, i.e., visualizations as Point Cloud or Mesh Shape are not possible.

To overcome this the 6th calibration step is the 3D calibration. Here a 3D calibration object with different, known height steps is scanned by the system. The 3D calibration object was designed in CAD and then sintered by a rapid prototyping system of type EOS Forminga P 110. It is a laser sintering system and has a minimum layer size of 0.06 mm. The CAD view and a photo of the sintered prototype are depicted in Figure 3.46.

Next, the 3D calibration object is scanned by the system. Unlike for the previous calibration step, here the full frame rate of the camera is required and lasers as well as background illumination are turned on, i.e., the recording setup is identical with online processing. The images are recorded while the calibration object on the conveyor is moved along-side the sensing head. The images recorded while scanning the test object, e.g. with three lasers, could look like the one given in Figure 3.47.

The recorded images are then processed through the image pipeline of the MWLP system. All steps are performed. Only the scan check of the assembled lines is skipped, as it is not yet possible. From the processed data the depth maps are created. However, here not a single depth map is created but one depth map is created for each mounted laser. Figures 3.48, 3.49 and 3.50 show the obtained distance maps in for the same constellation as it was used as example before

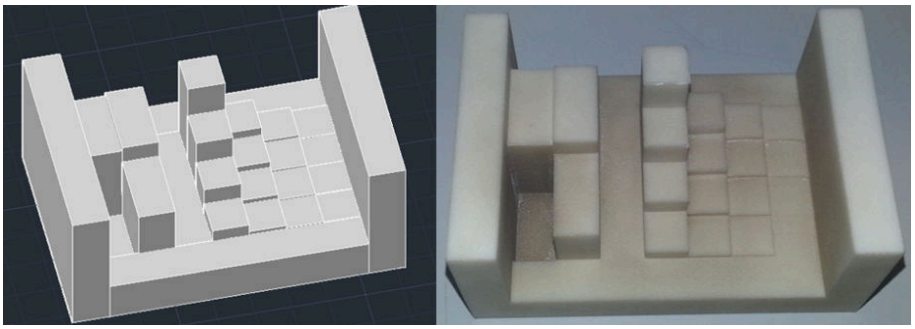


Figure 3.46: The 3D calibration object - left-hand side: CAD; right-hand side: photo of the sintered prototype.

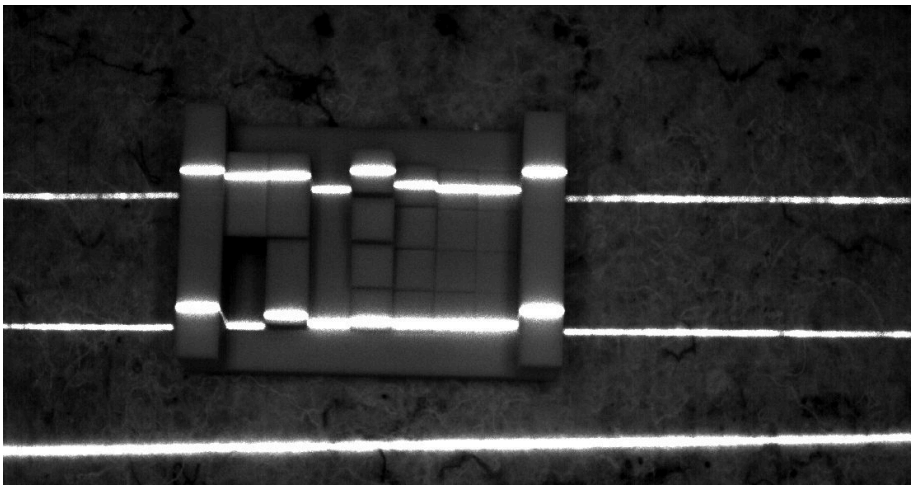


Figure 3.47: Camera raw image while scanning the 3D calibration object with three line lasers.

when the ROI setting and the line calibration have been described (I.e., lasers #2, #4 and #6 are mounted.). The structure of the test object can clearly be noted in all of these images. The differences in the size of the shaded regions are due to the differences in the mounting positions. As mentioned, the green laser is mounted in the front position, i.e., appears at top in Figure 3.47. Hence, it has the maximum resolution. In turn, it also shows the hugest shaded regions among sharp edges of measured objects.

For correlating the thereby obtained distance values with the metric dimensions the known distance steps of the 3D test object are used. The 3D calibration process automatically detects the segment with the 3D test object crops and scales it. In order to safely identify the height steps within the test object user interaction is once more required. Figure 3.51 shows the GUI for this calibration step. In the center the newly captured distance map is shown in the window 'This image'. On the right-hand side a similar distance map is drawn in the window 'Comparison'. It was captured during development of this calibration step. There are polygons and circles drawn on top of both views. These polygons and circles indicate the regions where the distance values

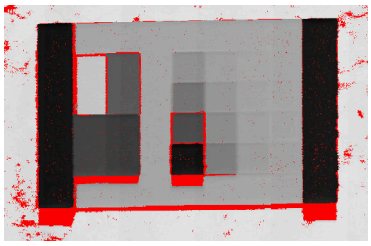


Figure 3.48: Gray-scaled distance map of the green laser.

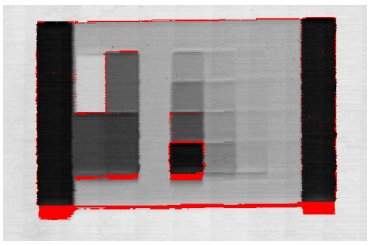


Figure 3.49: Gray-scaled distance map of the red laser.

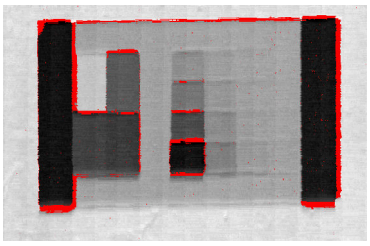


Figure 3.50: Gray-scaled distance map of the NIR laser.

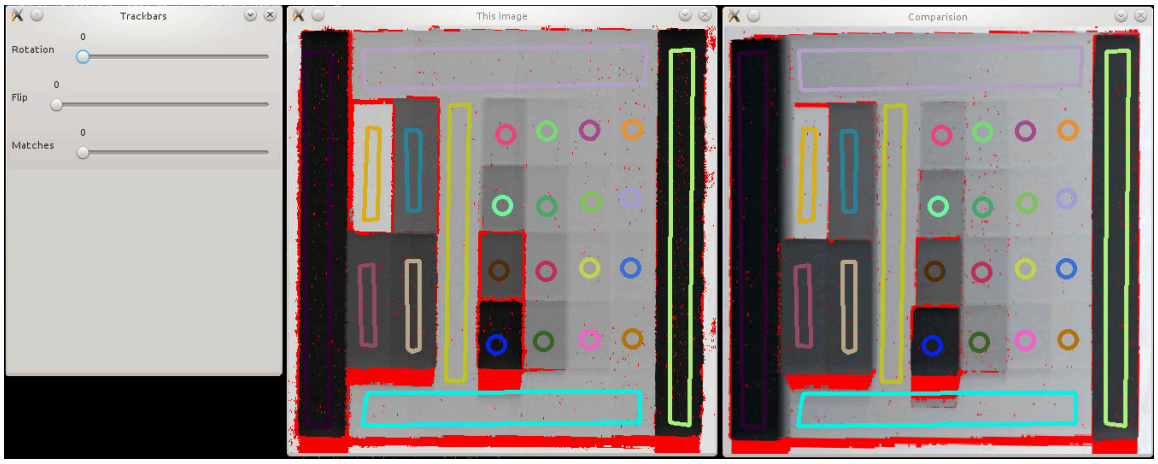


Figure 3.51: User interaction for 3D calibration.

are used for calibration. They may be completely inside the region of one height step. The user could redraw these regions for calibration each time. But as the calibration object is known, i.e., does not change and the extracted region is scaled usually they are exactly the same for all images. Therefore, the regions that have been marked once on the comparison image are drawn by default into each of such newly acquired distance maps of the 3D calibration object. However, sometimes the image has to be rotated a bit or flipped in order to let the pre-marked polygons and circles match the height steps of the calibration object. This depends on how the calibration object is oriented on the conveyor. This rotation or flipping can be done using the trackbars 'Rotation' and 'Flip' on the left-hand side of Figure 3.51. As the user sees the polygons and circles all mark the same height steps in both images in the windows 'This image' and 'Comparison' he can pull the 'Matches' trackbar and the 3D calibration for the respective laser is created and saved.

Thereby, the calibration is created by correlating all valid pixels that are inside a polygon or circle with the height steps of the 3D calibration object which are known from the CAD data. The values are again fitted to a linear model including intercept, i.e., $y = ax + b$, by least-squares regression. The resulting coefficients are saved to the calibration file for each mounted laser.

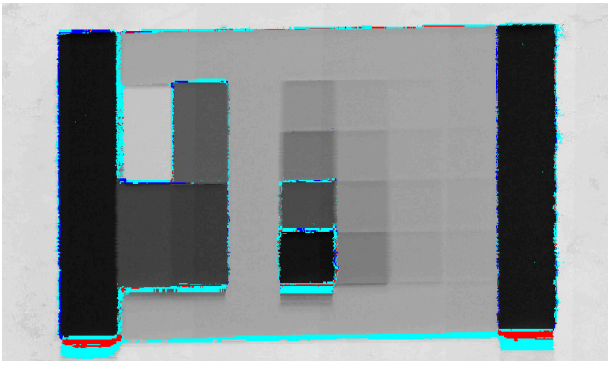


Figure 3.52: Merged gray-scaled distance map of the 3D calibration object from all lasers after scan check.

After the 3D calibration the all processing steps of the image pipeline of the MWLP system are ready to be used with the calibrated laser constellation, including the scan check, Point Cloud and mesh visualization. In order to provide an impression of the scan check, Figure 3.52 is given. It is obtained from processing the same data like shown in Figures 3.48, 3.49 and 3.50. However, here the data was processed after the 3D calibration has been done. The invalid pixels in this distance map are colored with differently. A red pixel means that this pixel could not be scanned by any laser. The light blue pixel means that there were insufficient lasers that were able to scan this pixel. Thereby, the scan check was configured such that at least two lasers had to provide distance data in order to let the pixel pass as valid. Deep blue pixel values indicate that the distance values obtained by different lasers after applying the 3D calibration differed by more than 5 mm, so the scanned values are apparently not robust. It can be noted that merging the distance information gathered from the different laser lines and applying the scan check allows to reduce the number invalid regions of the green laser (compare Figure 3.48) along with reduction of noise and crude outliers. As mentioned in Section 3.3.5, the parameters of the scan check, i.e., minimum number of different lasers providing distance values and maximum differences between these values, can be changed any time.

- **Calibration step 7: Calibration of default colorization settings**

As stated in Section 3.5.3, the upper and lower borders for scaling during the visualization of the MWLP-data can be modified manually by using the `/overlay_viz_controller` or it can be selected in the `/overlay_viz_controller` that the borders may be retrieved from the data to be visualized. However, retrieving the borders from the data is a quite expensive processing step, particularly as all the data samples (pixels) that shall be taken into account must be sorted in order to obtain the respective percentile values. Therefore, during online processing it is better to have fixed scaling borders that - of course - nevertheless can be modified or reevaluated.

This is done by the 7th - actually optional - processing step. It retrieves the scaling borders for the default colorization setting. Thereby, the default colorization setting is to sum up multiple *Intensity-Sum* values of different lasers multiplied with different colors, such as it is shown e.g.

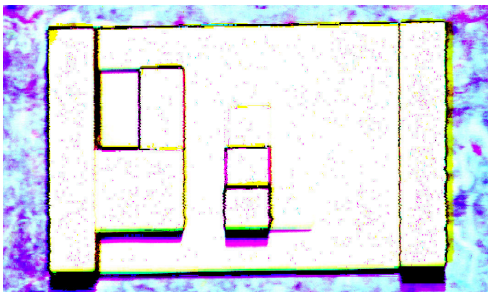


Figure 3.53: Image sample with bad selection of the scaling borders for colorization.

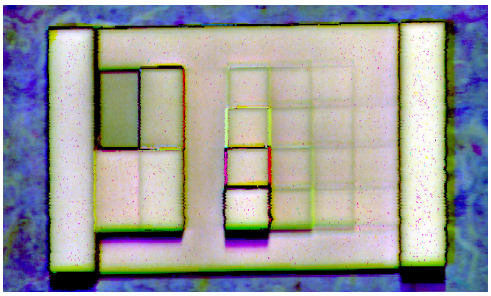


Figure 3.54: Image sample with after auto selection scaling borders for colorization.

in Table 3.11. Based on the scan data collected for the 3D calibration the upper and lower scaling borders are retrieved as described in Section 3.5.1 and saved to the calibration file as default values. As the calibration object has huge white parts these default values are typically a good starting point for colorization of other scenes as well. The values are retrieved from the data and then saved to the calibration file as defaults for this laser configuration.

To give an impression of the color calibration the Figures 3.53 and 3.54 are provided. In both the collected sample data for the 3D calibration of the laser constellation with lasers #2 (green), #4 (red) and #6 (NIR) are shown (identical scan data like shown in Figure 3.52). Thereby, in both cases the *Intensity-Sum* features are colorized, where laser #2 is colorized green and #4 is colorized in red. Laser #6 (NIR) is colorized in blue in both cases, which causes the blue sheen. Only the scaling borders for visualization differ. Figure 3.53 shows an example for a bad selection of the scaling borders. Many pixels are over saturated, thereby loosing a lot of detail in the visualization. In contrast, Figure 3.54 shows the same data after auto scaling of the borders. Clearly, a more descriptive view.

3.6.2 Calibration persistence and distribution

As the previous Section 3.6.1 describes, for calibration of the MWLP system many steps are required. Even if the conduction of many calibration steps do not take as long as the description might suggest, a calibration of the system should not be done more often than necessary. Fortunately, after the system has been calibrated the calibration must not be conducted again as long as the mounting of lasers and camera is not modified and the lens is not exchanged nor its configuration adjusted. So for many potential uses a calibration could be relatively seldom required.

In order to avoid having to calibrate the system each time, the calibration file can - of course - be saved. Extensible Markup Language (XML) is used to persist the calibration. After startup of the processing PC of the MWLP system, a ROS parameter [148] pointing the calibration file to be used must be specified. Then the launches of the MWLP system can be performed. This allows to used different calibrations on the same system, e.g. for different laser constellations.

A part from the calibrated information described, the calibration can also contain non-default, better suiting parameter values of a couple of processing parameters, such as the exposure time, frame rate and image format of the camera or adjustments of the parameters of matching or scan check.

The calibration is loaded by the `/line_detection_node` immediately after startup. In order to provide it to other nodes requiring the calibration file (and potentially running on other PCs) over the ROS network, the `/line_detection_node` sets up a ROS service server serving the calibration file - as requested - through the network. Other nodes that require the calibration have to wait for this service to come up before they can completely start up. This further assures a defined startup behavior. The main node of the MWLP system, i.e., the `/line_detection_node`, has to be up before the other nodes can become ready for processing.

Chapter 4

Validation experiments with the MWLP prototype

After the previous chapter described the hardware and software design and development of the MWLP prototype, this chapter will focus on the quality of the sensor data derived using the MWLP system, i.e., 3D range data with matched reflectance and scattering data at multiple wavelengths. Different experiments were conducted in order to show how the data performs and how the results are influenced by different processing conditions and/or modifications of the processing chain.

4.1 Sample scans

First, to give a feeling of the quality of the data this section will show a couple of sample scans derived with different laser configurations.

Figure 4.1 shows scan data of sugar beets collected using the lasers #1, #3 and #5 (cf. Table 3.2), i.e., in an RGB configuration of the MWLP prototype. Figure 4.2 depicts scan data of different vegetables. Top-left part of Figure 4.2 shows carrots, top-right European radishes, bottom left daikon radishes and bottom-center/right bananas. These scans are collected applying the lasers #2, #4 and #6, i.e., another RGB configuration. In both cases the data is visualized as 3D point clouds wherein each point of the point cloud is colorized as described in Table 3.11.

Figure 4.3 plots scan data of an apple which was collected using the red laser #6 at 650 nm and the NIR laser #8 operating at 850 nm. The view is showing gray-scaled data, wherein the depth map is given on the left-hand side and the gray-scaled *IntensitySum* data of the lasers #8 and #6 are given in the center and on the right-hand side, respectively.

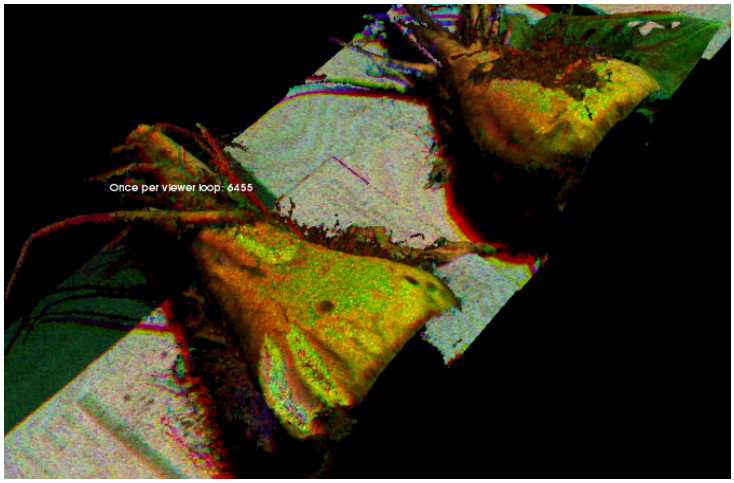


Figure 4.1: Sugar beets scanned with lasers #1, #3 and #5 visualized as point cloud [137].

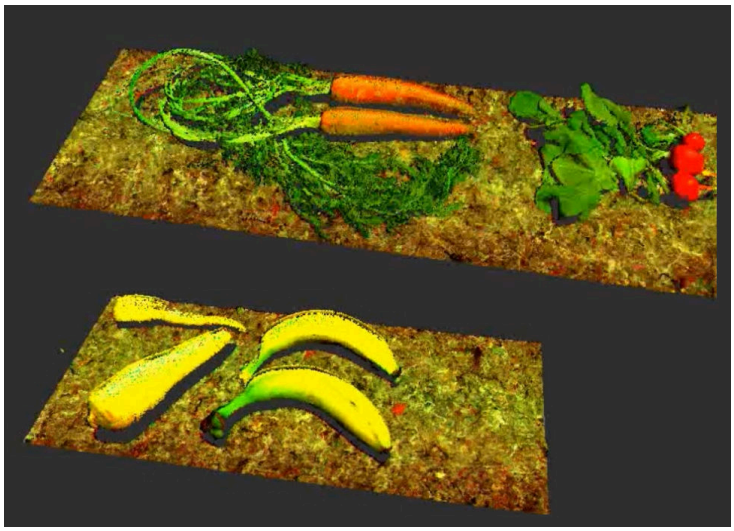


Figure 4.2: Different vegetables scanned with lasers #2, #4 and #6 visualized as point cloud.

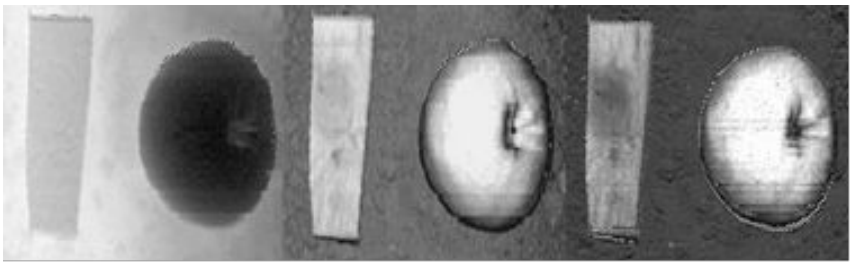


Figure 4.3: Depth map (left) and grayscale reflections of #8 and #6 for an apple and a piece of wood. Invalid pixels, e.g. shaded for triangulation, appear hatched [137].

4.2 Accuracy of the line assembly of multiple laser lines

Using the MWLP system the collected scattering and reflectance data of different lasers at different wavelengths are gathered in an SDM-manner, i.e., at a particular point of time each laser highlights a different position of the sensed object. Consequently, the captured data of the same object point highlighted sequentially by all the laser has to be assembled. Hence, (minor) mismatches of the different wavelength channels of the data are inevitable. This section intents to quantify these mismatches and monitors how they evolve with different system configurations and measurement speeds.

4.2.1 Method for quanitifying the mismatches

In order to quantify the mismatches of the different wavelength channels a 9-by-6 corners chessboard pattern was scanned by the MWLP prototype (Figure 4.4). Each chessboard field of the pattern was 8 mm times 8 mm. Thereby, for checking repeatability the chessboard pattern was scanned three times in each configuration. Afterwards in the processed sensor data the chessboard corners were detected independently in the image representation of the *Intensity-Sum* values of each wavelength channel (Figure 4.5). Next, the Euclidean distances between the independently detected chessboard corners in the different channels were calculated. These distances are assumed to describe the misalignments of the different channels well. Ideally, they would be zero. However, in real world due to the mismatches some values close to zero are expected.

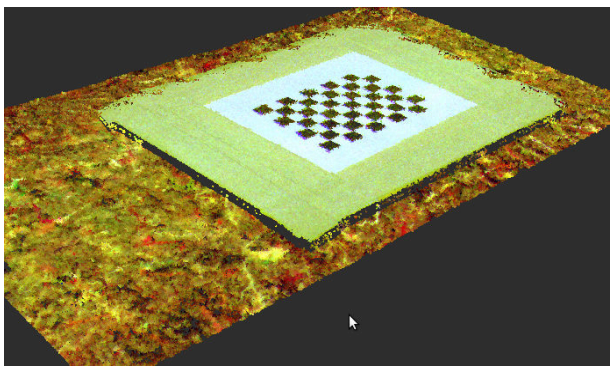


Figure 4.4: Point cloud of chess board pattern scanned with #2, #6, and #4 [137].

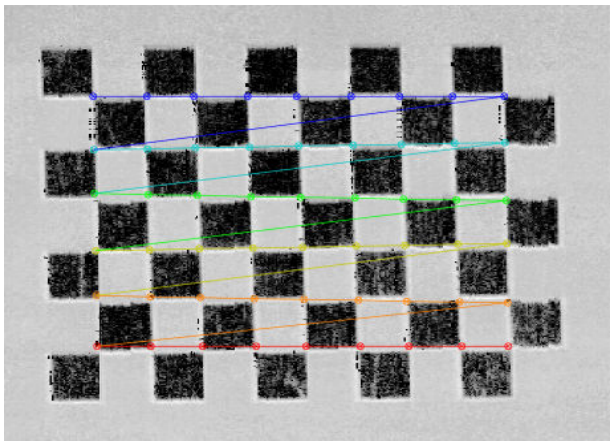


Figure 4.5: Detected chessboard corners in the gray-scale image of a single laser channel.

4.2.2 System configurations

The mismatches were monitored in different system configurations. Different lens optics and different laser constellations were taken into account.

- **Lens optics**

Three different lens optics were used, a Pentax C3516-M lens, a Schneider XENOPLAN 1.9-35 ruggedized lens and a RICOH Fl-BC1218A-VG lens. The lens optics are itemized in Table 4.1 and shown in Figure 4.6.

Table 4.1: Specifications of the tested lens optics.

Product	Focal length	Min f number	Min object distance
Pentax C3516-M	35 mm	1:1.6	0.4 m
Schneider XENOPLAN 1.9-35	35 mm	1:1.9	0.35 m
RICOH Fl-BC1218A-VG	12.5 mm	1:1.8	0.3 m

The difference between the Pentax and the Schneider lens is mainly that the Schneider lens is broadband compensated in the wavelength range between 400 nm and 1000 nm. Therefore, it is the expected to provide higher transmittance and less distortion particularly in the NIR range than the relatively low cost Pentax lens, which does not have this feature. However, the optical specifications of both lenses are fairly the same.

The major difference between the RICOH lens and the other optics is its lower focal length. Therefore, the FOV of the camera and, hence, the MWLP system is wider using this optics. While using the Schneider and Pentax optics typically only a stripe of approx. 20 cm width can be captured, with the RICOH lens this stripe can be widened to 60 cm. This is illustrated in

Figure 4.6: The used lens optics: Pentax C3516-M (left-hand side), Schneider XENOPLAN 1.9-35 (center) and RICOH Fl-BC1218A-VG (right-hand side).

Figure 4.7. The top-left image in this figure shows scan data of a scene with a cable collected with the Schneider lens. The bottom-left image depicts scan data of the same scene collected with the RICOH lens. A photo of the scene is given on the right-hand side. It can clearly be seen that the Schneider lens provides with more detail, but the left and right parts of the scanned cable are outside the FOV. Hence, they do not appear in the scan data. The RICOH lens, in contrast, captures the cable completely.

However, it has to be stated that the system was during this entire work mainly developed with focus on the FOV of 20 cm. The tests with the RICOH lens are mainly to show the feasibility of wider FOVs. But measurements at the same level of data quality would also require other lasers and reordering of the LEDs for background illumination. This manifests in the differences of point density and the darker pixel colors on the scan sides notable in the bottom-left image of Figure 4.7. This effect is caused by the inhomogeneous intensities along lines of the applied line lasers. As Figure 4.8 shows, the reflection intensity of the laser lines is notably higher in the center of the camera's FOV. Hence, if the system must be operated in this mode for an application, other line lasers with a more homogeneous intensity distribution have to be applied.

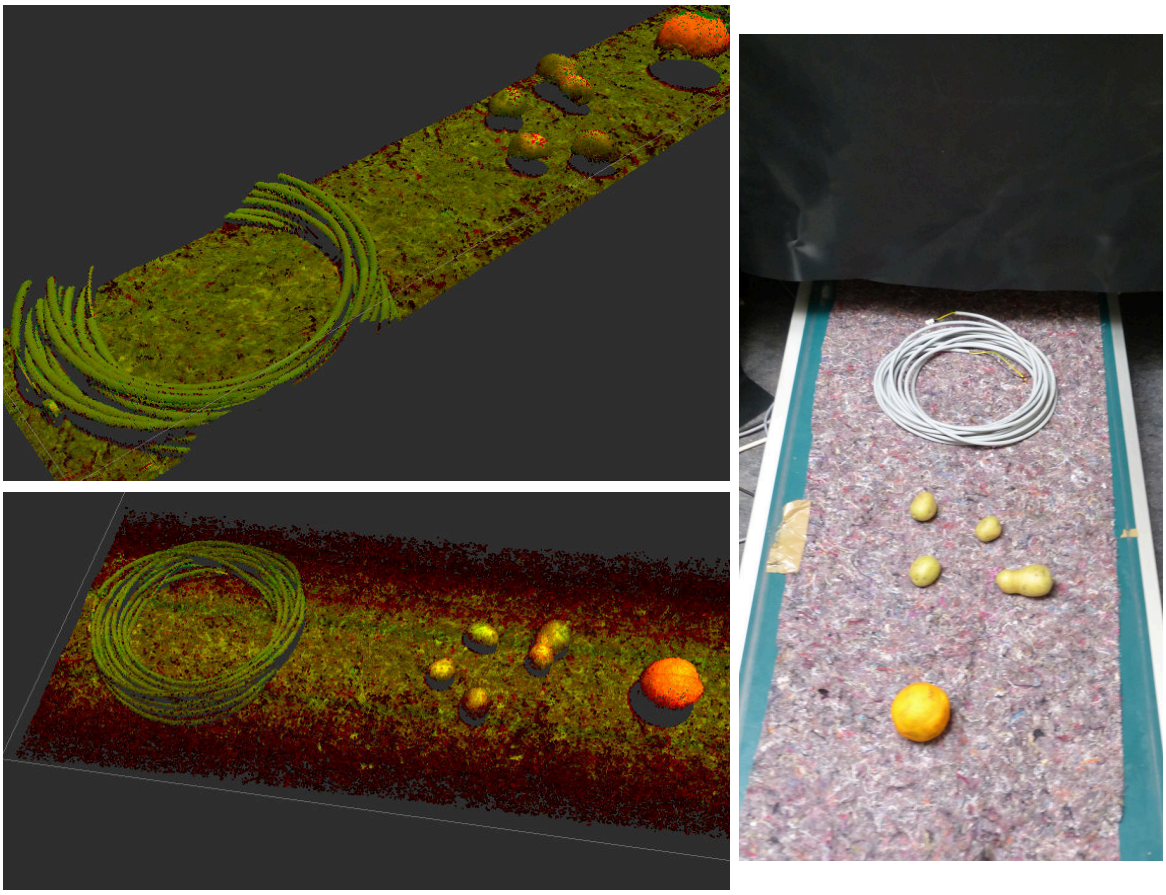


Figure 4.7: Wider FOV with the RICOH lens.

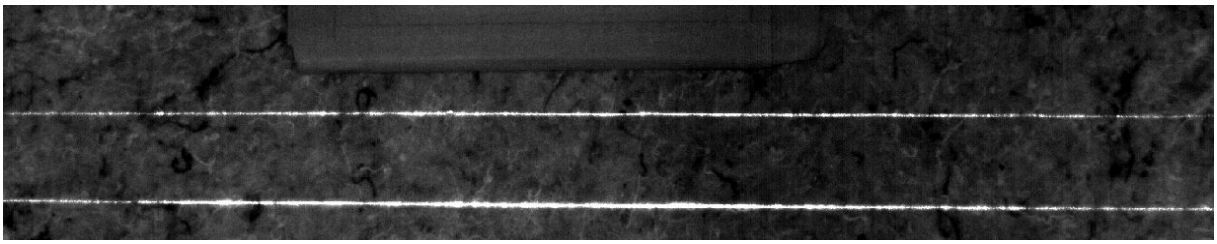


Figure 4.8: Inhomogeneous lines of the lasers #4 (top) and #6 (bottom).

- **Laser constellations**

The line assembly has been tested in a constellation with lasers #2, #4 and #6 (RGB) as well as with the combination of #4 #6 and #8 (RG_IR) (cf. Table 3.2). However, as will be seen in Chapter 5 the configuration with lasers #4 #6 and #8 is the most relevant for the focused applications. Therefore, the RG_IR constellation was more comprehensively checked.

- **Sampling modes**

As described in Section 3.1.1, the camera of the MWLP prototype of type Baumer HXG20NIR can be operated in different sampling modes. Thereby, the frame rate can be increased beyond the maximum full resolution frame rate of 100 Hz if the camera is operated in binning/subsampling modes and/or a partial scan ROI is active. For these tests the three sampling modes full frame, binning (here referring to 2x1 binning) and subsampling (here referring to 2x2 subsampling) are tested. E.g. in subsampling mode, theoretically up to 420 Hz camera frame rate are possible. However, as this high frame rate caused instabilities in the camera driver and the processing pipeline the frame rate in the subsampling modes was adjusted to approx. 340 Hz - 360 Hz.

4.2.3 Statistics for different system configurations

In Table 4.2 the descriptive statistics for the evaluation of the accuracy of the line assembly for different optics and sampling modes. All measurements were conducted with the minimum adjustable speed of the used conveyor of 0.025 m/s. For the 9-by-6 chessboard pattern and three wavelength channels, there were 162 corner pairs of which their distances could be calculated. Most measurements were repeated 3 times, i.e., the total number of (misalignment) distance samples was 486. The figured distances in pixels were multiplied with the pixel sizes obtained during system calibration in order to derive the misalignment values in millimeters.

Regarding the pixel scale, it is to mention that for the RICOH lens the FOV of the camera is widened, as already stated in Section 4.2.2. However, this does not only apply to the x-direction of the camera image but also to the y-direction. While widening the FOV in x-direction provides with useful data for widening the FOV of the MWLP system the widening of the y-direction FOV does not provide useful data for this experiment. The y-direction FOV corresponds to the MR of the system. But the chessboard is flat so an increased MR is not needed for this experiment. Therefore, only an ROI of the camera image in y-direction was read out from the camera. Hence, with RICOH lens mounted the processing pipeline was fed with a lower data rate than it was capable to process. This allowed interpolating the data collected with the RICOH lens in binning mode, such that theoretically it had the same pixel size like the full resolution data collected with the RICOH lens.

As Table 4.2 states, the best result with the minimal misalignment of the different laser channels was achieved with the Schneider lens and full resolution camera readout. The average chessboard corner distance in this mode was 0.371 mm. In the other capturing modes the average distances increase up to 0.699 mm at this minimal conveyor speed. The maximum corner distance found

Table 4.2: Descriptive statistics for the matching accuracy in different configurations.

Optics	Lasers	Sampling mode	Frame rate [Hz]	Pixel size [mm/pel]	Count	Min [mm]	Max [mm]	Mean [mm]	StdDev [mm]
Pentax	RGB	full	66.54	0.10468	162	0.050	2.065	0.699	0.367
Pentax	RG_IR	full	93.67	0.10461	486	0	2.039	0.644	0.365
Pentax	RG_IR	binning	196.58	0.15019	486	0	1.571	0.545	0.333
Pentax	RG_IR	subs.	344.96	0.22060	486	0	1.560	0.469	0.278
Schneider	RG_IR	full	99.86	0.10132	486	0	1.232	0.371	0.234
Schneider	RG_IR	binning	197.27	0.13557	486	0	1.749	0.476	0.350
Schneider	RG_IR	subs.	343.89	0.20270	486	0	1.623	0.434	0.294
RICOH	RG_IR	full	133.02	0.29979	486	0	1.464	0.444	0.263
RICOH	RG_IR	binning	264.26	0.29979[1]	486	0	1.585	0.505	0.257

at all was 2.065 mm, in the optimal capturing mode this maximum error was 1.232 mm. The standard deviations of the monitored values around their means fairly correlate with the mean at roughly 60 % of the mean's value.

Hence, it can be stated that with the suggested processing pipeline it was possible - neglecting individual outliers - to stably achieve sub-millimeter accuracy of the line assembly in all capturing modes.

4.2.4 Accuracy of the line assembly at different speeds

After the accuracy of the line assembly was checked for the different capturing configurations at minimal conveyor speed it was monitored how these values evolve if the speed of the conveyor is increased.

If the conveyor speed is increased with constant camera frame rate, the main difference for processing chain is the increased distance moved by the conveyor between two subsequent camera shots. Assuming differences in the motion blur of the images can be neglected, this allows emulating higher conveyor speeds with prerecorded camera images. This is done for all evaluations shown in this section. The motion blur of the camera images is very low due to the short exposure times in the range of one or two milliseconds. Further, in Chapter 5.1 experiments are described which show the system can actually be operated at the conveyor speeds mentioned here. However, in order to allow automating the experiments described here and not having to capture data for each configuration and each speed step the higher speeds are emulated here. This is done in the following manner: For increasing the speed from real capturing speed of 0.025 m/s to emulated speeds of 0.05 m/s, 0.1 m/s, 0.2 m/s and so on only every 2nd, 4th, 8th

[1] Interpolated

Table 4.3: Averages of the corner misalignments at different (emulated) speeds.

Optics	Pentax	Pentax	Pentax	Pentax	Scheider	Scheider	Scheider	RICOH	RICOH
Laser	RGB	RG_IR	RG_IR	RG_IR	RG_IR	RG_IR	RG_IR	RG_IR	RG_IR
Sampling	full	full	binning	subs.	full	binning	subs.	full	binning
Speed [m/s]	Mean of corner misalignments [mm]								
0.025	0.699	0.644	0.545	0.469	0.371	0.476	0.434	0.444	0.505
0.05		0.843	0.599	0.481	0.557	0.643		0.547	
0.1		1.272	0.817	0.550	0.765	0.646	0.546	0.628	0.600
0.15			0.779	0.609	0.670	0.543		0.758	
0.2			0.852	0.528	1.077	0.689	0.510	0.696	
0.25			1.094	0.621		0.722		0.878	0.727
0.3				0.641		0.755	0.562	1.282	
0.35				0.736		0.842		1.362	0.780
0.4				0.771		0.779	0.548		
0.45				0.789					0.937
0.5				0.944			0.708		
0.55				0.880					1.030
0.6				1.156			0.847		
0.65				1.108					1.015
0.7							1.146		
0.8							1.308		
0.9							1.605		

etc. image of the collected images is actually passed on to the processing chain. This allowed reusing the data captured for the previous section for these experiments.

The data captured for the previous steps was used for these evaluations. The number of images passed on to the processing chain was step by step decreased, thereby emulating increased conveyor speeds. At each step the chessboard detection was performed on each wavelength channel of each processing outcome image. The means of the corner misalignments for the different capturing configurations and different speeds are given in Table 4.3. However, it was not possible to detect the chessboard corners for all speeds tested. This can be seen in Figures 4.9 to 4.11. Figure 4.9 shows the data from the chessboard scan with the Schneider lens in subsampling mode at 0.025 m/s. The chessboard pattern is clearly identifiable. There are only minor mismatches notable by the colored sheen around some chessboard fields. Figure 4.10 shows the same scan with an emulated conveyor speed of 0.6 m/s. The mismatches have increased from 0.434 mm for Figure 4.9 to 0.847 mm for Figure 4.10. However, chessboard pattern is

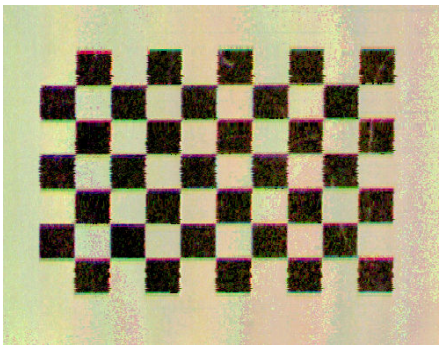


Figure 4.9: Checkerboard captured with Schneider lens in subsampling mode at 0.025 m/s. Chessboard corners clearly detectable, minor mismatches can be seen (colored sheens around the chessboard field).

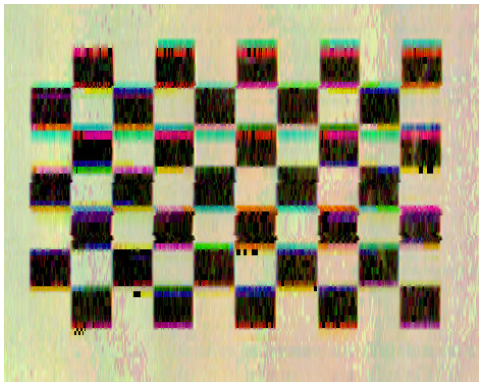


Figure 4.10: Checkerboard captured with Schneider lens in subsampling mode at 0.6 m/s. Increased mismatches and some invalid pixels can be seen. Chessboard corners are detectable, though.

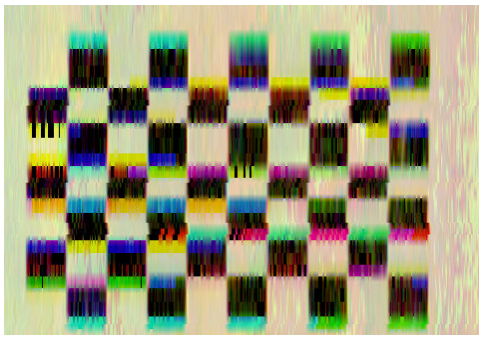


Figure 4.11: Checkerboard captured with Schneider lens in subsampling mode at 1.2 m/s. Increased mismatches and inconsistent sizes of the chessboard fields as well as many invalid pixels occur. Automatic detection of chessboard fails.

still detectable by human inspection as well as automatically, which allowed to quantify the mismatch. Figure 4.11 shows the scan with emulated speed of 1.2 m/s. The mismatches appear to have further increased. Moreover, the boundaries of the checkerboard fields are washed so much that the automatic detection of the corners fails. Hence, it is not possible to quantify the mismatch using this method anymore. This is the reason for all speed series of all configurations in Table 4.3 to stop at some point. Due to the higher capturing frame rate the subsampling and binning modes allowed higher emulated speeds without causing the chessboard detection to fail. As the allowed higher maximum speeds also the speed step was partly chosen higher for these modes, which is the reason for the couple of empty table cells below the maximum allowed speed for the respective series in Table 4.3.

A chart of the values given in Table 4.3 is depicted in Figure 4.12. Clearly, in tendency the inaccuracies increase with increased speed. Due to lower resolution the binning and subsampling

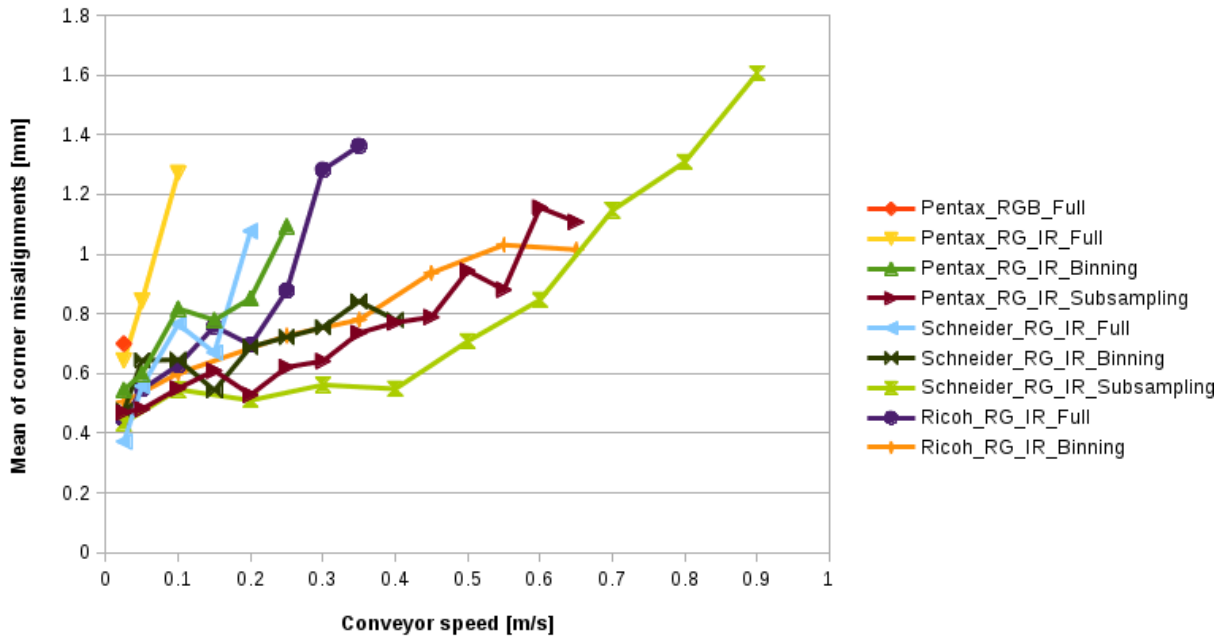


Figure 4.12: Chessboard corner misalignments over different speeds and configurations.

modes typically have a little higher mismatch at the lowest speed. However, the values are ascending much slower with the increasing speeds for these modes, which can be explained by the higher frame rate.

4.2.5 Different processing modes

To validate some choices made during the design and development the accuracy of the line assembly was further evaluated as suggested in Chapter 3 and with some thinkable modifications. These tests were only conducted for the highest accuracy achieved, i.e., using the Schneider lens, RG_IR lasers and full resolution capturing at minimum conveyor speed. The prerecorded data was used, again. This means all these processing tests were conducted on the same data processed multiple times with the mentioned modifications of the MWLP image pipeline.
The descriptive statistics of these tests are given in Table 4.4. Figure 4.13 depicts the mean misalignments in the different modes. The tested modifications of the MWLP image pipeline are:

- **With camera calibration**

As stated in Section 3.6.1 the camera calibration and image rectification prior to passing the captured images to the processing chain was skipped. As described and measured there, the industrial class lens optics used here in combination with the 2/3" image sensor do not cause much distortion so the modifications this step applies to the image are hardly notable. Nevertheless, for this test a camera calibration was performed using the camera calibrator ROS package `camera_calibration` [84] and an image rectification was performed prior to passing the images

Table 4.4: Misalignment statistics for different processing modes.

	As Suggested	With camera calib	With Center of Gravity	Without matching	Without skew calib	Without scan check
Count	486	486	486	486	486	486
Mean [mm]	0.367	0.463	0.369	0.445	0.647	0.385
StdDev [mm]	0.217	0.251	0.215	0.230	0.386	0.244
Min [mm]	0	0	0	0	0	0
Max [mm]	1.232	1.482	1.216	1.434	2.747	1.244

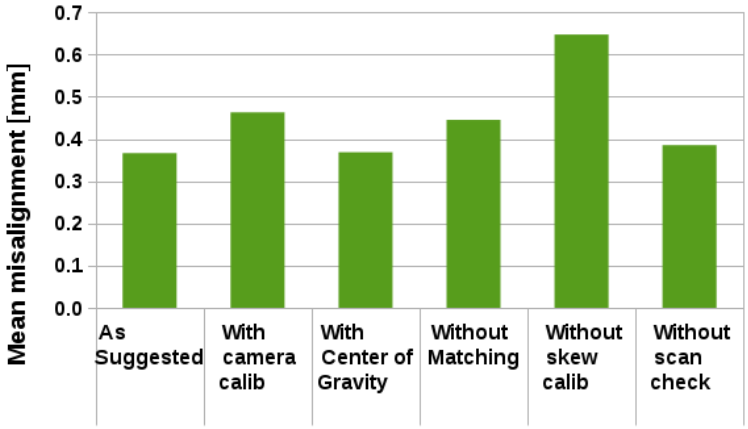


Figure 4.13: Chessboard corner misalignments for different processing modes.

to the processing pipeline. In this case, the rectification did even cause a slight increase of the mean mismatch (cf. Figure 4.13).

- **With Center of Gravity**

As elaborated in Section 3.4, a commonly used technique for detecting laser lines is Center of Gravity (COG) analysis. However, Mueller et al. have shown that a prior segmentation before the COG analysis improves the line detection, particularly if scattering is present [88]. By default, the line detection of the MWLP prototype detects the lines by thresholding, i.e., segmentation, and assumes the line position is in the center of the detected segment. The COG analysis is skipped favoring the reduced runtime.

For this test the combination of prior segmentation and following COG analysis, as suggested by Mueller et al. [88] was tested in order to compare the result with the default setting of the MWLP system. As Table 4.4 shows, there are next to no differences between both modes. Apparently, the observation, that the laser line induced intensity peaks in the images are fairly

symmetric (cf. Figure 3.27), which was the basis for the assumption that COG and center of the segment are very similar, can be generalized - at least for all the images captured here.

- **Without matching**

Without conducting the image matching and having the line assembly only relying on the data provided by the rotary encoder the mean misalignment increases in this configuration from 0.367 mm to 0.445 mm. This appears to be very few. In a prior test conducted at beginning of 2014 skipping the matching caused a more significant increase of the mismatch from 0.563 mm to 1.834 mm [137]. Likely, there was very few slip in the data of the rotary encoder during this test. And still, the matching is indispensable for outdoor use of the system on a vehicle as slip is always present during field tests.

- **Without skew calibration**

Without performing the correction of the skewness between the camera image plane and the object plane of the conveyor (cf. Section 3.3.2), the mean misalignment in this test increased from 0.367 mm to 0.647 mm. This correlates with the tests from 2014 where the increase was from 0.563 to 0.929 [137]. It again shows the necessity of this processing step and the respective calibration step.

- **Without scan check**

The scan check filters out pixels after the line assembly where either not enough laser lines could be detected for a particular point or the z-distance values obtained from different laser lines differ too much (cf. Section 3.3.5). Skipping this step does not change the so-defined accuracy of the line significantly, there is only a minor increase of the mean misalignments from 0.367 mm to 0.385 mm. This could be expected as the scan check mainly works on the distance data and the chessboard is flat, i.e., no significant filtering can be expected. This step does not help for the line assembly. It only filters out wrong 3D pixels in complex structures. Hence, skipping it should not have a major effect on the line assembly, which is shown here.

4.3 Precision and comparison of the distance measurement

After the accuracy of the line assembly was tested, the precision of the distance measurement of the MWLP prototype was evaluated. Further, it was compared with the precision of other contactless optical image-based range measurement devices. In order to conduct these tests the 3D test object already mentioned in Section 3.6.1 and shown in Figure 3.46 was used. As stated, it is a laser sintered object created using a laser sintering machine for rapid prototyping of type EOS Forminga P 110 with a minimum layer size, i.e., tolerance, of 0.06 mm. It comprises different flat height steps of different heights and different area dimensions.

4.3.1 Measurement procedure

In order to check the distance measurement precision of the different devices the image-based distance data of the 3D test object was recorded with each device and with the MWLP prototype

in different configurations. Next, the region containing the data with the 3D test object was cropped by manual inspection from each of the collected data sets as these oftentimes contained huge regions around the test object of interest.

- **Data correction and normalization**

After cropping, possible tilt of the data was corrected. This can be caused by the respective sensor system not being in perfect nadir position of the object, i.e., not looking straight vertically down onto the object. An example for such data is given in Figure 4.14. To recall the 3D test object it is depicted again in Figure 4.15. In Figure 4.14 gray-scaled distance data is shown, invalid pixels are colored red. Clearly, parts of the test object of identical height appear farther away in the top-left part of the image than test do in the bottom-right part due to an imperfect relative positioning of sensor and object during the measurement.

In order to correct this effect the user has to mark wide regions in the data that represent a single height step of the test object. These marks can be seen in Figure 4.16. Thereby, it is not necessary to outline the entire region. More importantly, the outlined regions may safely contain only pixels of the respective height step. Hence, ideally the user marked region outlines are smaller than the respective region actually is. Following this, a bilinear regression takes place. I.e., all distance values of pixels situated within the user marked regions are fitted to the plane model $z = ax + by + c$ using least-squares regression. Applying the so obtained model a tilt plane is estimated for the entire image dimension. The gray-scaled values of the estimated tilt plane of the data shown in Figures 4.14 and 4.16 are given in Figure 4.17. Next, the estimated tilt plane is subtracted from the measured distance values in order to correct the tilt caused by imperfect relative positioning of sensor and test object. Further, the images are resized to have the same size and are normalized to scale within the same range for all used sensors and configurations. The tilt corrected and normalized distance data as gray-scaled depth image for the data shown in Figures 4.14 and 4.16 are depicted in Figure 4.18. During the normalization the height range of the less than 10 cm high object is scaled in 11 Bit range. Hence, from the numerical point of view a distance resolution of less than 0.005 mm can be represented while the minimum step of the test object is 0.06 mm. I.e., the numerical resolution of the normalized data is one order of magnitude better than the minimum step.

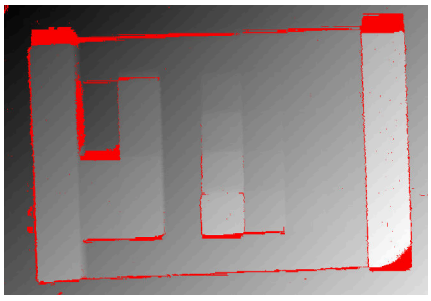


Figure 4.14: Tilt in the distance data caused by the sensor not looking straight from top onto the object.

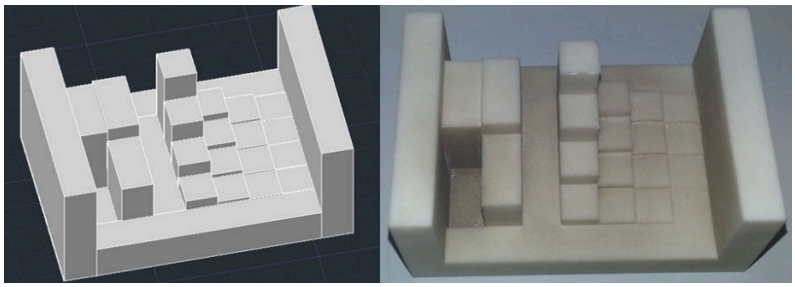


Figure 4.15: Recall CAD view and photo of the 3D test object already depicted by Figure 3.46.

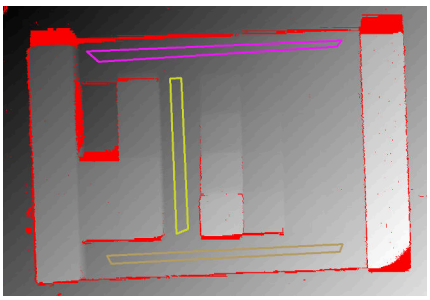


Figure 4.16: Regions belonging to a single height step of the test object marked by the user.

Figure 4.17: Grayscaled estimated tilt plane from the regions marked in Figure 4.16.

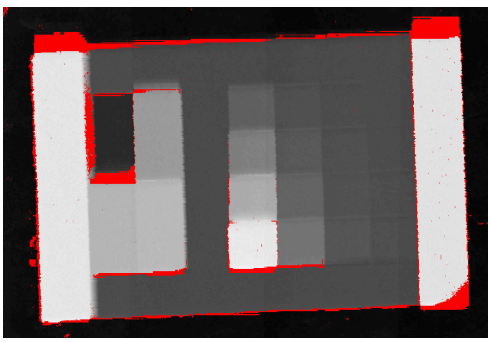


Figure 4.18: Tilt corrected and normalized distance data of the data shown in Figure 4.14.

- **Checking distance step measureablity**

For checking how precise the distance measurement of the different sensors is and, thereby, not relying on data sheets, the measureablity of the different steps was tested. Noise had to be taken into account. To check whether the different height steps are measureable, the user had to mark adjacent regions in the image. In the marked adjacent regions all pixels of each region must belong to a single plane and the respective height step must lie between these planes. The regions can be marked using circles (ideal for small area steps) or polygons (for planes with relatively huge area). In a perfect world without noise all pixels of these regions would theoretically have the same distance values and the distance values of different regions would differ from each other. However, as there is noise present this is clearly not the case. Therefore, a particular distance step was said to be measureable, if the maximum of the standard deviations of the pixel's values within each of the marked adjacent regions was less than the difference of the means of the pixel's values within each of the marked adjacent regions. Two examples for this are given in Figures 4.19 and 4.20. In Figure 4.19 the height step between the marked adjacent regions is measureable. In Figure 4.20 it is not measureable [137].

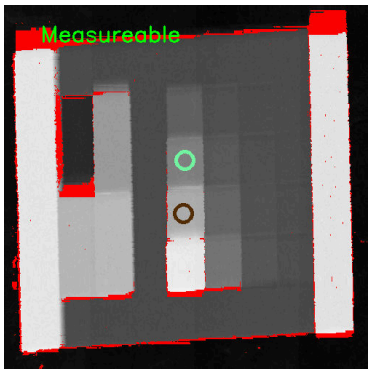


Figure 4.19: Two marked adjacent height steps that are measurable.

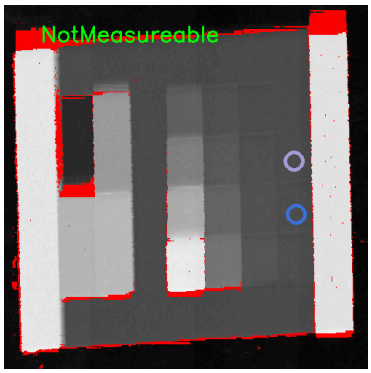


Figure 4.20: Two marked adjacent height steps that are not measurable.

4.3.2 Comparison of different sensors

In order to compare the distance precision of the MWLP system with other state-of-the-art distance measurement devices a measureablity test with the distance maps gathered using different sensors was conducted. Therefore, the 3D test object (cf. Figure 4.15) was scanned with the following TOF and structured light sensors:

- **Nippon FX 8** (TOF-based laser scanner with 2-axes rotating mirror) [48]
- **IFM O3D200** (TOF-based camera with PMD sensor) [53]
- **Microsoft Kinect I** (structured light projection)
- **Microsoft Kinect II** (TOF-based camera)

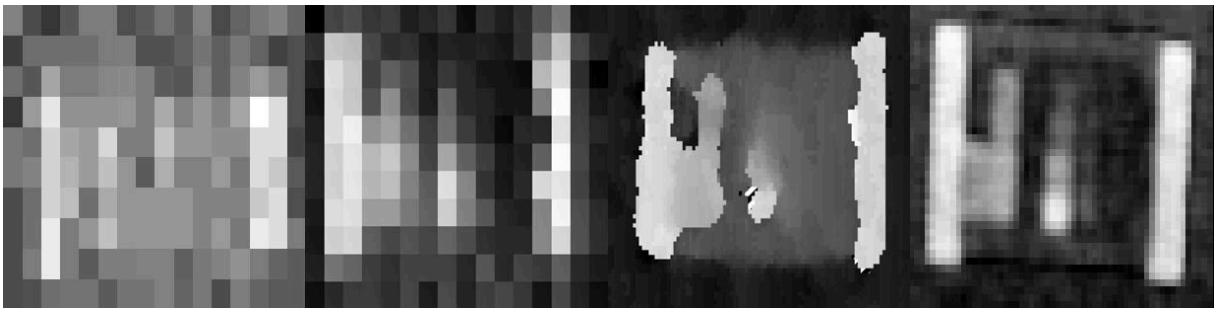


Figure 4.21: Depth maps the 3D test object obtained by different sensors scaled to same size. Left most: Nippon FX8, center-left: PMD TOF; center-right: Kinect I; right-most: Kinect II. Invalid pixels appear hatched [137].

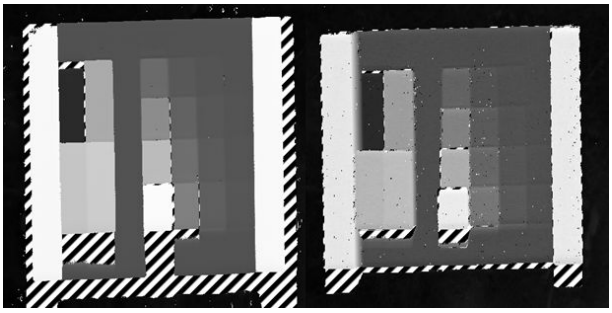


Figure 4.22: Depth maps the 3D test object obtained by different sensors scaled to same size. Left: LMI Gocator 2350; right: MWLP prototype. Invalid pixels appear hatched [137].

Further, it was scanned with these laser line profiling devices:

- **LMI Gocator 2350** (commercial laser line profile sensor) [73]
- **MWLP system prototype** (developed here)

The results of these scans are depicted in Figures 4.21 and 4.22. Figure 4.21 shows gray-scaled depth data of the Nippon FX 8, the IFM PMD 3D camera, Kinect I and Kinect II. Figure 4.22 shows the scan result for the tested line profile sensors LMI Gocator 2350 and the MWLP prototype.

Based on the distance measurements in Figures 4.21 and 4.22 the measurablities of the different steps were tested. The results are given in Table 4.5. The values for IFM PMD 3D, Kinect I, Gocator and MWLP of this Table were published in [137], the other sensors have been added later.

Table 4.5: Height steps of the test object. Height steps measureable with a sensor system are marked X [137].

Height step [mm]	0.06	0.12	0.18	0.24	0.30	0.42	0.60	0.84	1.14	1.56	2.16	3.00	4.14	5.70	7.86	10.86
Nippon FX 8																X
IFM PMD 3D														X	X	X
Kinect I													X	X	X	X
Kinect II												X	X	X	X	X
Gocator		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
MWLP					X	X	X	X	X	X	X	X	X	X	X	X

The tests for measureablity given in Table 4.5 are relatively rough and have only been conducted once. Nevertheless, the result is clear and one can come to the same conclusion by inspection the sensor data depicted in the different parts of Figures 4.21 and 4.22: The only field-applicable image-based range measurement devices capable of achieving distance resolutions in the sub-millimeter range are line profiling systems. The test may be a bit favoring the LP devices as the Measurement Range (MR) of all devices with results shown in Figure 4.21 is in the order of magnitude of meters while the MR of the LP is only a couple of decimeters. However, neglecting the MR in terms of resolution they clearly outperform the other sensors.

Further, inspecting Table 4.5 it can be seen, that the MWLP prototype does not reach the distance precision of the commercial single laser LP sensor Gocator 2350. This can be explained due to the reduced ROI that is only available per laser. As for a constellation with 3 lasers only one third of the image is available per laser, the distance resolution of comparable single laser systems cannot be achieved. This drawback has to be accepted when using the MWLP system.

4.3.3 Extended statistics for different configurations of the MWLP prototype

As stated, the tests conducted in Section 4.3.2 were relatively rough, as the differences between the different sensors are huge. This Section intents comparing different configurations of the MWLP prototype. Hence, the differences were assumed to become smaller. Consequently, the test procedure was refined further.

First change made to extend the statistics was scanning the test object 3 times in each configuration. A further extent was made to the manual placement of the marked regions that are taken into account for checking the measureablity of the height difference of adjacent height plane. Their random placement in a part of the height plane with more or less local deviation could slightly influence the outcome of measureablity - not for very significant steps or clearly unmeasurable steps, but at the boundary of measureability for one or to steps there were divergent results possible. Therefore, the manually marked regions were automatically shifted into five different places around the original marks and the measureablities for all these sub regions

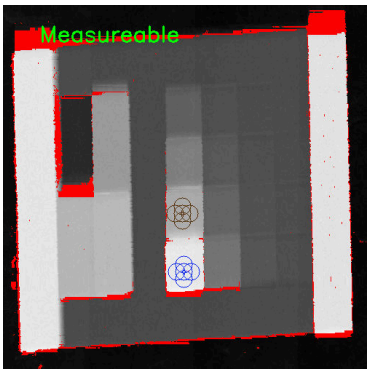


Figure 4.23: Automatically shifted marks for automatically comparing different sub sets of each plane.

were determined. In Figure 4.23 all the shifted and compared regions (five per height plane) are drawn for a specific height step.

Table 4.6 shows an example for the outcome of these 3D measurement evaluations for a specific configuration and speed. Here the statistics for the Schneider lens with full camera resolution and RG_IR laser constellation operating at minimum conveyor speed of 0.025 m/s are given as example. In the right most column the different height steps of the 3D test object are listed. The total number of compared image region pairs for each height step is always 15 as the object was captured three times and each manually marked region was automatically shifted to 5 adjacent positions. The column measurable count states how many of these monitored were measurable according to the above mentioned definition, i.e., the difference of the means of the pixel's values of both regions are less than the standard deviations of the pixels values. For most of the height steps either all monitored regions pairs are measurable or not measureable. Only for some regions in each configuration this depends on some random factors. In this case, only the height step of 0.3 mm is not clearly defined. In the right columns the aggregated difference of the means of the pixel values taking into account all pixels belonging to any monitored region in any captured record of the test object in this configuration is given as well as the respective overall standard deviation for all pixel samples. Using this as indicator, the height step of 0.42 mm was measurable while the height step of 0.3 mm was not.

Table 4.7 shows aggregated results for different capturing configurations. Thereby, for each configuration the step with the first measureable sample is listed. This step is the lowest step, where at least one out of the fifteen compared region pairs was measureable, i.e., 0.3 mm for the Schneider optics with full resolution according to the sample result given in Table 4.6. Further, the average measurable step is listed for each configuration, i.e., the minimum step, for which the mean aggregated over all monitored pixels is less than the respective standard deviation aggregated over all monitored pixels. In case of the sample data in Table 4.6, this step is at 0.42 mm. Finally, the lowest step measurable for all fifteen out of fifteen region pair comparisons is given. For data in Table 4.6 this value is also 0.42 mm.

Table 4.6: Sample 3D check statistics for Schneider lens, full resolution.

Physical distance [mm]	Measureable count	Total count	Overall difference of means	Overall standard deviation
10.86	15	15	490.13	17.03
7.86	15	15	357.73	16.45
5.7	15	15	256.87	15.14
4.14	15	15	161.47	14.20
3.00	15	15	128.67	14.47
2.16	15	15	89.40	13.99
1.56	15	15	68.07	12.45
1.14	15	15	45.87	13.55
0.84	15	15	29.07	13.76
0.60	15	15	21.93	12.43
0.42	15	15	18.93	12.77
0.30	4	15	10.80	15.01
0.24	0	15	7.40	18.14
0.18	0	15	7.27	23.49
0.12	0	15	8.13	21.44

There are no drastic variations between the different configurations in the 3D precision. Average values for precision in all configurations are between 0.3 mm and 0.6 mm. And even the 'worst-case' precision values of 1.56 mm is still significantly better than the values of TOF-based or Structured Light (SL)-based systems estimated in Section 4.3.2.

Table 4.7: Distance precision check results for different configurations.

Optics	Sampling mode	Min measureable sample [mm]	Average measureable [mm]	All samples measureable [mm]
Pentax	full	0.18	0.42	0.6
Pentax	binning	0.3	0.3	1.56
Pentax	subsampling	0.3	0.3	0.6
Schneider	full	0.3	0.42	0.42
Schneider	binning	0.18	0.42	0.6
Schneider	subsampling	0.18	0.6	1.56
Ricoh	full	0.42	0.42	1.56
Ricoh	binning	0.42	0.42	1.56

4.3.4 Influence of the conveyor speed on the distance precision of the MWLP prototype

Appending to the results shown in Table 4.7, the influence of the movement speed on the distance measurement was tested. Same as in Section 4.2.4 these experiments where not conducted by physically varying the conveyor speed but by down sampling recorded image data at lowest speed before passing it to the MWLP image pipeline. This - again - allowed reusing the data captured for the evaluation in Section 4.3.3 and automatically running this tests on the same, but sparsed data.

The results of these tests are listed in Table 4.8. Unlike for the line assembly, there are no significant changes of the distance measurement precision with different speeds. Even the 'worst-case' precision does not exceed the 1.56 mm step regardless of the emulated speed value. The average values for most configurations do not show any variations at all, as depicted in Figure 4.24. For a few configuration variations are notable. However, a tendency for correlation with the conveyor speed cannot be seen. The average values for all configurations stay in the range between 0.3 mm and 0.6 mm regardless of conveyor speed. This can be explained by the main factor influencing the distance precision of line profiling system being the triangulation angle, i.e., mounting of camera and lasers. This was not touched for these tests. Hence, the tests showed, that the conveyor speed does not notably influence the distance precision.

Table 4.8: Measurable speed steps for different configurations and different (emulated) speeds.

	Pentax full	Pentax binning	Pentax subs.	Schneider full	Schneider binning	Schneider subs.	Ricoh full	Ricoh binning
Speed [m/s]	Average measureable [mm] / All samples measureable [mm]							
0.025	0.42/0.6	0.3/1.56	0.3/0.6	0.42/0.42	0.42/0.6	0.6/1.56	0.42/1.56	0.42/1.56
0.05	0.42/0.6	0.3/0.3		0.42/0.42	0.42/0.6	0.6/1.56	0.42/1.56	0.42/1.56
0.1	0.42/0.6	0.3/0.3	0.6/0.6	0.42/0.42			0.42/1.56	
0.125					0.42/0.6			
0.15	0.42/0.6	0.3/1.56		0.42/0.42		0.6/1.56	0.42/1.56	0.42/1.56
0.2	0.42/1.02	0.3/1.56			0.42/0.84		0.42/1.56	
0.25		0.3/1.56	0.3/0.6			0.6/1.56	0.42/1.56	0.42/1.56
0.275					0.42/0.84			
0.3		0.3/1.56						
0.35		0.3/0.6			0.3/0.6	0.6/1.56		0.3/1.56
0.4			0.6/1.56					
0.425					0.3/0.42			
0.45						0.6/1.56		
0.55			0.6/1.56			0.6/1.56		
0.65						0.6/1.56		
0.75						0.6/1.56		
0.85						0.6/1.56		
0.95						0.6/1.56		
1.05						0.6/1.56		

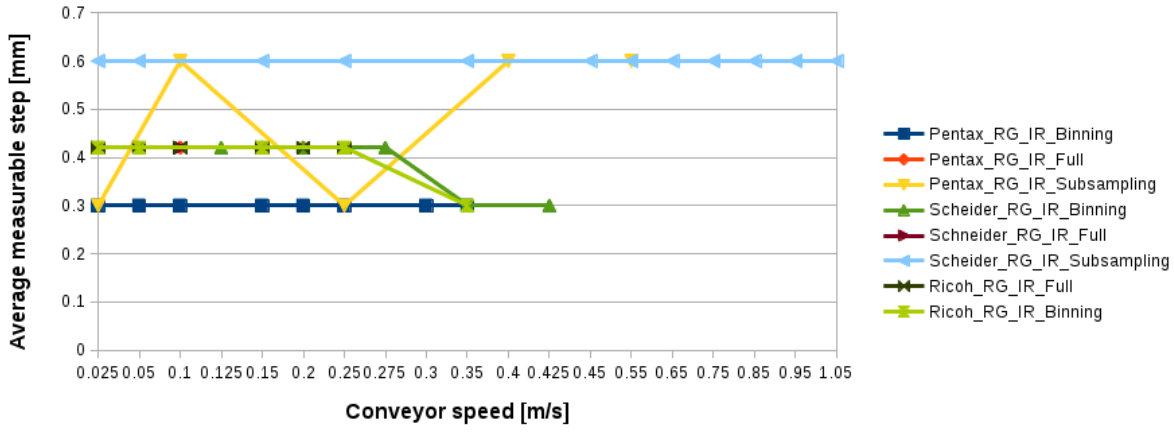


Figure 4.24: Average measurable steps over conveyor speed chart.

4.3.5 Variations of the processing chain

After the influence of the speed on the distance precision was checked, the effects of variations of the processing chain in terms of modifying the MWLP image pipeline (cf. Figure 3.10) was tested. The changes tested here were the same modifications, whose influences on the line assembly were already tested in Section 4.2.5. The results of these tests are given in Table 4.9.

Same as in Section 4.2.5 these tests were only conducted for the configuration with Schneider lens, full camera resolution, RG_IR lasers and minimum conveyor speed of 0.025 m/s. The tested modifications of the MWLP image pipeline are identical with those tested in Section 4.2.5:

- **With camera calibration**

Unlike in Section 4.2.5, where adding the camera calibration slightly weakened the final result, here including the camera calibration and adding image rectification as well as distortion using the pinhole camera model to the processing chain slightly improved the result. The average measurable step and the step, where all 15 monitored sample region pairs where measureable both improved by one step from 0.42 mm to 0.3 mm. However, having in mind the required computational resources and the slightly weakened result for the line assembly the minor improvements observed here still do not justify this processing step to be standard for the MWLP prototype.

- **With Center of Gravity**

Adding the COG evaluation after the segmentation for line detection - hence, line detection according to Mueller et al. [88] - did not have any effects on the result compared with the suggested method. Consequently, the suggested method, i.e., assuming center of the line segment as line position during line detection, was followed favoring the reduced processing time.

- **Without matching**

Adding the optical matching or skipping the matching and relying only on the rotary encoder data did also not influence the 3D measurements. This was predictable as the matching is primarily intended to improve the line assembly - which it showed to do (cf. Section 4.2.5) - and secondarily to allow evaluation of line intensity and backscattering. An influence here could not be expected and, hence, lacking an improvement here does not mean the processing step is not needed.

- **Without skew calibration**

Skipping the calibration and correction of the skewness between camera image plane and conveyor object plane does slightly improve the minimum measurable step and the average measurable step from 0.3 mm and 0.42 mm to 0.24 and 0.3 mm, respectively. However, apparently the deviations between the results are increased shown by a worsening of the step for which all sample region pairs were measurable from 0.42 mm to 0.6 mm. Having in mind the increased deviations here and the significantly improved result for the line assembly in Section 4.2.5 the relevancy of this processing step is clear.

Table 4.9: Distance precision results for different processing modes.

Processing mode	**As suggested**	**With camera calib**	**With COG**	**With out matching**	**Without skew calib**	**Without scan check**
Results	Min measureable [mm] / Average measureable [mm] / All measureable [mm]					
	0.3/0.42/0.42	0.3/0.3/0.3	0.3/0.42/0.42	0.3/0.42/0.42	0.24/0.3/0.6	0.3/0.42/0.42

- **Without scan check**

As Table 4.9 shows, adding or skipping the scan check did neither effect the quantified result here, nor the line assembly in Section 4.2.5. For the line assembly, this is clear because the scan check mainly works on the 3D data. In this case, one could expect an improvement of the step. However, the procedure used for checking the distance precision here does mainly take into account the pixels inside the height panes. In contrast, the scan check does primarily remove weakly defined pixels at sharp edges, as indicated by the light blue and dark blue pixels in Figure 3.52. Inside the height planes distance values of the different lasers do typically not differ much, i.e., the scan check just passes them. Hence, the lacking effect on the so-quantified result can be explained. In the end if a sufficient number of pixels pass the scan check, the level of trust in the values of these pixels is always improved because this means similar distance values are obtained by multiple laser lines. Furthermore, the scan check is not very computationally expensive. Consequently, it stayed part of the MWLP image pipeline.

4.4 Scattering for different wavelengths and different materials

As mentioned, a particular advantage of the MWLP system is that it allows evaluation of laser light backscattering at multiple wavelengths in an image-based manner. This section intents to show capturing backscattering data and its descriptiveness at different examples. This is of particular importance for wavelengths ranging between 600 nm and 1000 nm. In this range the laser can partly either organic tissue with high water content and is scattered back beneath the object surface [64].

Figure 4.25 shows an excerpt of an image collected by the camera of the MWLP system while scanning an apple and a piece of wood. The top line corresponds to laser #4 @ 532 nm, the bottom line is induced by laser #6 @ 650 nm. As expected, the red laser line at bottom gets scattered significantly by the orange while the green laser line is not scattered much. This is mostly due to less scattering because light of the green wavelength cannot enter the tissue as good as the light of the red wavelength. Partly this is due to higher absorption of the green light by the orange colored surface and tissue of the orange. Both lasers do not get scattered much by the piece of wood as the laser light can not enter the optically dense material.

With the MWLP system this observation can be monitored in an image-based manner. This is shown in Figure 4.26. The left part of Figure 4.26 depicts the *Scatter80Sum* values of the laser

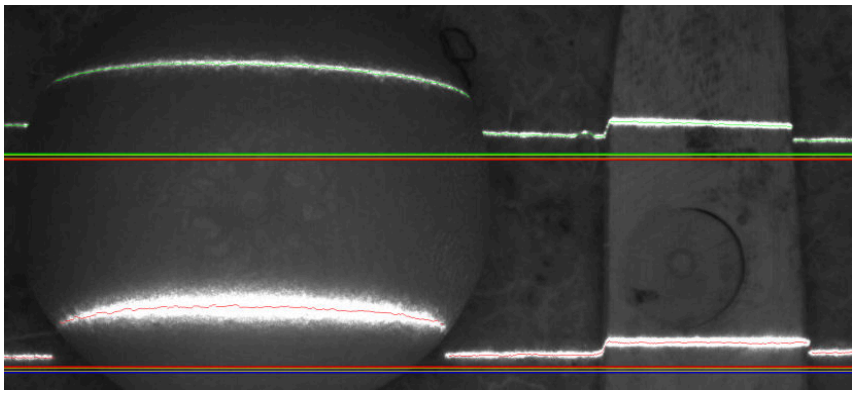


Figure 4.25: Scattering of a laser line @ 523 nm (upper) and @ 650 nm (lower) by an orange (left) and a piece of wood (right) [137].

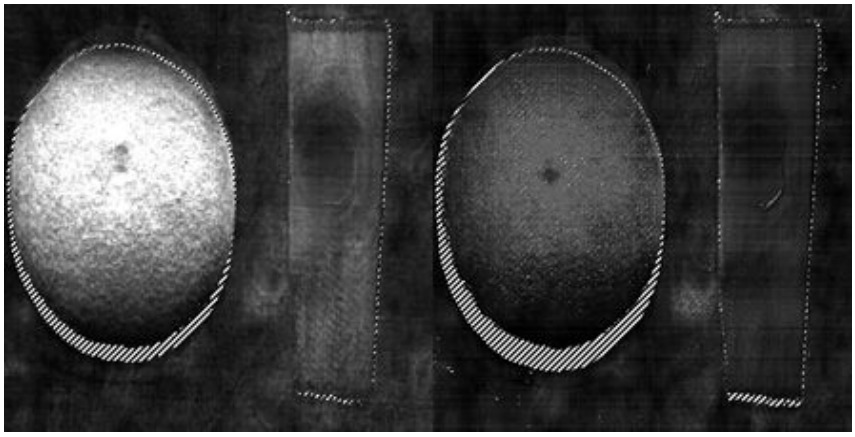


Figure 4.26: *Scatter80Sum* values as gray scale from scanning orange and wood with lasers @ 650 nm (left) and @ 850 nm (right). Invalid pixels, e.g. shaded for triangulation, appear hatched [137].

@ 650 nm as gray-scale image. On the right-hand side gray-scaled *Scatter80Sum* values of the laser @ 532 nm are shown. Both scans are made from mentioned scene given in Figure 4.25. Both images are normalized to the same gray-scale. Clearly, the scattering of the red laser is much higher than for the green laser in case of the orange. For the piece of wood both lasers do not get scattered much, as expectable after inspection of Figure 4.25.

This effect can be useful for determining fruit properties that manifest beneath the surface or classification of optically dense objects from those with high water content. For both of these cases a short example will be given here.

- **Classification of objects of different materials – potatoes and stones**

An example where the objects with high water content need to be classified from optically dense once is the separation of potatoes from stones, soil clods or alike residuals. An example of MWLP

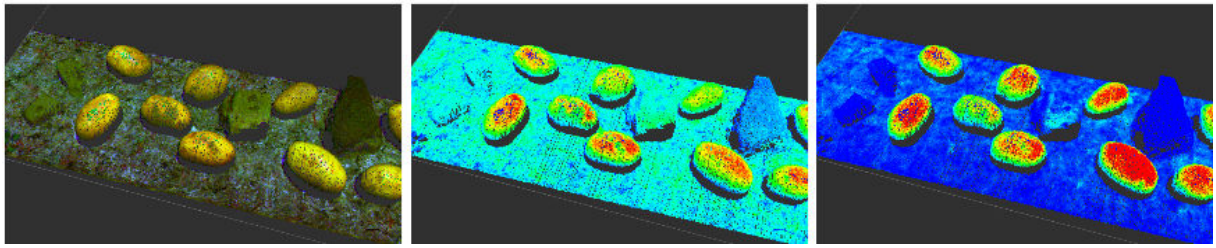


Figure 4.27: Different 3D visualizations of scan data obtained from potatoes and stones. Left: RGB visualization of *IntensitySum* values. Center: Heat map visualization of *IntensitySum* values of laser @ 650 nm. Right: Heat map visualization of *Scatter20Section* values of laser @ 650 nm.

data gathered by sensing objects this kinds is given in Figure 4.27. The left side of this image shows the RGB colored point cloud, colored using the *IntensitySum* values of the RGB lasers as described in Table 3.11. The view in center shows another visualization of the same data. Again, the data is visualized as 3D point cloud, but the pixels are colored in a heat-map-like manner with the highest numerical value mapped to red and the lowest to blue. The heat map colorization in the center shows the *IntensitySum* values of the laser @ 650 nm. The heat map colorization on the right-hand side of Figure 4.27 shows the *Scatter20Section* values of the laser @ 650 nm. Comparing these images clearly shows that the scattering feature is very selective for classification between the potatoes (high water content) and the stones/soil clods (optically dense). This will further be evaluated in Section 5.2.

- **Assessing properties beneath the surface – detection of apple damages**

In order to demonstrate assessing properties manifesting beneath the surface and, hence, are invisible to passive optical sensor systems an experiment with apples was conducted. The apples were damaged by dropping them from a height of approximately 70 cm onto a wooden plate. Immediately after the drop these damages cannot be seen, as illustrated by the photo of the apples given in Figure 4.28.

Next, the apples were scanned using the MWLP system immediately after the drop. The damages in the structure of the apple were revealed by the evaluation of the *Scatter40Section* values of the laser @ 850 nm. These values are significantly higher for the apples at damaged positions. Hence, the scattering of the light is influenced by the changes in the tissue structure beneath the surface that are induced by the drop. Consequently, the effect can be monitored immediately after the drop using the MWLP system. The observation of the damages could be approved 15 hours later with a camera. After the apples have been stored under room temperature of approx. 20 °C, due to decay the damaged positions were clearly identifiable in the image of an ordinary photo camera (Figure 4.30).

Clearly, the test with the apple is at very early stage. By now, it can only be reproduced for only green or only red apples as the *Scatter40Section_NIR* feature is also influenced by the color and geometry of the apple. Hence, to make this really usable a correction algorithm would

Figure 4.28: RGB camera photo of the damaged apples (top-left, top-right and bottom center) and other not damaged apples immediately after the drop.

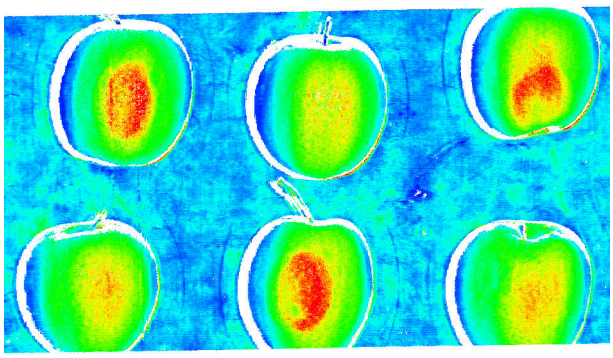


Figure 4.29: *Scatter40Section* values of NIR laser @ 850 nm photo of the damaged apples (top-left, top-right and bottom center) and other not damaged apples scanned with MWLP system immediately after the drop.

Figure 4.30: RGB camera photo of the damaged apples (top-left, top-right and bottom center) and other not damaged apples approx. 15 hours after the drop.

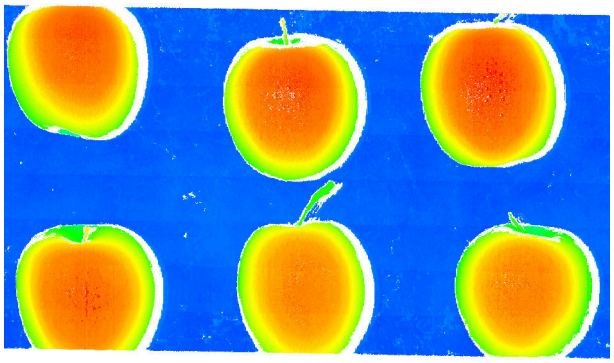


Figure 4.31: Heat map colorized depth map of the apples acquired with MWLP system while scanning the data visualized in Figure 4.29.

be needed for green/red sparkled apples. Such algorithm could e.g. improve the significance of the *Scatter40Section_NIR* channel by normalizing it depending on the apple color, i.e., the *IntensitySum* features. Further, the scattering and reflectance features extracted by the MWLP system as described in Section 3.4 do not take influences of the inclination angle of the object surface into account. Without further processing, i.e., as shown in Figure 4.29, the inclination angle is neglected. However, the inclination angle does influence the reflection and scattering (cf. Paulus et al. [101]). For leaf sensing with combination of hyperspectral imaging and LP sensors applicable correction methods taking into account the inclination angles have recently been proposed by Behmann et al. [8]. Likely, such correction could provide reproducible results for this application using only the MWLP system without need of additional sensors.

Consequently, this experiment gives an idea of the potential of the image-based scattering features derived using the MWLP system. For practical application correction algorithms for normalizing the scattering values taking into account the apple color and the inclination angle would be required. However, the required input for these corrections, i.e., reflection intensity values and 3D distance values is provided by the MWLP system (cf. distance map given for scanned apples in Figure 4.31), thus no need of an additional sensor for implementing that.

4.5 Field trials

After the system had been tested under laboratory conditions mounted on top of a conveyor, field trials were conducted in order to show the feasibility of outdoor use of the system under weakly defined agricultural field conditions.

In order to conduct these tests the MWLP system was mounted into the autonomous field robot BoniRob. It has an open module frame in its center where different modules, so-called 'BoniRob-Apps', can be mechanically attached as well as electrically and logically connected [4]. Photo of the prototype mounted into BoniRob is given by Figure 4.32. As Figure 4.32 shows, the space beneath the corpus of BoniRob is shaded to reduce distortions of the sensing process by sunlight exposure.

Figure 4.32: Photos of the MWLP prototype mounted into a field robot of type BoniRob.

4.5.1 System adjustments

A couple of system adjustments had to be made for operating the system inside the field robot BoniRob. These will be explained in this section.

- **Obtaining image positions**

As stated in Section 3.3.3, position stamps for the acquired images are required to enable the SAD-based matching within the available processing time per image. The position stamps are used as initial guess for the matching. Given them, the matcher must only find the local optimum rather than a global optimum. Hence, the required processing time is drastically reduced. In the previously mentioned constellation with the MWLP system mounted on top of a conveyor the position stamps are obtained by using a rotary encoder attached to the pulley of the conveyor belt.

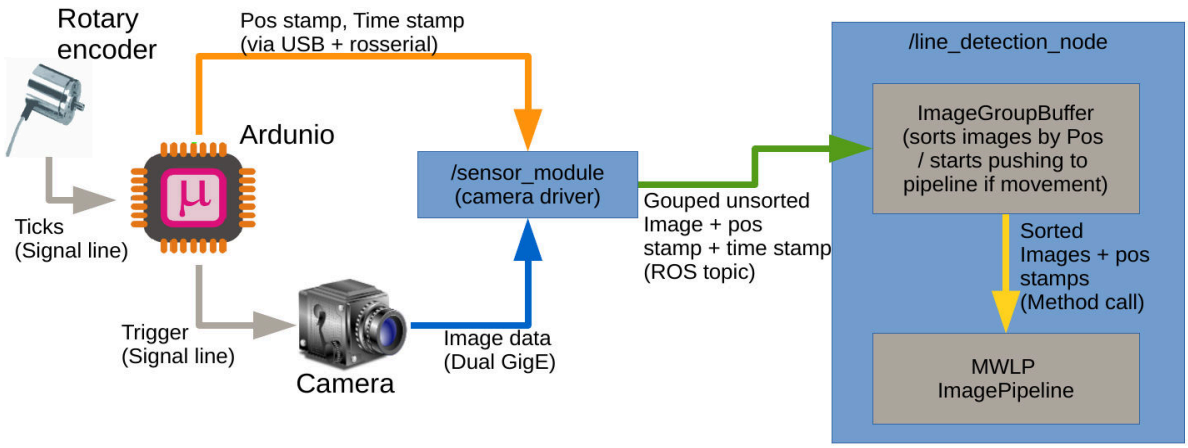


Figure 4.33: Information flow to the image pipeline if the system is operated on conveyor with rotary encoder.

The information flow of image and position stamp / time stamp data to the MWLP image pipeline with the system mounted on a conveyor with rotary encoder is drawn in Figure 4.33. The rotary encoder is connected to the Input/Output (I/O) ports of the Arduino Uno micro-controller. Consequently, the current number of rotary encoder ticks, i.e., conveyor position, is available for the Arduino at any time. As mentioned before, the Arduino further triggers the camera at a constant frame rate via the I/O ports. Hence, for each trigger applied to the camera by the Arduino a real time position stamp and time stamp is known. These are published to the ROS node `/sensor_module` which further operates as camera driver. It assembles the received image data from the camera transmitted via Dual Gigabit Ethernet (GigE) with the respective position stamp and time stamp. Further, it publishes grouped messages containing all the information for a single image to the ROS node `/line_detection_node`. The assembly of image data from the camera with position and time stamp from the Arduino micro-controller is done by matching the trigger counter of the Arduino with the hardware image counter of the camera. This assures that losing messages from the Arduino or failed image transfers from the camera over the network result only in single image groups missing, but the following groups are still assembled correctly. The `/line_detection_node` then receives the image groups. It mainly contains the multi-threaded MWLP `ImagePipeline` class for processing the image data to MWLP scans. However, due to the different transmission channels the images do not necessarily arrive sorted at this point. Therefore, the `/line_detection_node` further contains another class called ImageGroupBuffer. Its purpose is constantly buffering a set of images, sorting them by position and pushing the images in a safely defined order to the `ImagePipeline` via a class method call. Further, it assures there are no images pushed to the `ImagePipeline` if the conveyor is not moving, i.e., position stamp of consecutively captured image does not change. If the position stamp of a consecutively captured image is equal to the prior, it will just replace the respective image group in the buffer with the newly acquired one and discards the old one.

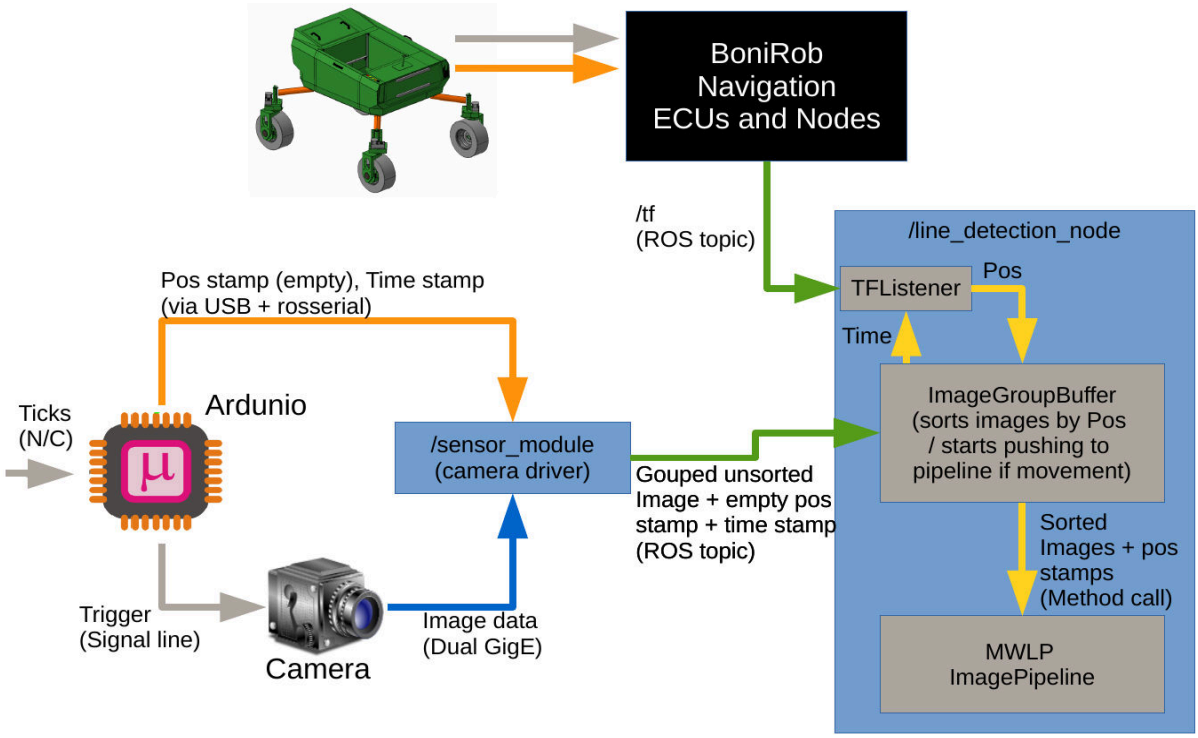


Figure 4.34: Information flow to the image pipeline if the system is with BoniRob.

I.e., only images at unique positions are pushed to the `ImagePipeline` in correct order sorted by position stamp.

In this configuration the time stamp of the image is in principle not required for the processing. It is passed on to the `ImagePipeline` and finally added unmodified as time stamp to the resulting MWLP scan that is published to other nodes. However, the `/line_detection_node` itself would work without it.

In case the system is operated inside the field robot BoniRob the information flow of the position and time information is different. The modified structure is depicted in Figure 4.34. The wheel encoders are connected to the Electronic Control Unit (ECU) and controllers of the drive train of the robot. Mounting an additional rotary encoder to one of the wheels does not appear to be a proper solution. Consequently, the 'tick' I/O ports of the Arduino of the MWLP system are not connected. Hence, the position stamps passed from the Arduino to the `/sensor_module` and further to the `/line_detection_node` are empty. The time stamps passed with the image groups representing the real time trigger time stamp are used for retrieving the position information from the navigation nodes and ECUs of BoniRob.

The ROS package `tf` is used to query the position stamp [28][29]. It allows keeping track of coordinate transformations along kinematic chains in a distributed system [27]. The naviga-

tion ECUs and nodes can thereby be seen as black box. All of them may publish individual transformations to the ROS topic `/tf` at high frequency, typically with 10 ms cycle time. Next, if a special parameter is set true before startup of the `/line_detection_node`, it internally further sets up an instance for the `tf::TransformListener` class. This class subscribes to `/tf` topic and allows querying the coordinate transformations between different frames along the kinematic chain or how they evolve over time [27]. In this case, the first image incoming to the ImageGroupBuffer is defined to be at zero position. The following images get position stamps retrieved from the `tf` query. Thereby, it is queried how the frame of the MWLP system evolved over time between the (real time) time stamp of an image and its prior image with the gobal navigation frame (`/odom` or `/odom_combined`) as observer frame. This allows filling the postion stamps of the ImageGroups inside the `/line_detection_node` and passing sorted and position stamped images to the MWLP `ImagePipeline`. I.e., the MWLP `ImagePipeline` is treated like in the constellation with conveyor.

The information flow in Figure 4.34 is more general than in Figure 4.33. It is not specified where the position information originates from. In addition, it would be possible to use this design also for the constellation with conveyor. To allow this, the ticks-information of the rotary encoder would not be assembled with the images but needed to be published to `/tf` topic at high constant frequency. This would allow always using the `tf::TransformListener` instance inside the `/line_detection_node` for gathering the position information and not relying on a parameter for selecting the position source. However, in this case, it would not be assured that the position information in the buffer of the `tf::TransformListener` does exactly match the time when the image was triggered. The listener would, therefore, in most cases interpolate between different position steps using Spherical Linear Interpolation (SLERP) [27]. This interpolation does usually not cause major differences if the `/tf` topic is available at high frequency. However, possible distortions can be avoided if the exact value is picked as done in the information flow depicted in Figure 4.33. If the mounting on the conveyor was only a test and development case and the usage inside the field robot was the only use case, it would make sense to change the design to the one shown Figure 4.34 for both cases. But as the constellation with the system mounted on top of a conveyor is a real use case and the design in Figure 4.33 is favorable for use this use case, it is preferable to keep both designs and switch by situation.

- **Matching vs. Driving modes**

Due to the nature of physics a conveyor has only one DOF while a robot navigating in a fields plane has three DOF. For the optical matching with use of the conveyor the assumption was taken that only image shifts in x-direction and y-direction are allowed. This assumption was made for reducing the search space and thereby speeding up the matching process. For the operation on the conveyor in combination the calibration and correction of the skewness between conveyor object plane and camera image plane this allows to fully represent the conveyor movements even if the camera's vertical axis is not perfectly justified with the movement direction of the conveyor. The latter would be required if the image shifts were restricted to the y-direction.

However, due to the assumptions made there not all three DOF of planar navigation (x- and y-

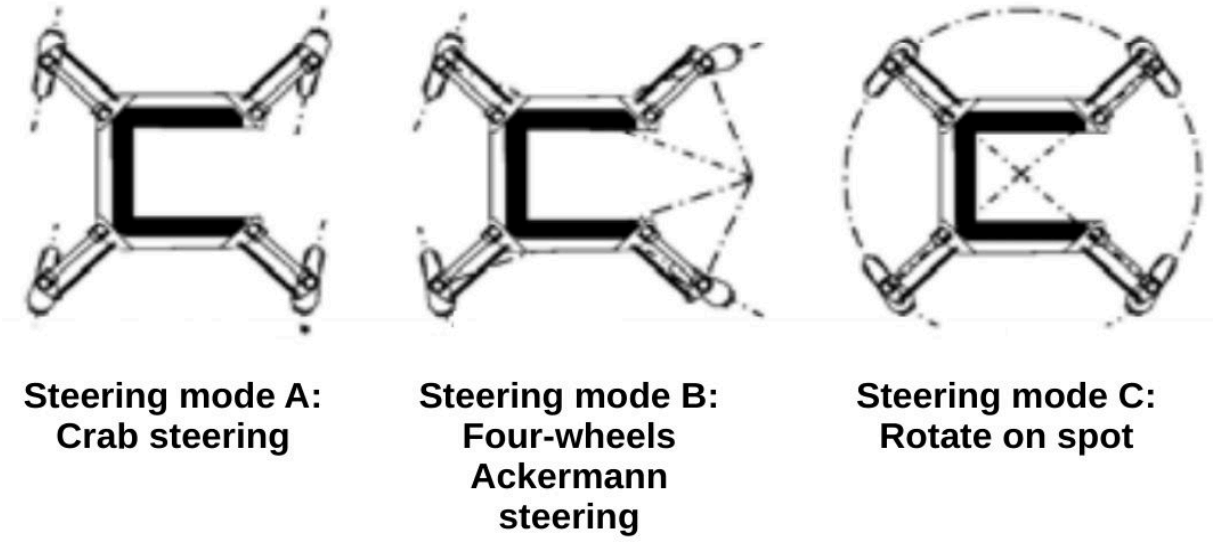


Figure 4.35: Steering modes of BoniRob so far. Source: modified from [117].

movement as well as z-rotation) can be represented without modifying the MWLP `ImagePipeline`. Fortunately, when measuring with the MWLP system the movement is fairly straight as the crop rows or ridges have to be followed. But as the crop rows cannot be assumed to be perfectly straight some (minor) steering must be allowed for the vehicle following the rows.

The BoniRob comprises four wheels that are independently steerable. The BoniRob - at the time these tests were conducted - provided three different steering modes. These are illustrated in Figure 4.35 and described as follows [117]:

A **Crab steering**
In this mode all wheels are rotated by the same steering angle. The vehicle can be moved and change its position in both planar translational DOFs without changing its orientation.

B **Four-wheels Ackermann steering**
In this mode front and rear wheels are steered together following the Ackermann condition. It results in a car-like steering with reduced turning radius. Both position and orientation can be changed simultaneously.

C **Rotate on spot**
All wheels are steered such that all of their virtual axles intersect in the vehicle center. In this mode the vehicle can rotate on a spot, i.e., change its orientation without changing its position.

For sake of completeness it is to mention that prospectively a fourth steering mode will be added to the capabilities of BoniRob in 2015/16. This mode will be the generalized approach for the above mentioned and follow the Instantaneous Center of Rotation (ICR) concept described by Schwesinger et al., 2012 [129]. However, the above mentioned steering modes will remain part

of the Application Programmers Interface (API) of the navigation of BoniRob as well as of the manual driving modes.

Coming back to the problem of mapping the matching concept of the MWLP system to the driving modes of BoniRob, steering mode A, i.e., crab steering, solves this problem. The movement takes place in the 2 planar directions that can be represented by the restricted and speeded up matching of the MWLP system. A rotation is not given, as the orientation of the vehicle does not change. This allows following rows with the BoniRob even if they are not perfectly straight without breaking with the assumptions made in order to restrict the search space of the matching step of the MWLP system. Hence, the steering mode A was chosen for the field tests conducted with the BoniRob.

4.5.2 Sample scans

This section will only show some sample scans obtained results of field trials with the MWLP prototype mounted into BoniRob. This is intended to show the feasibility of field-based application. Further, quantified results for these tests will be given in Section 5.3 where a plant/weed discrimination is tested for the data obtained during these tests. All these tests where conducted using the Pentax lens as the Schneider lens was not available due to other field tests conducted simultaneously. The speed for the scans shown here was at 0.025 m/s with full frame sampling. However, it was varied during tests as well which will be elaborated in Section 5.3.

Figure 4.36 shows a colored 3D point cloud of the scans gathered with RGB lasers at top-right. Further, in the top-left corner of Figure 4.36 a web cam view from inside the corpus of BoniRob is shown. The bottom-left corner shows the line detection, the bottom-right corner shows the incoming raw image of the Baumer camera.

Figure 4.37 shows scan data of the system obtained with RG_IR laser configuration. The left part shows a colored image. Here, the scaled *IntensitySum* values of the red and green lasers are drawn in red and green, respectively. The scaled *IntensitySum* values of the IR laser are additionally drawn in blue, hence the blue shine. The right part of Figure 4.37 depicts the obtained distance information as heat map.

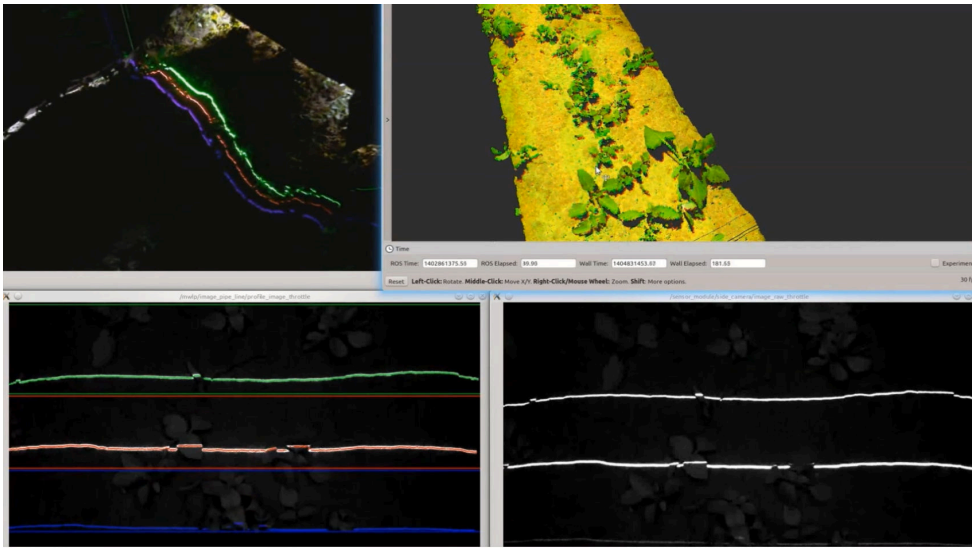


Figure 4.36: Point cloud views of carrot plants and weeds scanned in field with MWLP prototype and BoniRob.

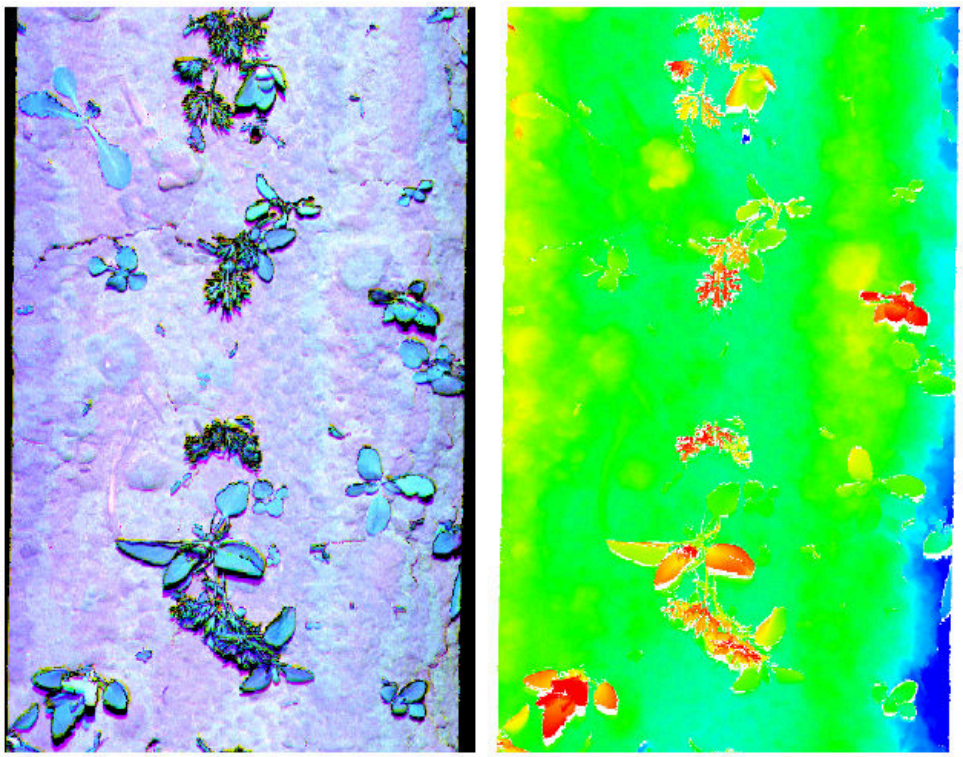


Figure 4.37: Depth map and intensity colored map of RG_IR lasers scanned form carrot plants and weeds in field with MWLP prototype and BoniRob.

Chapter 5

Classification of MWLP sensor data for agricultural applications

The previous Chapters 3 and 4 have shown the realization of the MWLP prototype as well as the validation of the derived sensor data under laboratory and field conditions. However, having in mind automated solutions scanning high-quality image-based sensor data is only a part of the process. A system-level view also includes algorithms and knowledge-bases for analysis of the captured sensor data such that qualified, automated decisions can be taken based on the captured data. Hence, classification of the sensor data is needed.

The goal of this chapter is to show that using the very sophisticated and descriptive sensor data from the MWLP-System even quite complex classification problems (such as plant classification for crop/weed discrimination) can be addressed with relatively low level (pixel-based) classification techniques. This makes the entire processing chain of sensor and classification more flexible and adaptable toward field situation specific changes compared with competing concepts with less descriptive sensor, e.g. RGB cameras, and more complex (object-based) classification algorithms.

As mentioned in Section 1.2, the usefulness of specific classification techniques cannot be shown in a generalized manner. Consequently, after the used techniques have been described the feasibility of classification of MWLP sensor data for agricultural application will be shown at the example of two applications here.

5.1 The used classification techniques

This Section will inform about the used classification techniques. These are adapted and developed with focus on agricultural applications.

There are several ways to overfit a model when conducting Machine Learning (ML). Apart from overfitting the model against the training set also using the test set for parameter tuning can introduce optimism. In particular, for agricultural applications using data of too few test situa-

tions during development can further lead to close-world assumptions were the model assumes rules valid for the known situations are valid for all situations. This will be explained in the following Digression 5.1. Readers who are familiar with this problem may continue reading after the digression.

Digression 5.1: Optimism in machine learning of agricultural applications

Optimism in the context of machine learning is a bad thing. It means that due to different kinds of overfitting the prediction errors estimated while assessing the model are to optimistic, i.e., in a real world application the model will not perform as good as expected during development.

The first and typically known step to avoid overfitting is to separate the data samples collected for development into a training set and test set. On one hand, for very simple models the prediction error of the test sample and the training samples are typically relatively similar but do not satisfy. This is because the assumptions of the model are to strong so it cannot be geared toward the underlying distribution. On the other hand, if the model is very complex, it may obtain very good results for the training data but perform weakly with the test data. This is because the model is overfitted to the training data so it cannot generalize. This issue is illustrated in Figure 5.1 [44].

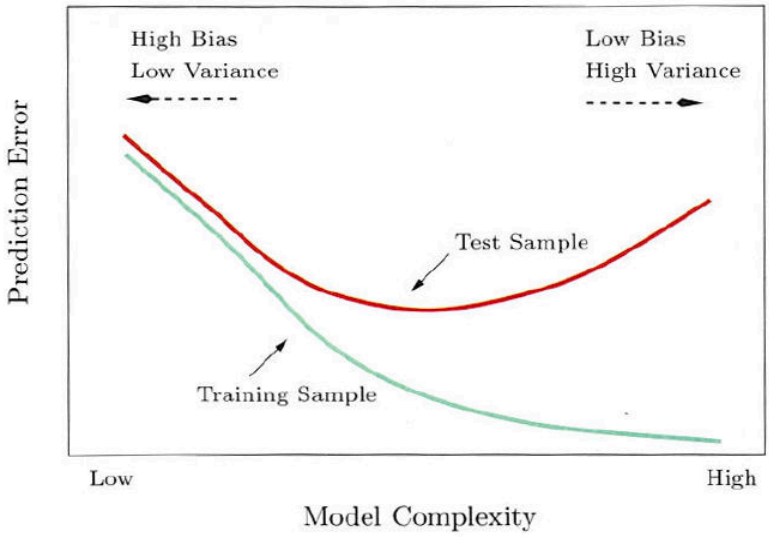


Figure 5.1: Prediction error over model complexity. Source: [44, p. 38].

However, even having in mind the separation of test and training set an unintended overfitting of the model to the test data may occur. This is a common mistake and happens if the

model parameters are tuned in order to optimize the prediction error of the test set. For complex models this hugely makes the predictions of the test samples dependent on their own reference values and, thus, introduces optimism. This is even valid for cross-validation where even the left-out values through multiple iterations and parameter tuning may influence its own prediction. This can be over come by having a third set of data samples, which is only touched ones for final assessment ('generalization set'), or nested cross validation [109].

Nevertheless, even with this in mind there is still a third way to introduce optimism which is of particular importance for agricultural applications. The field conditions in different situations often vary a lot with respect to weather, soil, plant growth stages, breed and the mounting situation of the sensor [17]. For instance, Figure 5.2 shows the deviations of different image processing parameters. The data was collected for detection of fertilizer grains. The deviations are measured by SAD and normalized, i.e., 0.0 means the distributions are completely equal, 1.0 means they are completely different. The compared distributions are distributions of the image processing parameters from samples that are labeled with the same outcome, but they are collected at different dates and/or on different fields. Clearly, a couple of distributions do drastically differ [133]. Hence, a model obtained from data at one day may not be useful at another day.

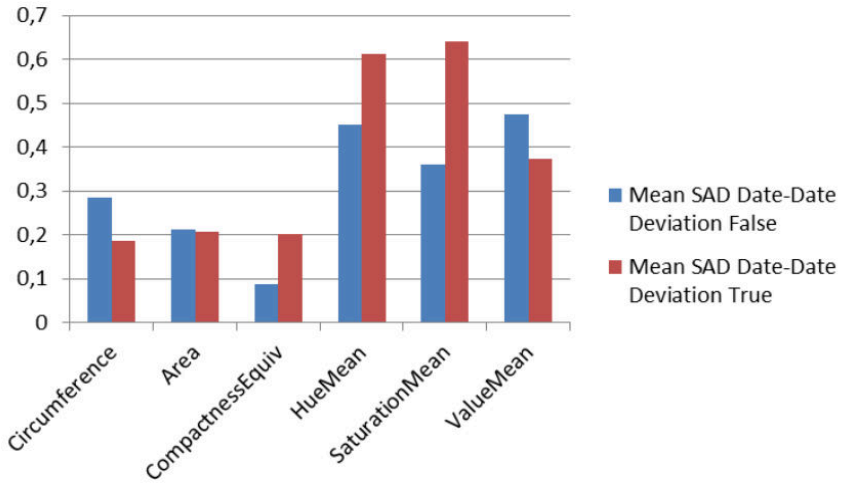


Figure 5.2: Sample deviations for sample distributions labeled with the same outcome but collected at different dates [133, p. 57]

Having this in mind even a prediction error given for a real generalization set collected at the same date and field situation like test and training set might be quite optimistic as the prediction error estimation might be valid for the respective field situation but not in general. In order to overcome this, a huge variety of training and assessment data from a huge number of different field situations has to be collected and evaluated. Alternatively,

the end user must have an option to adjust the model toward the situation he finds. In both cases life-long machine learning will - likely - be required.

5.1.1 In-Field-Labeling concept

As stated, image-based optical sensor data collected under field conditions varies a lot between different situations [46]. It is impossible to cover all thinkable field situations during algorithm development. For complex problems, this obliges the necessity of having the end user to make the final adjustments of the processing chain. This is oftentimes addressed by introducing complex parameter sets exposed on the user interface for adjustments. However, these parameters are oftentimes difficult to adjust [17]. Particularly, in the application area focused here the end users are operators of agricultural machines. They can be assumed untrained with respect to image processing systems. Hence, they are not able to parameterize complex image processing algorithms [139].

Alternatively, there are Machine Learning (ML) techniques. These allow 'abstracting away' the image processing system from the user interface. The user must only provide the algorithm with data samples and labels for generating a statistical model in a supervised ML manner. This kind of labeling is in general a simpler task, particularly as the user - again farmer/agricultural machine operator - is typically trained in inspecting and assessing plant or crop objects. He or she is usually able to mark the respective individual objects whether they belong to this or that group and using these marks the ML system can create a model for automatically taking the decision. Hence, ML systems can help to simplify the task for user interaction.

As described, using ML techniques the user interaction can be simpler compared with approaches exposing the entire algorithm parameter set to the user interface. However, ML systems require a very huge set of labeled samples in orders of thousands of samples for creating meaningful classifiers. I.e., if e.g. plants shall be classified at object level, this means hundreds or thousands of plants have to be labeled in the sensor data before the statistical model for automatic classification can be obtained. This is possible for academic purposes or during algorithm development, but is not possible for In-Field-Labeling.

In-Field-Labeling means that in order to adapt the vision system to the specific field situation the user may collect a couple of sample scans from the specific field. Next, a quick and easy way to mark the label data is required, such that a model for automatic classification can be obtained. Adjustment of processing parameter sets is not possible, as it is not easy for the machine operator. Labeling at the object level is also not possible, as labeling many - say 1000 - plants it not quick. Consequently, the In-Field-Labeling concept implies not only the process of capturing and labeling sample data in the specific field prior to operation but also that the classification is conducted in a pixel-based manner. If the classification is pixel-based, even relatively small marks in high resolution images contain ten-thousands of (pixel) samples. This means for pixel-based classification even very few - say 10 - marks provide enough data samples for obtaining a statistical model applying ML techniques.

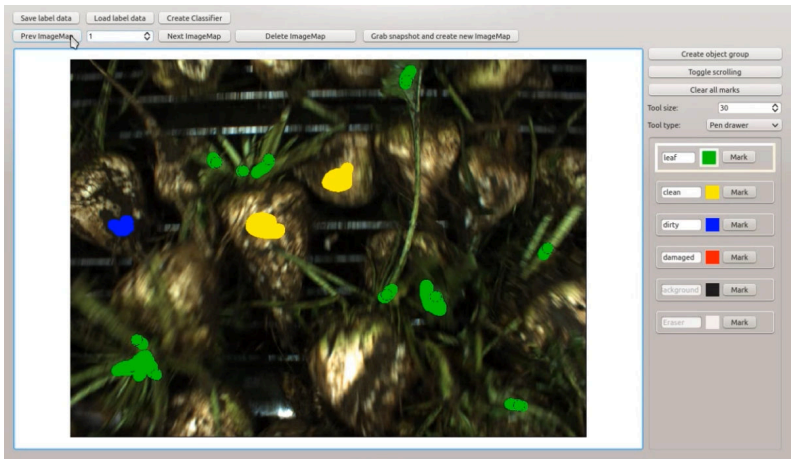


Figure 5.3: Labeling of sugar beet surfaces. The user outlines significant regions for each classified group in the respective color [139].

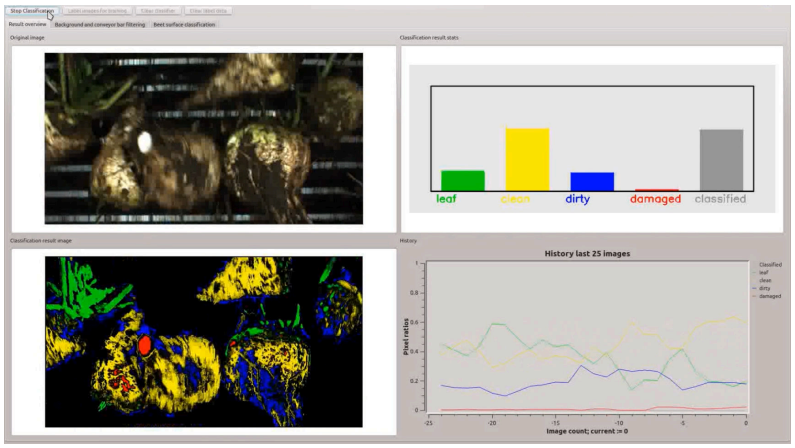


Figure 5.4: Runtime of the beet condition estimator. The algorithm automatically classifies all image pixels into the user-defined groups [139].

Pixel-based classification has its drawbacks. Shape-based patterns relying on image segmentation cannot be detected. Further, the applications were pixel-based classification is applicable on RGB camera data are very limited as only three features with 8-Bit range per sample are available. It has shown to work for determination of sugar-beet surface condition [153]. However, using the simple pixel-based classification and simple RGB camera image data complex problems such as crop/weed discrimination cannot be solved. Nevertheless, if the classification pipeline is fed with more descriptive pixel information, such as MWLP sensor data, even complex problems might be solvable using simple pixel-based classification.

The advantage of In-Field-Labeling with pixel-based classification is the quick and easy on-site generation of the classification model. This allows having up-to-date knowledge bases for the

classifier as well as recalibrating classifiers more often and, thereby, increasing their robustness against changes of ambient conditions. To illustrate the In-Field-Labeling concept further, Figures 5.3 and 5.4 are given. These show the In-Field-Labeling concept at the example of the RGB camera-based beet condition estimator [153]. Figure 5.3 depicts the labeling view. The user here picked a sample shot from the camera images on the field and outlined significant regions for each group the pixels shall be classified into. The groups for classification can be specified at runtime. Figure 5.4 shows the system during automatic classification runtime. The model was generated based on the labeled samples and is applied to classify the image pixels into the respective groups.

5.1.2 Image and label data persistence and management

As basis for pixel-based labeling and classification a system for image and label data persistence and management is required. The image data as well as the label information have to be connected at pixel level and the respective pixel labels have to be linked with the group they belong to as well as optionally additional metadata. The image map framework published in [135] was used for this purpose.

The image map framework connects label and image data with the relational data entries by saving ID-based data in an 8-Bit overlay matrix of same size like the original image data. Thereby, each object in the `ImageMap` table contains the original image data and the overlayed `ImageMapMatrix` data. Each pixel of the `ImageMapMatrix` encodes two information entries in its 8 Bit. The Most Significant Bit (MSB) states whether the pixel was actively set by the user or is assigned by an algorithm (e.g. for default values). The remaining seven bits contain the information about the assignment of the respective pixel in the original image. The pixel can be not assigned to anything. In this case, the pixel value of the original image is not taken into account when creating the classifier. The pixel may be assigned to the background, i.e., does not belong to an object of interest but to a remaining image part, that may not influence the classification result. Or it may be assigned to a particular object. In the latter case the remaining seven Bits contain the ObjectID if of the respective entry in the table `Object` that is linked to the `ImageMap` table entry the `ImageMapMatrix` belongs to. In this case, the pixel value of the image will be used as training sample for the respective group during training. This is depicted in Figure 5.5.

In this manner up to 126 objects can be labeled per image. Unlike for polygon-like drawings tool here arbitrary segments can be marked in the ImageMap and even unconnected regions can be assigned to an `Object`. These `Objects` can then be grouped into separate classification groups by linking the entries of table `Object` with those in the table `ObjectGroup`. Further, `Objects` and `ObjectGroups` can be enriched with additional metadata. Furthermore, the entries of the `ImageMap` table can be grouped by linking them to entries of the `Situation` table in order to represent different gathering situations on different fields and/or at different dates. The `Situation` entries can also optionally be enriched with metadata (Date, Global Positioning System (GPS) data etc.).

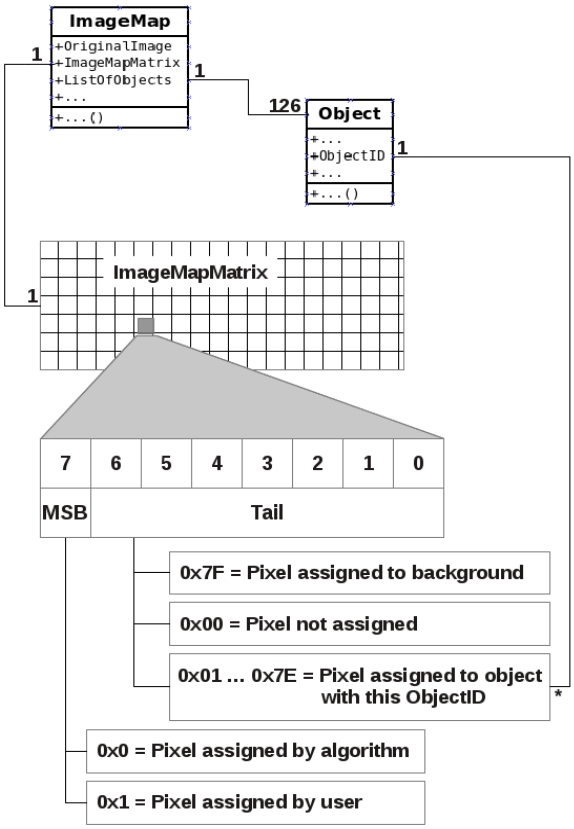


Figure 5.5: Connection of label and image data with the database-like entries [135].

In this manner the image map framework links pixel-based label data and image data with an Entity Relationship Model (ERM). This is the basis for flexible storage, persistence and synchronization of the data. The images thereby may have an arbitrary number of channels of arbitrary numeric data type, i.e., is not limited to RGB color images (3 channels, uint8).

Further flexibility of the ImageMap framework comes with the abstraction layers for front-end and the back-end. The back-end for data persistence can be provided by a PostgreSQL database [40], e.g. on servers. Alternatively, stand-alone version based on an XML back-end provides the same functionality. Files generated using the XML-based standalone version can be synchronized into the data base server. This opens possibilities for long term learning from data obtained in different situations and by different users. On the front-end side there is a web-based front end available and a native App from mobile Android devices. The most recently developed front-end used during the tests described here is the Qt-based front end.

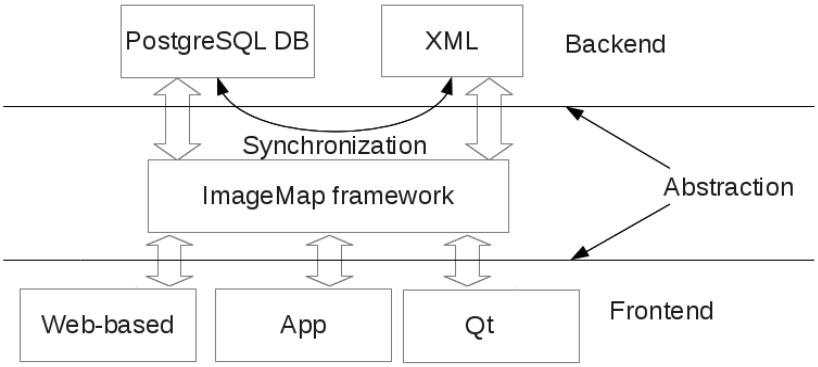


Figure 5.6: Front-end and back-end abstraction of the ImageMap data framework [135].

5.1.3 Pixel-based classification

Classification techniques for supervised ML are a vast topic. There is a variety of different classifiers such as K-Nearest Neighbors, Naive Bayes Classifier, Neuronal Networks, Support Vector Machines, Decision trees with Boosting and more [15, p. 459 - 463]. However, as stated at the beginning of Chapter 5 the goal of this chapter is not to check which classifier performs best under which condition with the MWLP sensor data but just to show that the classification of MWLP sensor data for real life agricultural problems is possible, ideally using relatively simple and primarily adaptable classification techniques.

Consequently, a Naive Bayes [85] [81] kind of classifier was used. These classifiers usually assume the features are statistically independent and Gaussian distributed. Notwithstanding these assumptions are most commonly not true, the classifier works often quite well [16, p. 362]. Therefore, it is a good 'first shot' for many classification problems.

The Bayes classifier used here was implemented for the work published in [136]. It works on histograms rather than fitting Gaussian distributions to the data. Hence, it does not require the data to be Gaussian distributed. However, statistical independence of the features is still (theoretically) required. The Bayes Classifier was implemented as described here [150, p. 26-33] and [150, p. 86-96].

During training the classifier just sums up the histograms per feature and label (i.e., `ObjectGroup` or 'background'). Further, the histograms are normalized, such that they contain relative frequencies. The second effect of normalization is that the amount of pixels labeled for each group does not matter, i.e., the user must not take care that he or she has to label the same number of pixels in one group and the other groups. In a binary case (i.e., only foreground / background classification) the obtained histograms could then look like the one given in Figure 5.7. Here - just based on this histogram - as during automatic classification a new sample comes in where this feature has the value 170 (near blue arrow), it is likely that the respective sample belongs to group 'false'. In the opposite case a sample with value 230 (near red arrow) most probably

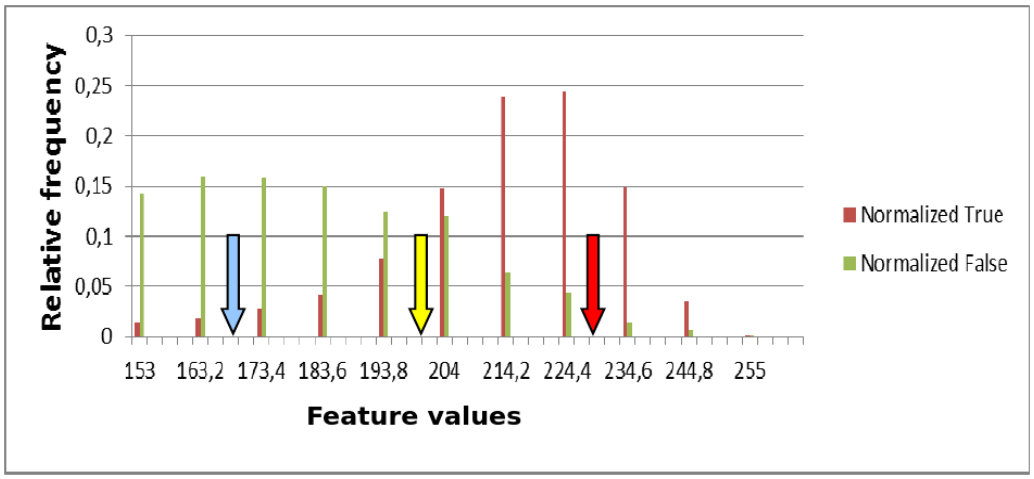


Figure 5.7: Sample histogram for a binary case [135].

belongs to group 'true'. For a sample with this feature's value at around 200 (near yellow arrow) the probability to belong to group 'true' just taking this feature into account will be near 0.5.

Now, the Bayes classifier does not only take one feature into account but many features and their respective histograms obtained during training. During automatic classification, as a new sample comes in the conditional probability of the sample to belong to a group under the condition of the sample value is obtained from the histogram for each feature. These conditional probabilities are fused using Bayes Theorem (Formula 5.1) in order to derive the overall probability of the sample to belong to each `ObjectGroup`. The detailed description of the fusing process is given in Digression 5.2.

Formula 5.1: Bayes Theorem.

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Digression 5.2: Fusing conditional probabilities using Bayes Theorem

This digression describes how the conditional probabilities derived from the histograms are fused in the Bayes filter algorithm as described here [150, p. 28-33] and here [133, p. 55-56].

In order to obtain the updated probability for a statement A to be true (e.g. 'the monitored sample belongs to `ObjectGroup` with id 2') taking into account another feature value X_j of the sample (e.g. '*IntensitySum* for Infra Red (IR) laser equals 2000') the following Formulas 5.2 and 5.3 can be applied using. They require the conditional probabilities of the feature value (measurement) under the condition of the respective outcome, e.g. $P(X_j|A = true)$.

These can be obtained from the histograms created from the label data. Further, the prior probability of statement A to be true taking into account the prior feature values $X_{1...j-1}$ but not X_j is needed, i.e., $P(A = true|X_{1...j-1})$ [150].

Formula 5.2: Updated likelihood of statement A to be true [150].

$$P(A = true|X_{1...j}) = \eta * P(A = true|X_{1...j-1}) * P(X_j|A = true)$$

Formula 5.3: Updated likelihood of statement A to be false.

$$P(A = false|X_{1...j}) = \eta * P(A = false|X_{1...j-1}) * P(X_j|A = false)$$

As mentioned above, a central assumption of the Bayes classifier is that the individual feature distributions are mutually independent. Without this assumption last term in Formula 5.2 would have to be $P(X_j|A = true, X_{1...j-1})$. However, this would be far more difficult to obtain from the training data.

η in case of Formulas 5.2 and 5.3 is a normalizer that is not known prior. However, the equations given in Formulas 5.4 and 5.5 have to be true [150].

Formula 5.4: Prior condition.

$$P(A = false|X_{1...j-1}) = 1 - P(A = true|X_{1...j-1})$$

Formula 5.5: Posterior condition.

$$P(A = false|X_{1...j}) = 1 - P(A = true|X_{1...j})$$

$P(A = true|X_{1...j-1})$ is known from the previous iteration of the Bayes classifier iteratively looping over all features taken into account. At the beginning of the iteration an a-priori known probability of the statement can be used as initial value or - if no additional information is available - it is set to 0.5. Further, $P(X_j|A = true)$ and $P(X_j|A = false)$ can be obtained from the feature distributions saved in the histograms applying the respective feature value X_j of the sample to be classified. This leaves four unknown values in the four above equations. Hence, the desired value $P(A = true|X_{1...j})$ can be derived in the four steps listed in Formula 5.6.

Formula 5.6: Bayes classifier iteration steps [150].

$$P(A = true|X_{1...j})\prime = P(A = true|X_{1...j-1}) * P(X_j|A = true)$$

$$P(A = false|X_{1...j})\prime = (1 - P(A = true|X_{1...j-1})) * P(X_j|A = false)$$

$$\eta = \frac{1}{P(A = true|X_{1...j})\prime + P(A = false|X_{1...j})\prime}$$

$$P(A = true|X_{1...j}) = \eta * P(A = true|X_{1...j})\prime$$

Iteratively applying these steps for all feature values of a sample returns the probability for the sample to belong to the respective group.

The Bayes classifier is implemented such that it can process feature values of arbitrary numeric types. For continuous wide ranged data during training the samples are first collected raw and the histograms are filled at the end of training after estimating the data range based on the sample. By now, the samples are filled into a fixed number of grid cells, e.g. 1000. During the tests with In-Field-Labeling conducted so far the number of marked pixel samples was typically in the order of magnitude between 10 kilo and 10 million samples. If there were situations in which the number of samples would drastically increase beyond 10 million samples or decrease below 10000 samples, a method for proper adjustment of the number of histogram cells would become required. Such methods for approximation of the density of continuous distribution are known as Parzen window [67] [97].

5.1.4 Feature selection techniques

Same as for classification techniques feature selection techniques are also a vast topic. However, the Minimum Redundancy, Maximum Relevancy (mRMR) approach [103] has become quite popular in recent years. Peng et al. proposed a method for automatic feature selection, where the features are selected with focus on [103]:

- **Minimum redundancy**
  The selected features should be as mutually independent from each other as possible. This is important as many classifiers assume statistical independence in their mathematical proof. Hence, adding many features that are very similar 'under different names' to the classification does typically not improve the result.

- **Maximum relevancy**
  As obvious figure, the relevancy of the selected features for the desired classification outcome should be maximized.

For using their concept a method for measuring the so-called 'mutual information' has to be provided. It measures how dependent or independent a pair of distributions is. The aggregated mutual information measures for the selected feature set is minimized for the mutual information

of the individual feature pairs. Further, the aggregated mutual information measure of the selected feature set is maximized for the individual features and the output. For categorized data deriving mutual information measures is relatively simple while for continuous data there is no generalized approach [103].

In case of the histogram-based data with categorical outcome the mutual information measure can be calculated straight forward. The SAD has been used as measure for comparing the relevancy of the different features for the outcome. The SAD is calculated by summing the absolute differences of the frequencies of the histograms for one outcome and the compared other outcome over all histogram cells. Further, the so obtained SAD is normalized by the sum of the frequencies of both histograms. Hence, the obtained value lies in the interval between 0.0 and 1.0. The case of a so defined SAD value equaling 0.0 is depicted in Figure 5.8. Obtaining a SAD value of 0.0 means that the compared histograms are equal and, hence, there is no mutual information between this feature and the outcome, i.e., the feature is irrelevant for the classification. A value of 1.0 (cf. Figure 5.9) indicates that the histograms for the outcomes can be completely separated, i.e., the outcome can be safely determined taking only this feature into account. However, usually the values lie between the boundaries (cf. Figure 5.10 for a sample distribution with SAD 0.5) signaling that there are parts of the histogram where the values are equal or similar as well as parts where the histograms differ. Typically, values around 0.75 or higher indicate that the respective feature is valuable as classification input.

As stated, selected features may have a maximum relevancy and minimum redundancy [103]. The relevancy can be measured using the so described SAD as the outcome is categorized. However, comparing different features for obtaining their mutual information is not possible in this trivial manner both value distributions might be continuous.

However, because as mentioned before this chapter mainly aims to show the feasibility of classification of MWLP sensor data but the focus of this dissertation this the MWLP sensor system. Therefore, an automatic feature selection was not implemented for this first classification step. The described SAD method was used to provide meaningful measures of the relevancy of the features. Based on these measures the features were manually selected, having in mind the issue of minimum redundancy. For the MWLP system the 'character' of each extracted feature is known (cf. Section 3.4). This is unlike for other complex ML problems where many more-less 'anonymous' features are generated and tested for their relevancy. For such anonymous features issuing for avoiding redundancy is not possible. However, for the described features of the MWLP system with known 'character' it is possible to do this 'by hand'. For instance, if for a specific classification problem scattering of a wavelength is relevant, likely all *Scatter_X_Sum* and *Scatter_X_Section* features of the respective laser will have some relevancy. Feeding the classifier with all these values will lead to high redundancy, though. Consequently, in this case, only the best of the scattering features for the relevant lasers will be manually selected to be passed on to the classifier.

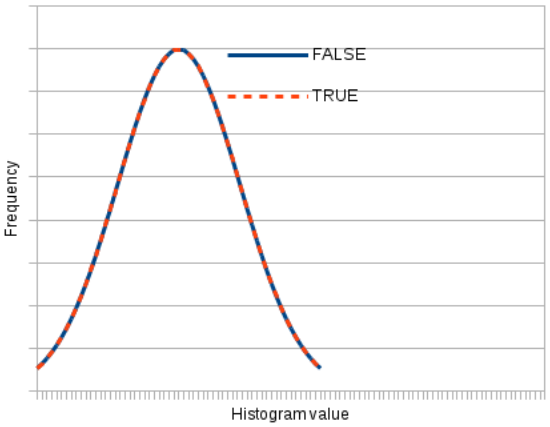


Figure 5.8: Sample histograms for SAD 0.0.

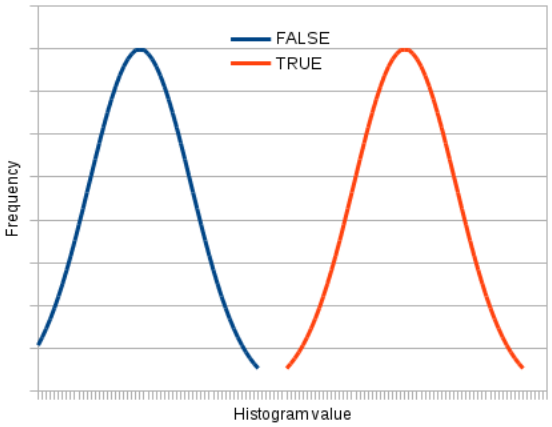


Figure 5.9: Sample histograms for SAD 1.0.

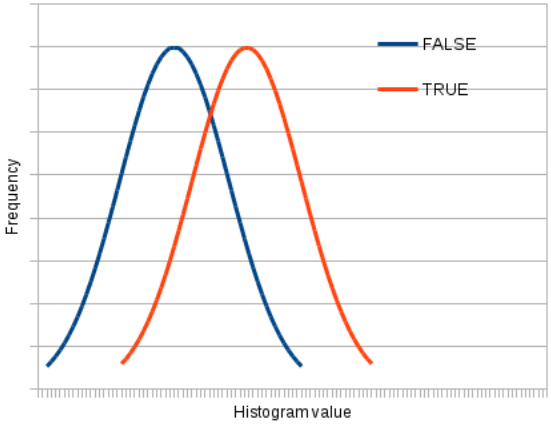


Figure 5.10: Sample histograms for SAD 0.5.

5.1.5 Classification pipeline

The implemented image classification pipeline connects the classifier and the ImageMap framework to a reusable infrastructure. Same as the classifier and the ImageMap framework it allows processing image data where the different data channels of the pixels can contain arbitrary combinations of arbitrary numeric types. Consequently, it was applied for different purposes and working on different input data streams. The beet condition estimator applied the image classification pipeline for RGB camera images [153]. The classification approaches shown here applied it for classification of MWLP sensor data.

The design of the image classification pipeline is drafted in Figures 5.11, 5.12, 5.13, 5.14 and 5.15. The boxes in light blue in these Figures is a fixed element of the pipeline. Grey input and output streams (cf. Figures 5.11 and 5.13) depend on the computational environment the image classification pipeline is applied in. For instance, these could be ROS topics for feeding the

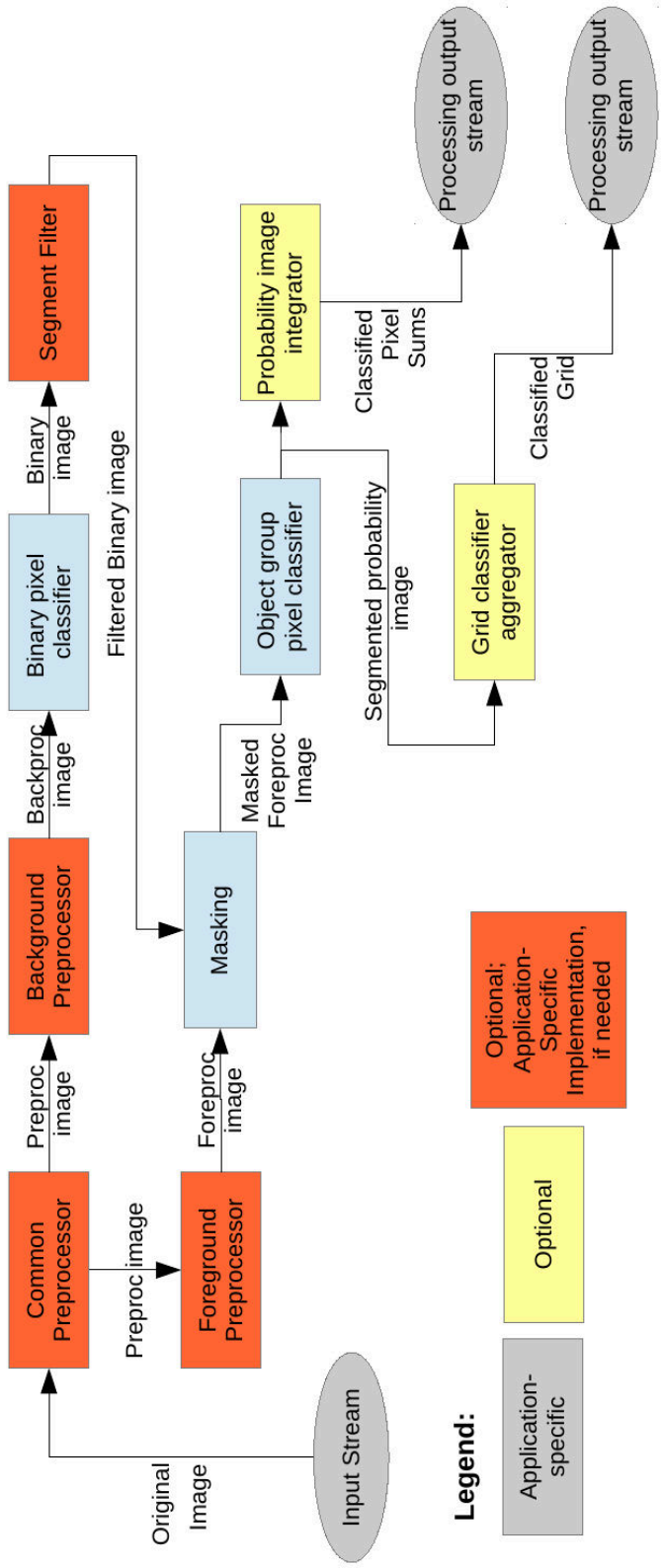


Figure 5.11: Information flow in the multi-threaded image classification pipeline during automatic classification.

pipeline with image data of different kinds. The outcome can either be a summed statistic with pixel ratios of the different groups for each image. This was the processing result for the beet condition estimator [153]. Alternatively, a classified grid can be created based on which object group primarily occupies each grid cell. This can be seen as input for actuators treating the grid cells depending on the classification result. It was used as output stream for the approaches with MWLP sensor data described in this work. An object classification is intentionally avoided as it would break with the pixel-based labeling concept and would require too much effort for In-Field-labeling.

The classification pipeline comprises two pixel classification steps. The binary classifier distinguishes between background (no object) and foreground (any object). Next, the object group classifier classifies foreground pixels into the respective object groups. For many applications the character of the background does highly distinguish from the foreground. This means that for separating the background other preprocessing steps might be required and other features might be relevant than for separating the different object groups. Further, the binary classification might be faster than the multi-class object group classification such that for images with high percentage of background the entire pipeline may run faster if the pre-separated background pixels are skipped for object group classification. For these reasons background and object group classification are separated as Figure 5.11 shows.

The image classification pipeline can be extended with optional preprocessing filters using a plug-in concept. The added filters can conduct individual image processing operations, such as e.g. blurring the image. Further, multiple filters fulfilling the plug-in interface can be grouped to a stacked filter that performs multiple operations and can be plugged in to each of the positions as a hole. The filters can be added at the following positions (cf. 5.11):

- **Common preprocessor**
  This filter performs filtering actions on the incoming original image from the input stream. The filter will affect binary and object group classification.

- **Background preprocessor**
  This filter performs filtering actions on the output of the common preprocessor before it is passed on to the binary pixel classifier. Hence, the filter will only affect binary classification.

- **Foreground preprocessor**
  This filter performs filtering actions on the output of the common preprocessor before it is passed on to masking and to the object group classifier. Hence, the filter can be used to improve the object group classification but will not affect the segmentation.

- **Segment filter**
  This filter performs filtering actions on the binary image created by the binary classifier. It can be used to improve the segmentation by options like eroding, dilating or removal of small segments.

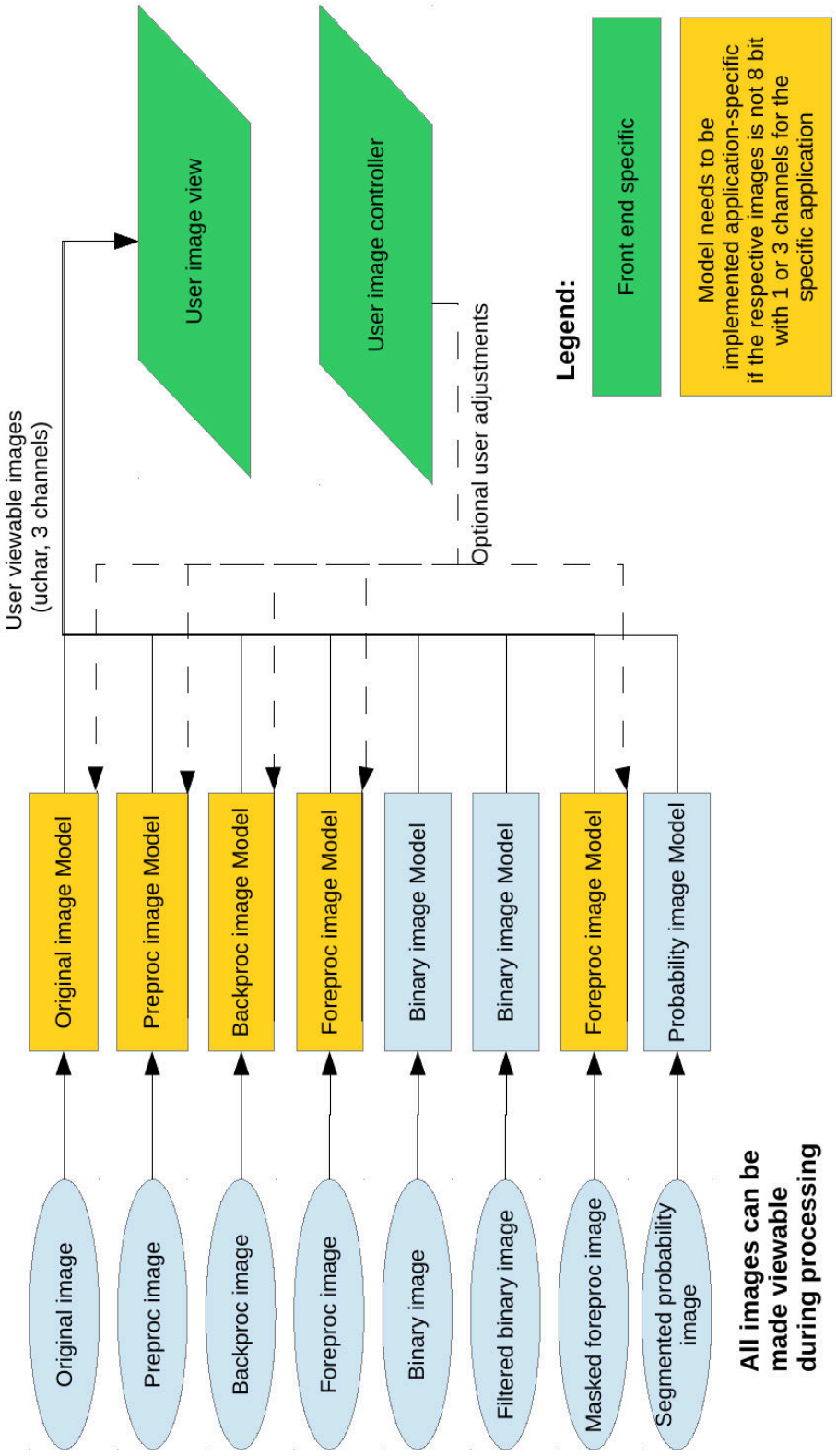


Figure 5.12: Information flow for user visualization during automatic classification.

Table 5.1: Description of the image name terms used for the image classification pipeline.

Name	Description	Type	Channels	Remarks
Original image	Original raw image as derived from the sensor data	arbitrary	arbitrary	Model for visualization required for this image; others optional
Preproc image	Image after applying some kind of preprocessing, pixel feature generation, if any	arbitrary	arbitrary	Can (but does not need to) be identical with the original image
Backproc image	Preprocessed image after emphasizing the background in any kind of way, if any	arbitrary	arbitrary	Can (but does not need to) be identical with the preproc image
Foreproc image	Preprocessed image after emphasizing the foreground in any kind of way, if any	arbitrary	arbitrary	Can (but does not need to) be identical with the preproc image
Binary image	Binary image after binary pixel classification; thresholded at probability 0.5	uchar	1	Only values 0 and 255
Filtered binary image	Binary image after applying some kind of filtering, e.g. removing small/huge regions, if any	uchar	1	Can (but does not need to) be identical with the binary image
Masked foreproc image	Foreproc image masked with filtered binary image; all 'zero' values in the filtered binary image are zero here, all non-zero pixels contain the respective pixel value of the foreproc image	arbitrary	arbitrary	Channels and type are identical with the foreproc image
Segmented probability image	Probability image with one channel per object group to be classified; each channel for the object group contains the probability of the pixel to belong to the respective group (probability scaled between 0 (0.0) and 255 (1.0))	uchar	= number of object groups	As specified by the object group pixel classifier
Overlay image with marks	Overlay image containing object IDs of marked objects	uchar	1	As specified by the ImageMap framework
User viewable image	Image for user output in BGR color space	uchar	3	Image visualization models for the other image types are required if they should be visualized and have differing type, channels or color space

During automatic operation of the classification it is still possible for the user to inspect all intermediate result images of the classification pipeline. The respective images are listed in Table 5.1. The design of visualization flow of the images to the user follows the Model-View-Controller (MVC) pattern (cf. Figure 5.12). The view showing the image to the user it not specified by the pipeline, it could be `rqt_image_view` of the ROS stack rqt [149] but other front end specific solutions are possible with the design.

As the image classification pipeline can process images with arbitrary number of channels of arbitrary numeric data the images flowing through the chain are not necessarily in a form, that can be visualized to the user. Hence, a visualization model that converts the image data into RGB images is required. In the simplest case this model can just pass the image data, if the chain is processing RGB images. If it is processing HSV images, it will just have to transform the color space. However, for more complex image data, such as chunks of MWLP scan data, more complex visualization models are required for scaling and overlaying the individual data channels. For MWLP data this can be done by the MWLP image visualization model (cf. Section 3.5.3). There is also an adaptive generic model available that can be configured by the user. However, in this case, user interaction is required for adjusting the visualization model. A controller for this purpose can be the `overlay_viz_controller` described in Section 3.5.3.

Figure 5.13 shows the information flow during labeling. The design is MVC-based, again. The front end is not specified by the pipeline, same as for the image map framework (cf. Section 5.1.2). However, as arbitrary and, hence, not necessarily viewable image data is stored in the `ImageMap` instances a visualization model is required. The default visualization model, used for the RGB camera data just passes the image data, i.e., no controller for image visualization required. For MWLP data the MWLP image visualization model with its respective controller for adjustments is again required, though.

Figures 5.14 and 5.15 show the information flow in the pipeline during training of the classifiers. The data in the image map framework is either loaded from the respective storage or might be still in memory from the previous labeling actions. Next, for training the original image is processed through the respective filtering preprocessors and passed on to the classifier along with the overlay image containing the labels.

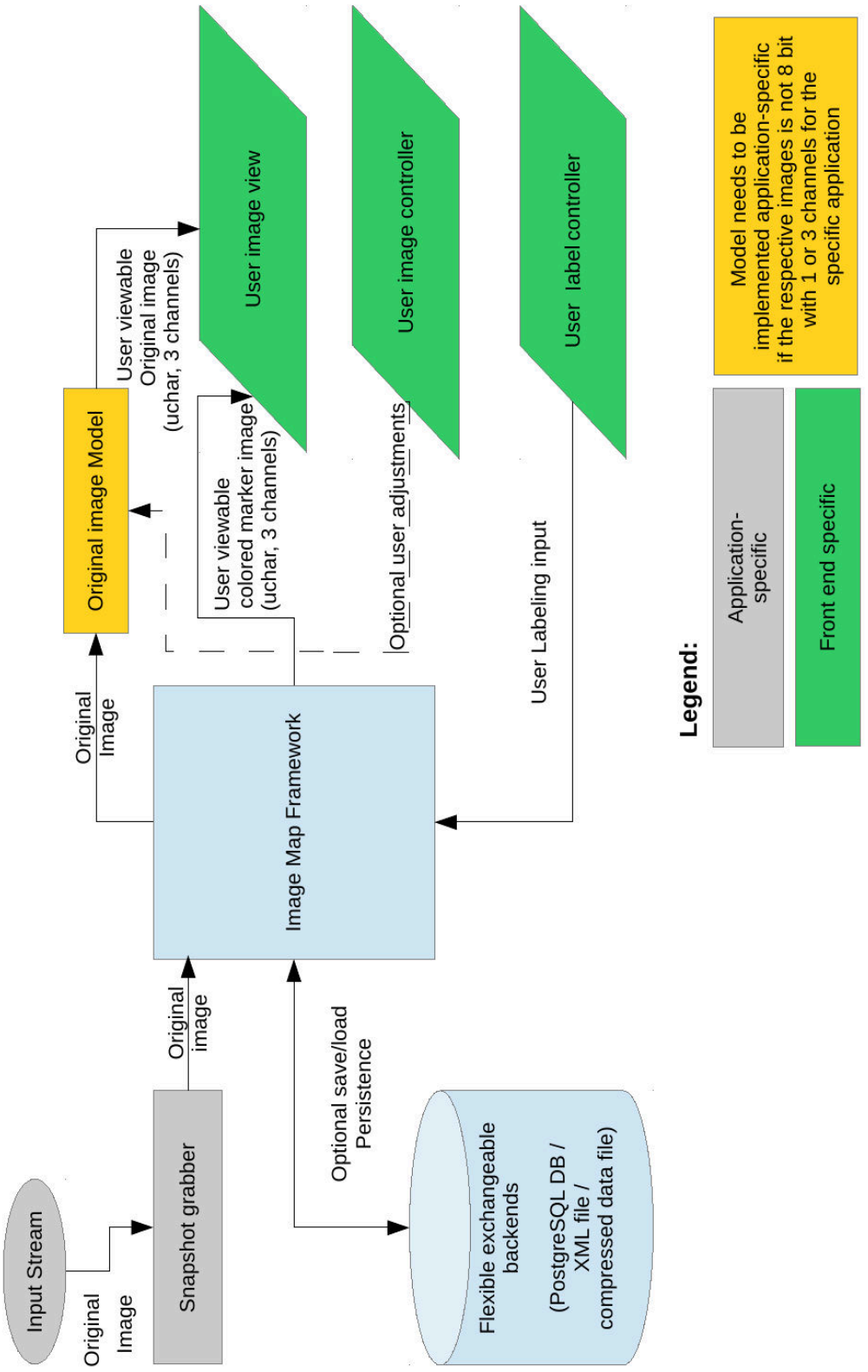


Figure 5.13: Information flow during labeling.

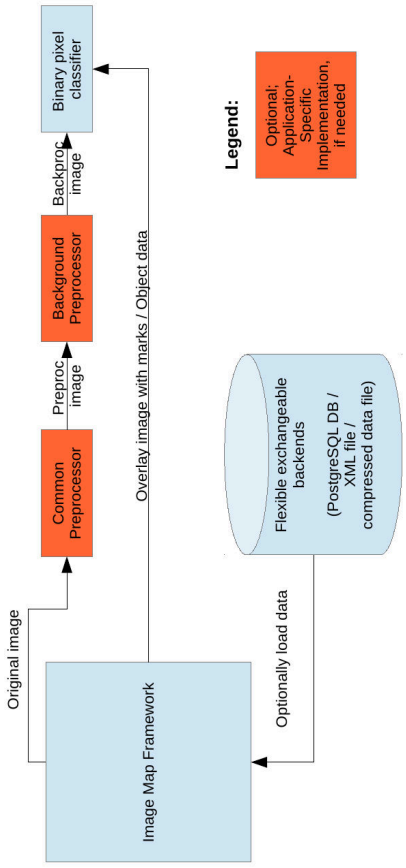


Figure 5.14: Information flow during training of background classifier.

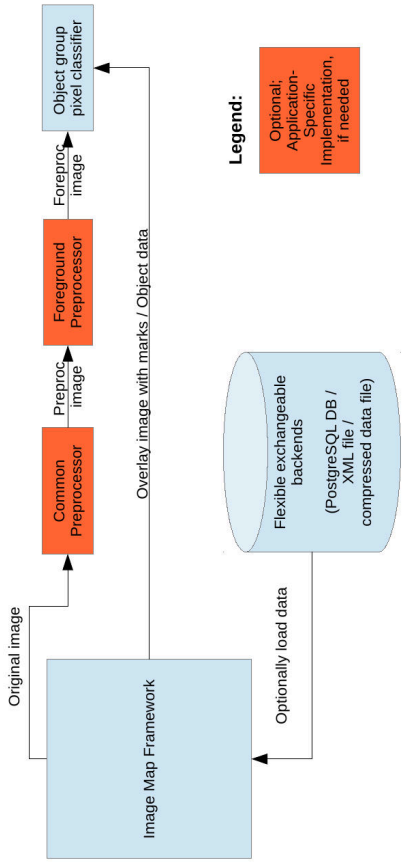


Figure 5.15: Information flow during training of foreground classifier.

5.1.6 Overlapped filtering

As stated before, the same classification pipeline was applied for the examples of MWLP classification which had been used for the beet condition estimator based on RGB camera images [153] [139]. The image classification pipeline is implemented multi-threaded, i.e., multiple images can be processed in the pipeline at the same time.

Obviously, there is a (minor) overhead for calling into the filtering methods of all filtered and piping the images through pipeline. Hence, even if the main factor for determining the computational resources required for processing the data is the number of pixels processed (i.e., frame rate times resolution) there is some influence on the performance that depends only on the frame rate. Further, the optional outputs of the processed and intermediate images to the user are triggered for each image, such that the user view gets updated for each image. In order to avoid the user's view being updated at the frame rate of the MWLP system and reducing

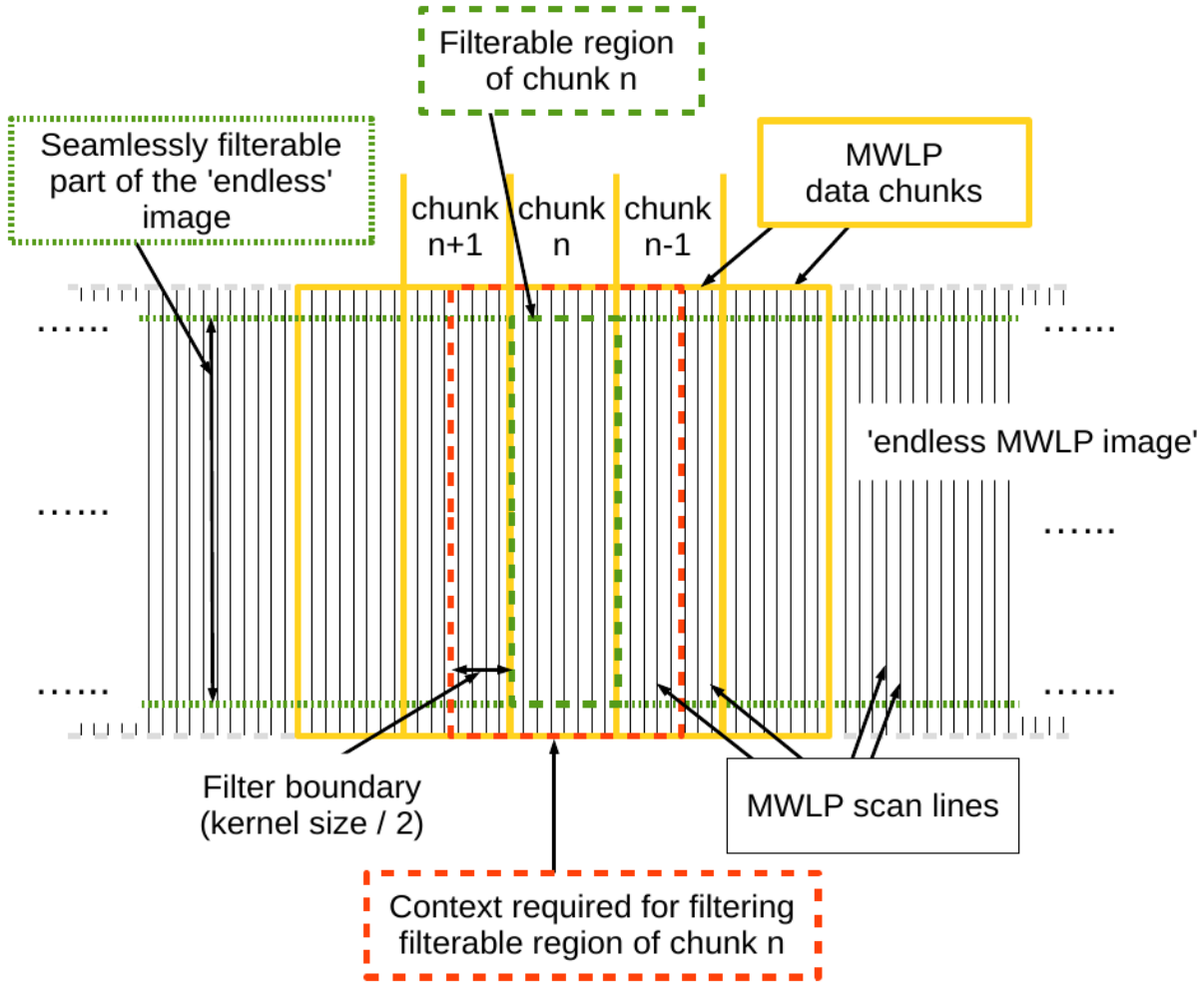


Figure 5.16: Filters can overlap the chunk boundaries.

the piping overhead, the MWLP data is not processed through the classification pipeline scan by scan but in chunks of e.g. 20 to 100 grouped and ordered scans. The chunks size has to be specified depending on the current frame rate of the MWLP system and the movement speed. If it is too small the user views will cause very high CPU load and the piping overhead will increase. However, the result of a chunk will not leave the pipeline before all containing scans are processed. Hence, the first scan of a chunk may have a too long time between scanning and known result if the number of scans per chunk is too high.

Another difference between processing MWLP sensor data and camera has to be mentioned for filtering. For processing MWLP data filters can overlap the chunk boundaries. Many filtering operations, such as erode, dilate, blur etc. or - for instance - normal estimation in distance image data, apply a kernel for filtering the pixels and, thus, require some context pixels around the pixel to obtain the result for. For a normal camera image - without application of image

stitching or similar - this context cannot be provided by other images that are processed through the pipeline parallel to the image to be filtered. Consequently, small regions at the boundaries of each image cannot be filtered (without special treatment). In contrast, for the MWLP sensor data all scans can be seen as part of an 'endless MWLP image' to which continuously new scans are pushed at to and old ones are popped at bottom. The chunk boundaries are set arbitrarily depending on the chunk size and do not represent a contextual break. Consequently, the chunks pushed to the pipeline before and after the chunk to be filtered can provide context for filtering the respective chunk. This is illustrated in Figure 5.16. Filters filtering the data of a specific chunk can overlap the chunk borders. Hence, there might be regions of the 'endless image' at the sides of the scanned area but the boundaries between subsequent chunk do not cause unfilterable regions. Thus, the filterable regions of subsequent chunks in the middle of the scan data can be seamlessly attached to each other.

The interface class `PipedGenericOverlayedImageFilterBase` provides this basic overlapped piped filtering logic. This class implements the filter plug-in interface of the image classification pipeline (cf. Section 5.1.5). It manages the assembly of multiple chunks that are parallel processed through the pipeline in a thread-safe manner. Further, it extracts the filterable region and the data required for filtering including context from its internal chunk buffer and calls a virtual method with these arguments. Hence, the child classes only have to implement the filtering operation without taking care of thread-safety and chunk-assembly as context for filtering. Consequently, the so implemented filters can be plugged into the image classification pipeline or stacked together as this can be done with context-free image filters.

5.1.7 Grid aggregation

- **Background**

The system concept that was in mind for both classification problems covered in this dissertation (potato classification and plant classification) is a combination of the MWLP system with classification and an actuator that somehow reacts according to the classification result.

In Figures 5.17 and 5.18 such systems combining the MWLP system with an actuator are drafted. Figure 5.17 plots a sorting system for potatoes from residuals based on MWLP technology. The MWLP system (#3 in Fig. 5.17) consisting of a camera (#2) and line lasers (#1) is combined with an actuator line of multiple small actuators. The MWLP system scans the objects as they are passed by on the moving conveyor and a classification system decides whether the objects can pass the sorting step or have to be rejected. If an object must be rejected the individual small actuator (#5) in line with the object kicks out the object from the product flow as it reaches the end of the conveyor. This separates the product flow into separate flows of passed ('good') and rejected ('bad') objects. Figure 5.18 depicts a similar structure applied as mechanical weeding system. In this case, the system itself is moved over the crop ridge or crop row and scans the plants, including crop plants (#7) and weed plants (#6). A classification system classifies the plants and controls the actuator line such that the individual small stamp-like actuators hit the weed plants but do not hit the crop plants.

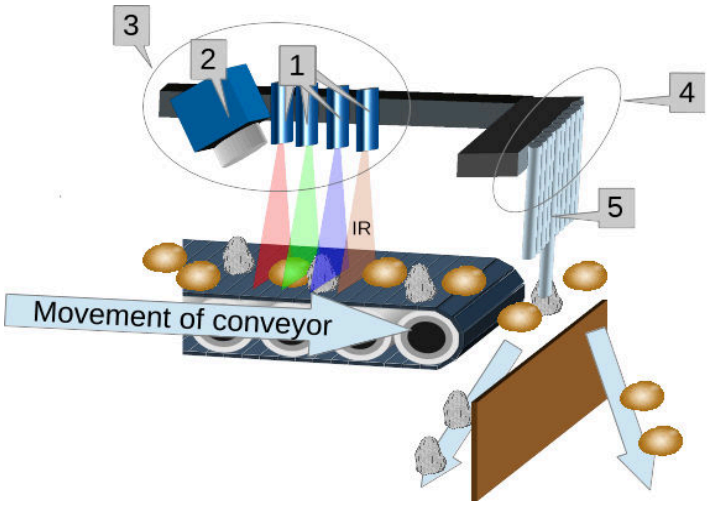


Figure 5.17: Draft of a sorting system based on a MWLP sensor system.

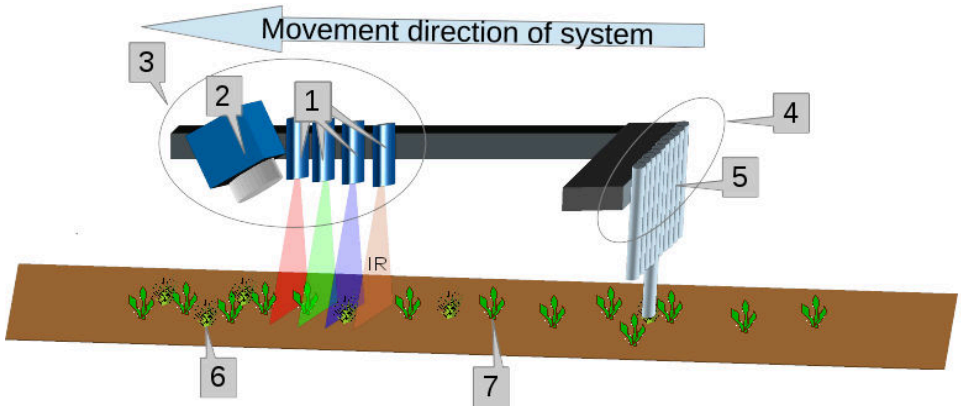


Figure 5.18: Draft of a weeding system based on a MWLP sensor system.

For sorting applications, such system structure - if not restricting the MWLP system - is very common [13]. For weeding application, there is not yet a broad adoption of such stamp-based systems. However, Langsenkamp et al. have recently shown the effectiveness of stamp-like actuators for mechanical weed control [69]. Further, the start-up Bosch Deepfield Robotics has shown a mock-up for a weeding system with a pneumatic line actuator array at the Agritechnica 2015 [113]. This shows that there is a potential for such weeding system with line actuators comprising multiple stamps.

Though, most of the classification parts are not geared toward a specific system configuration for the evaluation of the correctness of the classification this system concept was in mind. The system concept with line actuator implies that the individual actuator cannot be positioned arbitrarily as it does not comprise a manipulating arm. Therefore, a proper representation of

the system from the classification's point of view is a grid classification. Multiple pixels of the MWLP-data are summed up to a grid cell in this grid. The resolution of the grid is determined by the distance of the individual actuators in the actuator line as well as by the speed of the movement and the actuator. Hence, a grid cell size of one square centimeter appears reasonable. For each grid cell it does not matter if it is occupied by one or two or only a part of a plant but only if it has to be treated or not. This is determined by the classification. E.g. for weeding it is determined whether the grid cell is mainly occupied by pixels belonging to weeds or soil or crop plants. However, due to this constellation it is not necessary to 'detect' individual plants - or objects if sorting - as no processing happens at the object level. For the validation the automatically generated grid is then compared by a manually labeled reference grid.

- **Implementation and management of uncertainty**

As stated before (cf. Section 5.1.5), the primary outcome of the image classifcation pipeline is an image containing probabilities to belong to each of the classified classes or the background for each pixel. Consequently, in order to obtain a classified grid the different probabilities of the different pixels of each grid cell have to be aggregated somehow. In this context, also different uncertainties play an important role. First, the probability of an individual pixel to belong to a specific object group as outcome of the classification pipeline must not necessarily be 0.0 or 1.0. Values in between are also valid and the most frequent case. These value further imply an uncertainty measure. A probability of e.g. 0.9 implies a safer classification than a probability of e.g. 0.6. Further, different pixels aggregated to the same grid cell may belong to different groups and, thus, have different probability values for the individual object groups. Hence, the uncertainty in the classification has to be handled when obtaining the classification result for the different grid cells.

As the first step for the grid aggregation, the probability values of all pixels of the grid cells are averaged for each `ObjectGroup` to classify. Next, these probability values are mapped to a decision map. Finally, the grid cells are classified into treated and not treated grid cells depending on the respective decision map value. The following paragraphs will explain this procedure in detail.

In order to map the probability values of multiple `ObjectGroups` each of the `ObjectGroups` gets a `TreatmentValue` assigned. E.g. for the application of plant classification these could be as stated in Formula 5.7.

Formula 5.7: Default `TreatmentValues` for plant classification.

$$TreatmentValue_{WeedPlant} = 1$$

$$TreatmentValue_{CropPlant} = -1$$

These `TreatmentValues` are defaults, but they can be adjusted by the user via a GUI. The

Treatment value of the grid cell is then calculated as follows in Formula 5.8.

Formula 5.8: Obtaining the `TreatmentValue` for a grid cell.

$$TreatmentValue_{GridCell_k} = \sum_{i=1}^{N} \overline{P(ObjectGroup_i)} * TreatmentValue_{ObjectGroup_i}$$

$$, where\ N\ is\ the\ number\ of\ ObjectGroups\ to\ classify\ into.$$

For a grid cell where all pixels are absolutely safe to belong to the background the outcome probabilities to belong to the `ObjectGroups` of the pixel-based classification will be all 0. Hence, the `TreatmentValue` of the grid cell will be 0 regardless of the configuration. In other cases depending on the configuration `TreatmentValues` different from 0 are possible. For the configuration given in Formula 5.7 the `TreatmentValue` for a grid cell where $\overline{P(CropPlant)} < \overline{P(WeedPlant)}$ will greater than 0. In the opposite case it will be less than 0 and if both probabilities are equal, i.e., no safe classification is possible, it will remain 0.

Particularly in letter case, an assessment of the uncertainty of the grid cell classification is useful. In order to obtain a certainty measure first the probabilities of all `ObjectGroups` are summed up and the maximum value is selected (cf. Formula 5.9).

Formula 5.9: Precalculations for certainty measure.

$$\overline{P}_{sum} = \sum_{i=1}^{N} \overline{P(ObjectGroup_i)}$$

$$\overline{P}_{max} = arg \max_{[i]} \overline{P(ObjectGroup_i)}$$

$$, where\ N\ is\ the\ number\ of\ ObjectGroups\ to\ classify\ into.$$

Again, note that the probability of the background is not included in $\overline{P}_{sum}$. The background probability is implicitly represented by $1-\overline{P}_{sum}$. A certainty measure that scales between 0 and 1 can then be calculated using the steps in Formula 5.10. Ideally, all grid cells would belong to one selection (an `ObjectGroup` or the background) with a probability of 1.0. However, practically this is usually not the case. The first step in Formula 5.10 uses the probability differences between the ideal case and the practically observed value for obtaining an uncertainty measure. Thereby, a distinction of cases is required. If the most probable selection would be the background , i .e $(1 - \overline{P}_{sum}) > \overline{P}_{max}$, the uncertainty is defined by $\overline{P}_{sum}$ as it represents the difference between the probability of the selection and 1. In the opposite case probability of the selection is the probability of the most likely `ObjectGroup`, hence $\overline{P}_{max}$ has to be subtracted from 1. In

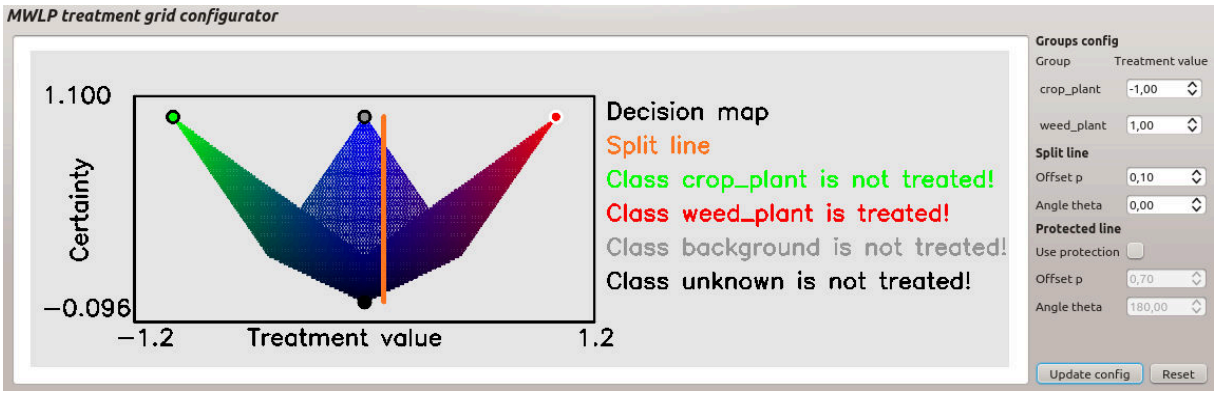


Figure 5.19: Screen shot of the grid configuration with decision map.

order to derive a normalized certainty measure scaling between 0 and 1, the maximum possible uncertainty is obtained. The maximum uncertainty represents the case where all `ObjectGroups` have the same probabilities and these probabilities also equal to $(1-\overline{P}_{sum})$, i.e., the probability of the background. In this case, all probability values are equal to $\frac{1}{N+1}$, hence this value must be subtracted by 1 in order to obtain the maximum uncertainty. The normalized certainty measure is calculated by subtracting the normalized uncertainty from 1.

Formula 5.10: Obtaining certainty measure.

$$Uncertainty = \begin{cases} \overline{P}_{sum} & , \; if \; (1-\overline{P}_{sum}) > \overline{P}_{max} \\ 1-\overline{P}_{max} & , \; otherwise \end{cases}$$

$$Uncertainty_{max} = 1 - \frac{1}{N+1}$$

$$Certainty = 1 - \frac{Uncertainty}{Uncertainty_{max}}$$

$$, where \; N \; is \; the \; number \; of \; ObjectGroups \; to \; classify \; into.$$

The decision map is spanned by drawing the possible treatment values on the x-axis and the possible certainty values on the y-axis. Figure 5.19 shows a screen shot of the configuration GUI for the grid aggregation with a plot of a decision map for the configuration given in Formula 5.7. The colored region in the center of Figure 5.19 represents the mathematically possible values of the so-constrained decision map. The bounding points of the decision map are in this example:

- `TreatmentValue = -1; Certainty = 1` –> safely crop plant
  The grid cell is absolutely safe classified to belong to the `ObjectGroup CropPlant` with a probability of 1.

- `TreatmentValue = 1; Certainty = 1` –> safely weed plant
  The grid cell is absolutely safe classified to belong to the `ObjectGroup WeedPlant` with a probability of 1.

- `TreatmentValue = 0; Certainty = 1` –> safely background (soil)
  The grid cell is absolutely safe classified to belong to the background with a probability of 1, i.e., the probabilities of both `ObjectGroup WeedPlant` and `CropPlant` are 0.

- `TreatmentValue = 0; Certainty = 0` –> absolutely no clue
  The probablities of both `ObjectGroups WeedPlant` and `CropPlant` are 0.33, hence the remaining part for the background is also 0.33. Based on the outcome of the classification pipeline the result for this grid cell is completely unknown, not even a tendency. Hence, in Figure 5.19 the black dot represents the coordinate origin.

In order to take both values of the decision map into account when making the decision whether a grid cell is to be treated or not, this decision is not taken based on a threshold for one value. It is taken a based a split line. The split line divides the decision map into a treated and an untreated region. Grid cells with probability values mapped to the decision map situated on one side of the split line are treated. If the mapped value lies on the other side, the respective cell is not treated.

The positioning of the split line allows the user pursue different treatment strategies. For instance, for mechanical weed control a one strategy may be to safely remove all weeds - condoning a certain amount of damaged crop plants. The opposite strategy may be to safely avoid damaging crop plants during weed control - condoning a certain amount of remaining weed plants. At the same time a treatment of the background, i.e., soil loosening, may or may not be intended by the farmer. Figures 5.20, 5.21, 5.22 and 5.23 show how the decision map and the split line can be configured using the GUI in order to match these different weeding strategies.

Another option that the configuration of the treatment grid aggregation offers is the setting of different `TreatmentValue` for different `ObjectGroups`. Same as the labeling tool and the classification pipeline the treatment grid aggregation can handle multiple object groups and these groups can be configured at runtime. This allows classifying the data not only in two but three or more groups. For the treatment grid, however, the intended actuator can only treat of not treat. Hence, a binary decision has to be taken at this point. However, a multi-class classification can be used as more sophisticated input for the treatment grid aggregation based on experiences for likelihood or impact of misclassification. For instance. For potato classification one could classify the residuals into different groups such as stones and soil clods. In principle, both should be separated from the product flow, i.e., treated. However, it might be more important to remove stones because they might cause damages to tools that follow in the potato processing chain. On the other hand one might want to remove soil clods only if it is very certain, that the respective object is a soil clod as misclassification with potatoes that have soil on their surface are common and the soil clods do not cause damages in the following steps. In such case it would be useful to configure the `ObjectGroups` soil clods and stones with different `TreatmentValues`, a higher for stones (i.e., more likely removal) and a lower of

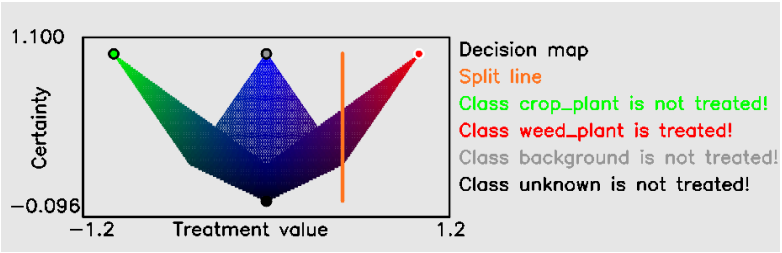


Figure 5.20: Decision map and splitting line for weeding strategy: Safely avoid damaging crop plants, no intended soil loosening.

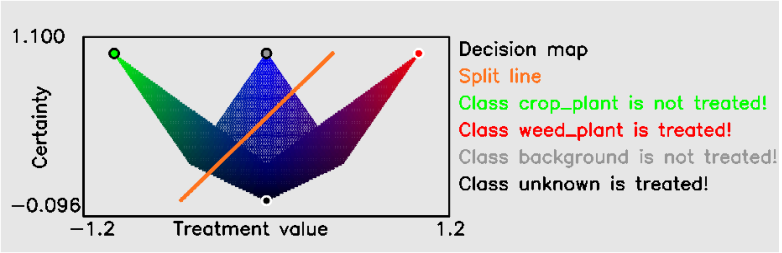


Figure 5.21: Decision map and splitting line for weeding strategy: Safely treat all weed plants, no intended soil loosening.

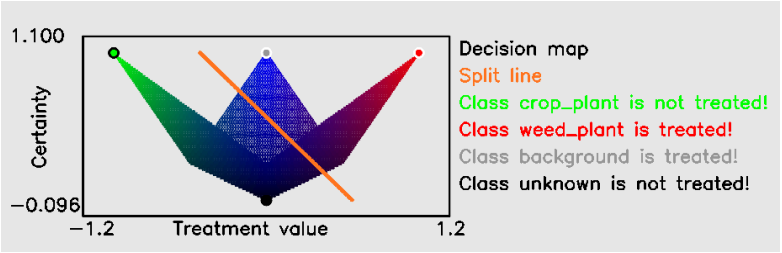


Figure 5.22: Decision map and splitting line for weeding strategy: Safely avoid damaging crop plants, with intended soil loosening.

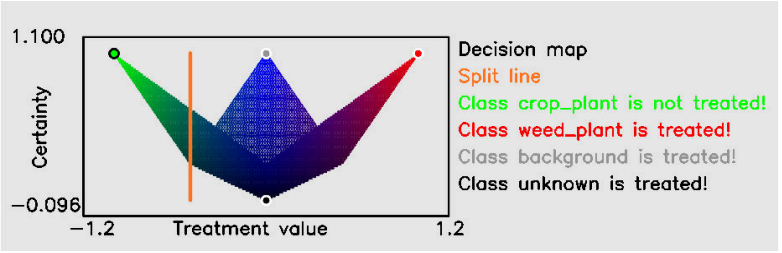


Figure 5.23: Decision map and splitting line for weeding strategy: Safely treat all weed plants, with intended soil loosening.

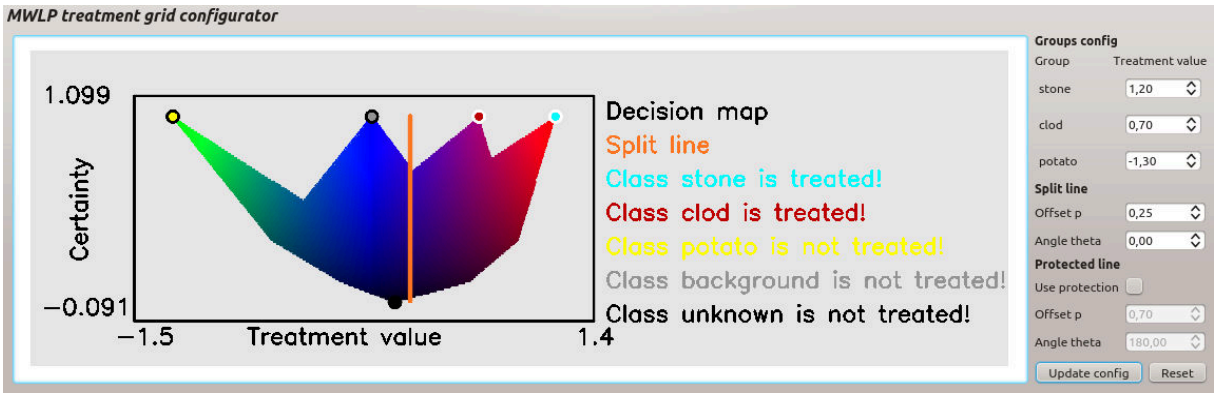


Figure 5.24: Screen shot of the grid configuration with a differentiation of the treatment values between different treated groups.

for soil clods (i.e., less likely removal). An example for this configuration is given in Figure 5.24.

The last configuration possibility that is to be discussed here is the option to create protection zones around very safely classified grid cells that may not be treated. For instance, for plant classification the grid cells that are in the center of a plant are typically safer classified than those at the plant boundary. This limits moving the split line toward unknown, i.e., for uncertainly classified grid cells because at some point weed plants may not yet be fully treated where crop plant already start being treated at their boundaries. This is where the option to specify a protection line comes in. For grid cell values where treatment value and certainty are mapped to the protected side of the protection line the cells are not treated and further their adjacent cells are forbidden to be treated regardless of their own values. E.g. for plant classification the protection line may be specified such that very safely classified crop plant grid cells are protected, i.e., not treated including adjacent cells. This allows to move the split line closer to unknown and treat even very uncertainly predicted weed plant grid cells as long as they are not protected because the protection avoids damaging to many crop plants. An example for such configuration is given in Figure 5.25.

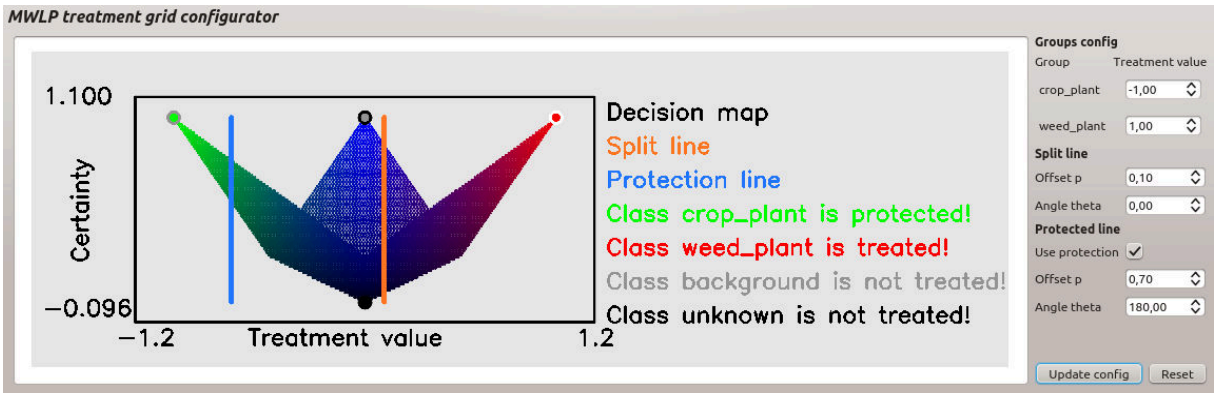


Figure 5.25: Screen shot of the grid configuration with a weeding strategy including protection of certain crop plants.

5.2 Potato classification using MWLP

The first classification application that was tested was the discrimination between potatoes and residuals conveyed with them.

5.2.1 Problem description

During the harvest process potatoes have to be separated from the soil and other objects that surrounded them during growing. Most of these material that enters the intake of the harvester can be separated from the potatoes by sieving using conveyors with web belts and/or roller/finger-based separating devices. However, most of the time some residuals remain in the product flow which cannot be separated from the potatoes in this manner. Examples are stones, soil clods, parts of crop canopy or even plastic foils/waste that have been disposed improperly on the field the potatoes are grown on.

The automatic detection and discrimination between potatoes and the residuals conveyed with them for electronically sorting them has been covered by research for many years [2] [112]. There are electronic devices for industrial use available that perform the optoelectronic separation [2] [14] [10] [9]. These typically consist of a conveyor that presents the objects (potatoes and residuals) to a sensor vision system, a vision system in including sensor and data processing for taking the decision whether an object can pass or has to be rejected and a rejection device that separates the rejected objects from the product flow [13]. The sensors are typically matrix or line cameras or photo-multipliers that might be combined with single lasers or LED lighting [2] [10] [9]. Scattering data as well as color data is of interest when analyzing potatoes [2] [41].

The field-based electronic separation of potatoes and residuals is also a problem that was solved end of the 1970s. In the former eastern part of Germany between 1976 and 1989 many machines with capacities around 25 tons per hour have been produced and sold that realized the electronic separation. However, the detection was based on X-ray technique and the increased skepticism of the consumer toward the X-rayed potatoes caused the production and application to be

discontinued [112]. Consequently, for field-based separation a manual sorting step is typically included in the process till today.

There are also non-electronic devices for field-based final separation that reach separation results better than conventional mechanic sieving or roller/finger-based separating devices. These use the differences in the densities between potatoes and typical residuals (stones, clods) for separation. The Rowadest harvester presented by the company KJK GmbH at PotatoEurope 2010 separates the potatoes in a water bath. The potatoes swim on the water surface while clods and stones sink to the ground and can thereby be separated [90, p. 326]. However, parts of crop canopy or plastics cannot be divided in this manner. The AirSep presented by the company Grimme Landmaschinenfabrik GmbH & Co.KG at Agritechnica 2013 and honored there with a gold medal [19] separates the product flow using an air flow [60]. This allows separating clods and stones (denser than potatoes) as well as leaf-like/dry canopy parts or foil (less dense/more flat than potatoes). However, still form and density remain the only discrimination criterion. Hence, the separation of objects of same form and density (e.g. waste) or defect potatoes, such as greened, damaged or diseased/decayed potatoes is still not possible. Further, the energy effort for generation of the air flow has to be kept in mind. Thus, a field-based electronic separation unit still could provide improvements in terms of discrimination flexibility and energy effort.

There was research conducted also on electronic separation for field-application. A hyperspectral imaging system was used for sensing here [33, p. 15]. However, these approaches have not yet succeeded in field-based commercial applications. Consequently, testing the novel MWLP sensor for this task is a promising approach for giving fresh impetus on this application. Particularly, the scattering and color data appear to be of interest here. The additionally delivered 3D data can further be used to improve the robustness of the system with focus on field applications.

5.2.2 Feature relevance and selection

In order to select the features for classification of potatoes and residuals, a sample data of uncleaned potatoes, stones and clods was scanned with the MWLP system. The data was passed on to the Qt-based labeling front end of the image classification pipeline and was manually marked with some labels in there. Figure 5.26 shows a screen shot of the labeling tool with potatoes and residuals. The visualization is adjusted by the `/overlay_viz_controller` to a colored representation of the *IntensitySum* values of the lasers at 532 nm and at 650 nm.

- **Classification of potatoes and residuals**

The image and label data shown in Figure 5.26 was saved to a file using the XML-based back end of the image map framework. The data was processed using a script file in order to create all normalized histograms of all contained channels and for obtaining the SAD values of all channels. Hence, the importance of all features extracted by the MWLP system for the classification problem, i.e., pixel belongs to potato, clod, or stone, was derived. The obtained SAD-based relevance measures are listed in Table 5.2.

As mentioned, it was expected that the *Scatter...*-features are very relevant for classification of

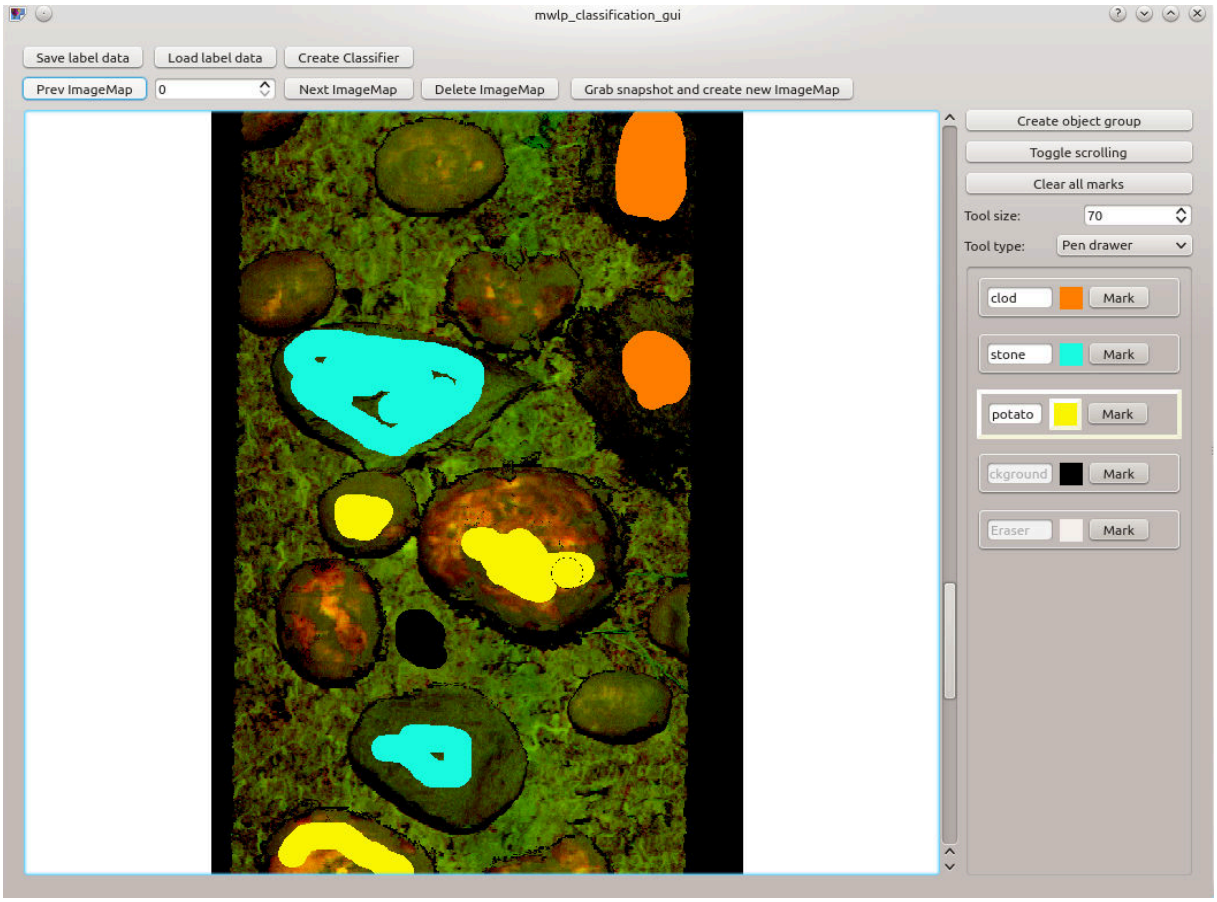


Figure 5.26: Screen shot of the labeling tool with MWLP sensor data of potatoes and residuals.

potatoes (high water content) and residuals like clods or stones (optically dense). This expectation was proven true during these tests. In Figure 5.27 the histograms of *IntensitySum* feature of laser at 650 nm are given. The labeled pixel's values are grouped into different histograms depending on the respective labels (cf. Figure 5.26). Note, the pixels belonging to potatoes can partly be separated. However, there are also overlapping regions. The normalized SAD of 0.719 shows some relevance for the output but is not as high as desirable. In contrast, the histograms of *Scatter20Section* feature of laser at 650 nm depicted in Figure 5.28 show a quite clear separation of the potato pixels from the other groups. The SAD of 0.967 proves the high relevance of this feature.

This effect was also shown to have good selectivity for the green laser (@ 532 nm) and the NIR laser (@ 850 nm). The *Scatter20Section* feature had the highest SADs for all these lasers. This is pointed out by Table 5.2 listing the SADs obtained for all channels during these tests. As stated, the redundancy between features passed on to the classifier has to be minimized (cf. Section 5.1.4). Consequently, only the *Scatter20Section* features of each of these lasers were selected as selecting more scattering features per laser would mean redundancy. The blue laser

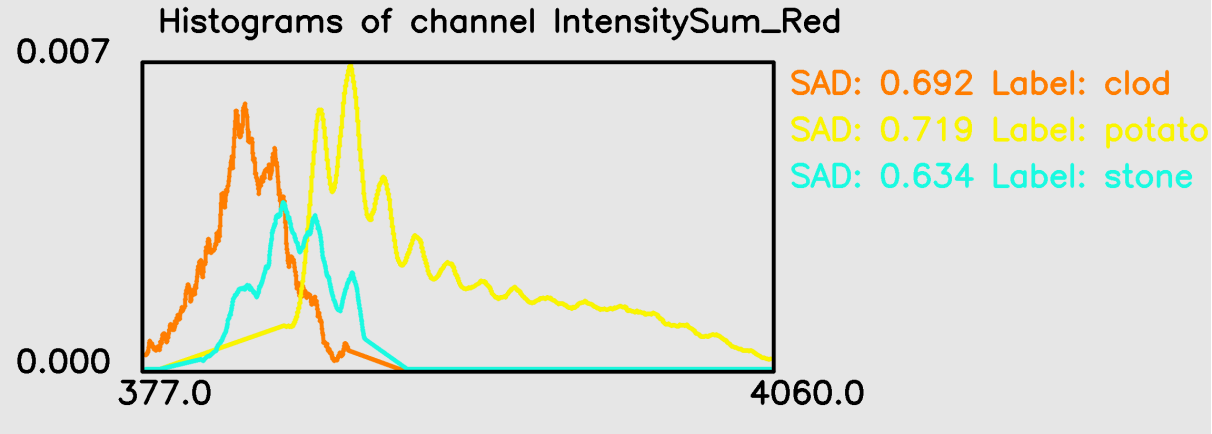


Figure 5.27: Histograms of *IntensitySum* feature of laser at 650 nm.

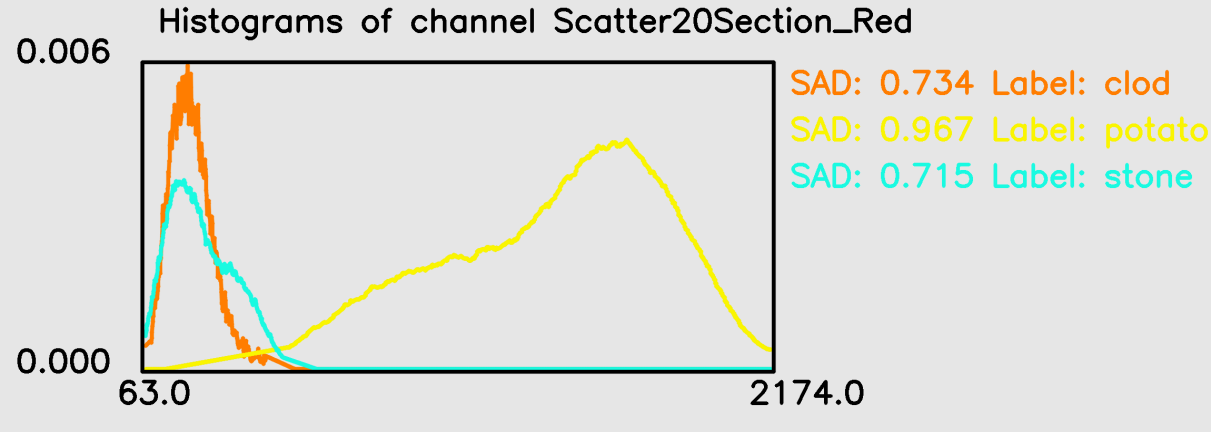


Figure 5.28: Histograms of *Scatter20Section* feature of laser at 650 nm.

(@ 405 nm), however, did not have this selectivity of the *Scatter20Section* channel nor any other of its features was relevant. Consequently, it was turned off during following classification tests.

Table 5.2: SADs of all channels extracted by the MWLP system.

Channel name	potato	stone	clod
DistMap	0.632	0.542	0.490
LineWidth_Blue	0.510	0.475	0.499
IntenMax_Blue	0.512	0.502	0.530
IntensitySum_Blue	0.244	0.596	0.606
Scatter20Section_Blue	0.533	0.506	0.592
Scatter40Section_Blue	0.508	0.507	0.483
Scatter40Sum_Blue	0.542	0.517	0.556
Scatter60Section_Blue	0.510	0.501	0.497
Scatter60Sum_Blue	0.529	0.506	0.586
Scatter80Section_Blue	0.525	0.509	0.499
Scatter80Sum_Blue	0.521	0.507	0.572
LineWidth_Green	0.442	0.333	0.581
IntenMax_Green	0.729	0.648	0.639
IntensitySum_Green	0.398	0.257	0.692
Scatter20Section_Green	**0.870**	0.620	0.635
Scatter40Section_Green	0.754	0.380	0.616
Scatter40Sum_Green	0.869	0.538	0.669
Scatter60Section_Green	0.627	0.339	0.639
Scatter60Sum_Green	0.857	0.530	0.670
Scatter80Section_Green	0.520	0.342	0.629
Scatter80Sum_Green	0.859	0.534	0.673

Channel name	potato	stone	clod
... continued ...			
LineWidth_Red	0.754	0.350	0.518
IntenMax_Red	0.786	0.440	0.463
IntensitySum_Red	0.719	0.634	0.692
Scatter20Section_Red	**0.967**	0.715	0.734
Scatter40Section_Red	0.833	0.458	0.605
Scatter40Sum_Red	0.964	0.678	0.714
Scatter60Section_Red	0.402	0.338	0.507
Scatter60Sum_Red	0.948	0.664	0.716
Scatter80Section_Red	0.392	0.334	0.352
Scatter80Sum_Red	0.955	0.671	0.715
LineWidth_NIR	0.766	0.333	0.483
IntenMax_NIR	0.694	0.463	0.446
IntensitySum_NIR	0.747	0.337	0.530
Scatter20Section_NIR	**0.963**	0.669	0.641
Scatter40Section_NIR	0.590	0.333	0.496
Scatter40Sum_NIR	0.954	0.514	0.526
Scatter60Section_NIR	0.340	0.365	0.338
Scatter60Sum_NIR	0.952	0.524	0.542
Scatter80Section_NIR	0.351	0.361	0.388
Scatter80Sum_NIR	0.955	0.520	0.537

- **Foreground / background classification**

As mentioned in Section 5.1.5, the image classification pipeline has different classifiers for object group classification and for binary classification. The details given on feature selection so far only covered the object group classification. The binary classification, i.e., classification between foreground (any object regardless if potato or residual) and background (empty conveyor), has not been mentioned in this section so far. However, using the MWLP system also providing distance data it is a no-brainer as any pixels of any object will exceed the height of the conveyor. Hence, this classification can be performed based on the distance data.

This reasoning was also proven true by the SAD-based evaluation of feature relevances for binary classification, as Table 5.3 shows. The channel *DistMap* was shown to be the most relevant with a nearly perfect SAD of 0.997. Consequently, it was selected for background/foreground classification. Note that Table 5.3 has only one column with SADs - unlike Table 5.2. This is because for binary classification or classification into only two groups the SAD values for both groups are the same. I.e., foreground and background SADs would not differ if they were listed separately.

Table 5.3: SADs of all channels extracted by the MWLP system for binary classification.

Channel name	Background / Foreground SAD
DistMap	**0.997**
LineWidth_Blue	0.147
IntenMax_Blue	0.124
IntensitySum_Blue	0.325
Scatter20Section_Blue	0.40
Scatter40Section_Blue	0.175
Scatter40Sum_Blue	0.371
Scatter60Section_Blue	0.209
Scatter60Sum_Blue	0.332
Scatter80Section_Blue	0.249
Scatter80Sum_Blue	0.291
LineWidth_Green	0.422
IntenMax_Green	0.674
IntensitySum_Green	0.373
Scatter20Section_Green	0.652
Scatter40Section_Green	0.734
Scatter40Sum_Green	0.696
Scatter60Section_Green	0.334
Scatter60Sum_Green	0.703
Scatter80Section_Green	0.268
Scatter80Sum_Green	0.700

Channel name	Background / Foreground SAD
. . . continued . . .	
LineWidth_Red	0.439
IntenMax_Red	0.701
IntensitySum_Red	0.555
Scatter20Section_Red	0.830
Scatter40Section_Red	0.760
Scatter40Sum_Red	0.807
Scatter60Section_Red	0.379
Scatter60Sum_Red	0.788
Scatter80Section_Red	0.409
Scatter80Sum_Red	0.781
LineWidth_NIR	0.631
IntenMax_NIR	0.613
IntensitySum_NIR	0.647
Scatter20Section_NIR	0.786
Scatter40Section_NIR	0.544
Scatter40Sum_NIR	0.792
Scatter60Section_NIR	0.196
Scatter60Sum_NIR	0.787
Scatter80Section_NIR	0.558
Scatter80Sum_NIR	0.776

- **Configuration of MWLP and classification pipeline for potato classification**

Table 5.4 summarizes the configuration of the MWLP system and the image classification pipeline which was selected for potato/residuals classification based on the here described analysis and was used for the experiments described in Section 5.2.3.

Table 5.4: Configuration of MWLP system and classification pipeline for potato classification.

MWLP system	
Lasers	• Green (#4 @532 nm) • Red (#6 @650 nm) • NIR (#8 @850 nm)
Image classification pipeline	
Common preprocessor	• none
Background preprocessor	• none
Foreground preprocessor	• none
Segment filter	• none
Features binary classifier	• *DistMap*
Features object group classifier	• *Scatter20Section_Green* • *Scatter20Section_Red* • *Scatter20Section_NIR*

5.2.3 Experimental results

The so-implemented potato classification was tested with the MWLP system mounted on top of a conveyor in the lab. The potatoes used were freshly harvested potatoes which had not been further processed or cleaned, i.e., including some soil adhesion. These potatoes were conveyed along the MWLP system together with stones and soil clods at different speeds. The classification worked out using the scattering features of the MWLP system at different conveyor speeds and in subsampling mode of the camera with frame rates up to approx. 400 Hz. The data gathered by the camera of the MWLP system can processed online down to the classified treatment grid by 2 computers. One is the Personal Computer (PC) mounted into the control cabinet of the MWLP system. It processes the camera images through the image pipeline of the `/line_detection_node` (cf. Figure 3.10) including matching, line detection and line assembly and publishes out MWLP scans. The scans are transferred to the second PC. The second PC aggregates the scans to chunks (running the `/scan_buffer_node`, cf. Section 3.2.2) and processes the data through the classification pipeline (cf. Figure 5.11).

- **Simple functionality test**

The resulting images of the data processing chain are given in Figures 5.29 to 5.33. Figure 5.29 shows the input data for the potato classification, i.e., the output data of the MWLP system. As mentioned, the MWLP system was operated with red, green and NIR laser. However, for the color representation shown in Figure 5.29 only the *IntensitySum* features of the green and red laser are used for colorization. Figure 5.30 shows the background (conveyor) / foreground (potatoes or residuals) classification based on the distance information of the MWLP system.

Figure 5.31 shows the probability image of the pixel classification into the different object groups.

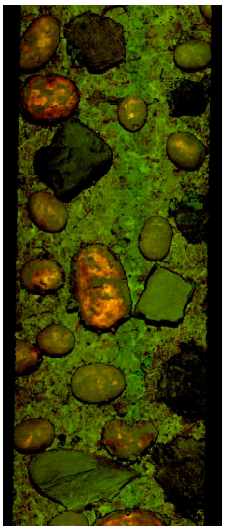


Figure 5.29: Input image: Scan data for the MWLP system.

Figure 5.30: Binary pixel classification result: Black pixels belong to background, white pixels to foreground.

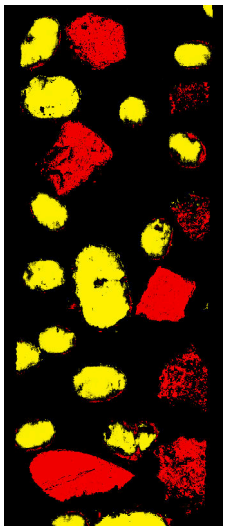


Figure 5.31: Result of pixel classification into object groups: yellow pixels are classified to belong to a potato, red pixels belong to trash. Shadings according to pixel's probabilities.

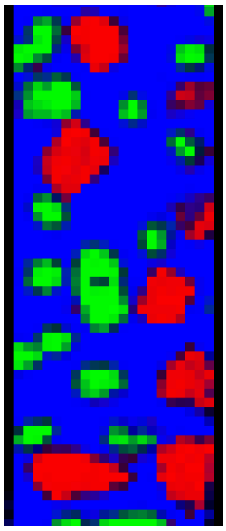


Figure 5.32: Treatment grid with the colorized values of the decision map.

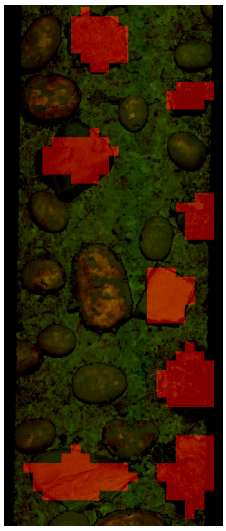


Figure 5.33: Original input image overlayed with the final treatment decision. Red shaded grid cells must be treated, i.e., stamped for separating the contained object from the product flow.

In this example the classification pipeline is trained simply to classify into 2 `ObjectGroups`, 'potatoes' and a single `ObjectGroup` 'trash' for all residuals, i.e., stones and soil clods. Thereby, each pixel is colored depending on the pixels probabilities and the colors assigned to the different object groups (here potato: yellow; trash: red) according to Formula 5.11.

Formula 5.11: Colorization of the probability image.

$$ColorPixel_k = \sum_{i=1}^{N} P(ObjectGroup_i|k) * Color_{ObjectGroup_i}$$

$$, where\ N\ is\ the\ number\ of\ ObjectGroups\ to\ classify\ into\ and$$

$$Color_{Potato} = RGB(255, 255, 0)$$

$$Color_{Thrash} = RGB(255, 0, 0)$$

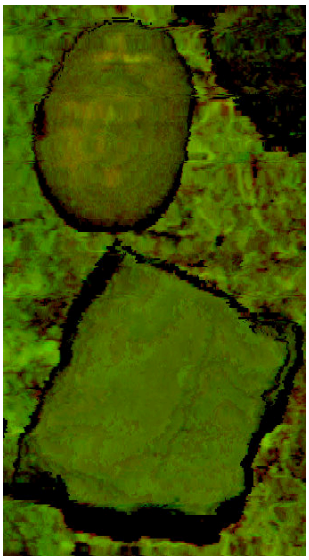


Figure 5.34: Input image for the selected region.

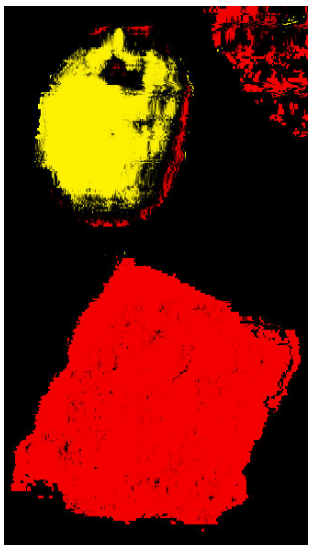


Figure 5.35: Probability image of the selected region.

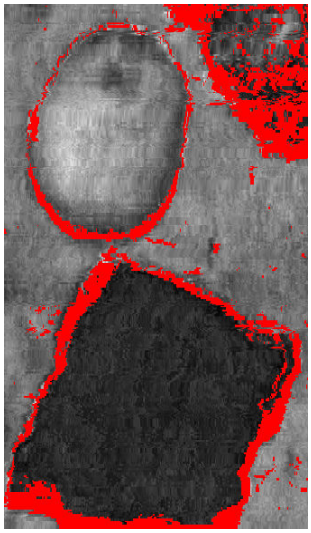


Figure 5.36: Gray-scaled *Scatter20Section_NIR* values of the selected region.

Inspecting Figure 5.31 and Figure 5.29 it is particularly notable that there are some potatoes which - due to soil adhesion - do not look very different in the color image, but they are still classified relatively good in the probability image. For instance the colors of the potato in the selected region of the original image depicted in Figure 5.34 does not look very different to the adjacent stone. Despite this, the vast majority of pixels of both objects are classified correctly as shown in detail by Figure 5.35. There are some misclassified pixels at the boundaries of the potato. However, these are very few in comparison to the number of correctly classified pixels of the potato. Hence, they do not influence the grid aggregation. The correct classification despite very similar color values is due to the feature selection for classification. Figure 5.36 shows the gray-scaled *Scatter20Section_NIR* values of the selection. Note, these features do significantly differ for both objects. As mentioned, these *Scatter...*-features do influence the classification while the *IntensitySum*-features do not. Consequently, the classification still works despite some potatoes having (little) soil adhesion.

Figure 5.32 shows the corresponding mapped grid for the image show in Figure 5.29. Here, the grid cells are colored according to the decision mapped values obtained for the respective grid cells, i.e., `TreatmentValues` and `Certainty`. Green grid cells correspond to 'not to treat' cells. Red cells have the highest `TreatmentValues`, i.e., must be treated. Blue cells represent the background. Black cells are 'unknown'. Shadings are included according to the respective `TreatmentValues` and `Certainty`. The configuration of the treatment grid generation for this experiment is given in Figure 5.37.

Figure 5.33 shows the original image (cf. Figure 5.29) overlayed with the finally generated

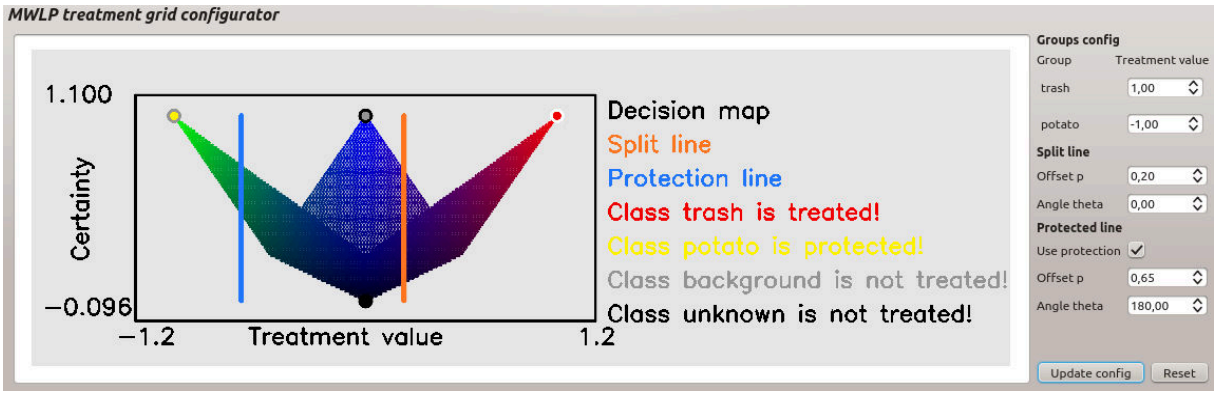


Figure 5.37: Configuration of the treatment grid aggregation.

treatment grid. The red shaded grid cells indicate the cells that have to be treated, i.e., a stamp actuator would have to hit these cells in order to separate the contained object from the product flow. As the figures show, this experiment was successful. All residual objects have cells which are to be treated. All potatoes stay untreated.

- **Verification**

After first experiments showing the functionality for potato classification had been successfully conducted, a method for verification under different conditions was needed. Unfortunately, due to lack of access to harvester machines field tests for this application were not possible. Only indoor tests could be conducted. However, there were tests conducted using freshly harvested potatoes - as mentioned before - in order to address the problem of little soil adhesion. Additionally, for verification the effects of shocks/vibrations predictable during machine operation and different conveyor speeds for increased capacity were systematically assessed in indoor measurements.

In order to assess the effects of shocks and vibrations a test bench for sensors was used. Figure 5.38 depicts the test bench. It comprises a programmable conveyor that moves the objects to test along the used sensor. The conveyor can be moved forward and backward listening on sensor feelers on both ends for repeated measurements. Further, the test bench comprises a sensor carrier into which the sensor can be mounted. The sensor carrier can be agitated up/down and tilted with different programmable amplitudes in order to monitor the effects of the sensor agitation on the measurements and image processing [128].

For the previously described experiments the potatoes and residual were conveyed as loose bulk on the conveyor. Unlike this, for experiment described here a set 6 potatoes and 6 stones/clods were separately placed and glued to a piece of wood, as depicted in the bottom left corner of Figure 5.38. This was done in order to allow operating the conveyor at its maximum speed of approximately 1.1 m/s. When the conveyor hits its end sensor, it automatically reverses its movement direction. At this point the loose potatoes do not stay in place at maximum speed.

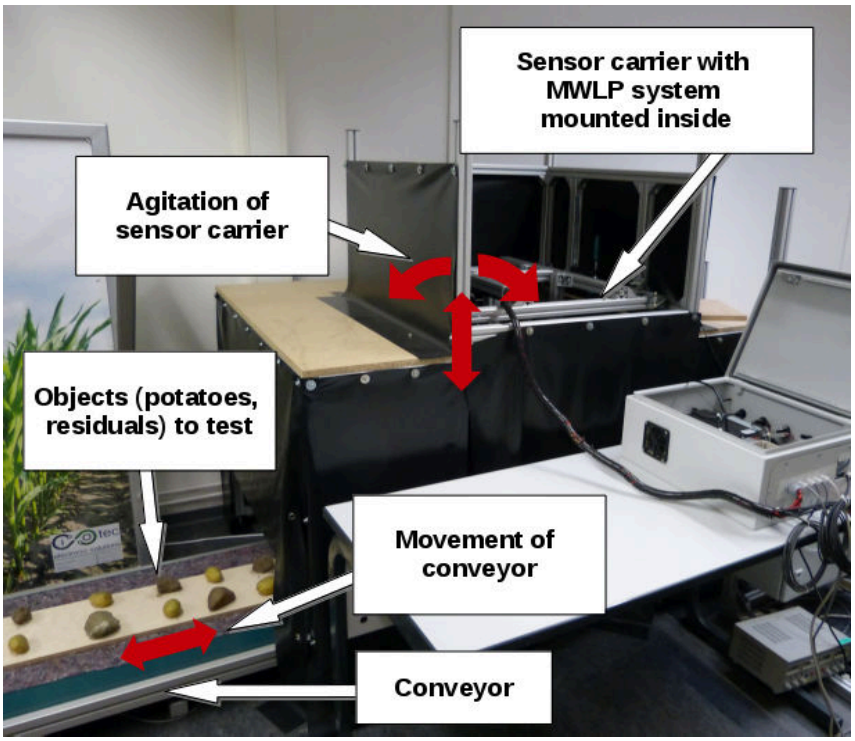


Figure 5.38: Configuration of the treatment grid aggregation.

Due to this in the previously described simple functionality tests with loose objects the conveyor speed could only be increased to approx. 0.6 m/s. In contrast, here speeds up to approx. 1.1 m/s were tested.

For this experiment the speed of the conveyor was varied in five steps between 0.12 m/s and 1.09 m/s. For each speed step the amplitude of the oscillation of the sensor carrier was varied in 6 steps between 0 mm and 25 mm. At each of these conveyor-speed/oscillation-amplitude steps the set of 6 potatoes and 6 residuals have been scanned and processed to the treatment grid at least 3 times.

Figures 5.39 to 5.44 show some example results of these operations. Thereby, like in Figure 5.33 the original data of the MWLP system visualized as color image (here including NIR laser in the blue image channel) is overlayed with the treatment grid generated as end result of the classification pipeline. Red shaded grid cells are to be treated. As the figures show, for low speeds and agitations the objects are clearly identifiable in the MWLP data and the grid cells for treatment are placed correctly. For higher speeds and moderate amplitudes this stays the case. Of particular interest is Figure 5.43. Here the objects are hardly manually identifiable. However, knowing the positions of the objects to treat it can be verified that still each of the objects to treat is overlayed by at least a couple of grid cells to treat while the potatoes stay

untreated. Only in Figure 5.44 the second residual objects from top stays completely untreated and would remain in the object flow.

Figure 5.39: Result image at speed 0.12 m/s with agitation amplitude 0 mm.

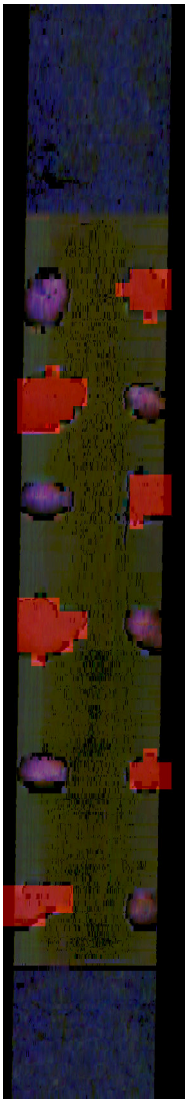


Figure 5.40: Result image at speed 1.09 m/s with agitation amplitude 0 mm.

Figure 5.41: Result image at speed 1.09 m/s with agitation amplitude 10 mm.

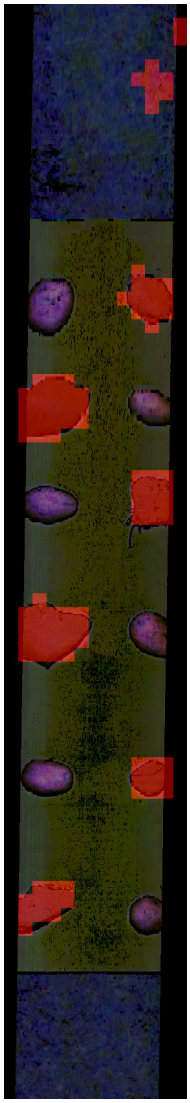


Figure 5.42: Result image at speed 0.39 m/s with agitation amplitude 15 mm.

Figure 5.43: Result image at speed 1.09 m/s with agitation amplitude 20 mm.

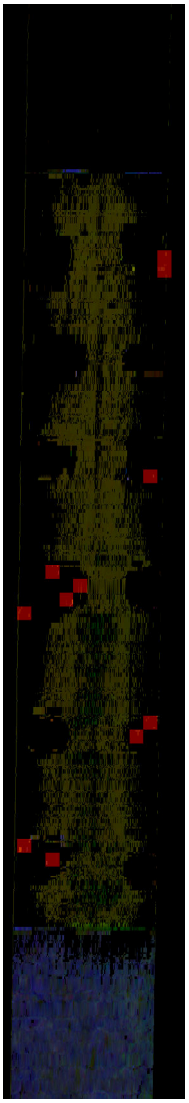


Figure 5.44: Result image at speed 1.09 m/s with agitation amplitude 25 mm.

Table 5.5 shows the summary of all of these experimental steps. The first two columns contain the information on the oscillation amplitude and the conveyor speed. The following two columns contain information on how many potatoes are scanned and how many potatoes are overlayed

with a treated grid cell. Potatoes are counted as 'with treating' as soon as then are overlayed with at least one treated grid cell. The next two columns state how many residual objects are scanned and how many of these objects are overlayed with at least one treated grid cell. The last column states how many grid cells are treated that belong to the background, i.e., the cell represents the blank conveyor without any objects. Examples for the latter kind of misclassification can be seen at the top of Figure 5.42.

The misclassification of the background are uncritical. These, in application, would only cause additional wear of the actuator tool but would not influence the process. The other kinds of misclassification, i.e., if there are potatoes with treatments or residuals without treatments, are critical and cannot be tolerated.

Summarizing the content of Table 5.5 it can be stated that the classification quality is apparently not significantly influenced by increasing conveyor speed up to the maximum speed of the used conveyor of 1.09 m/s. Agitation amplitudes of up to 15 mm cause only some background misclassification while all relevant objects stay treated or untreated correctly. Starting with amplitudes of 20 mm also misclassification of potatoes and residuals occur. Consequently, for field application an appropriate mechanical sensor mounting would be needed for avoiding oscillations between the sensor and the conveyor of more than 15 mm. However, while this would have been critical if this misclassification had started at 1.5 mm amplitude, for an allowed amplitude of approx. 15 mm this seams feasible even for a field application.

Table 5.5: Results for the potato/residuals classification at different conveyor speeds and sensor carrier agitation amplitudes.

Oscillation amplitude [mm]	Conveyor speed [m/s]	Potatoes [N objects]		Residuals [N objects]		Background [N grid cells]
		Scanned	With treating	Scanned	With treating	Treatings
0	0.12	18	0	18	18	0
	0.21	18	0	18	18	0
	0.39	18	0	18	18	0
	0.74	18	0	18	18	0
	1.09	18	0	18	18	0
3	0.12	18	0	18	18	0
	0.21	18	0	18	18	0
	0.39	18	0	18	18	0
	0.74	18	0	18	18	0
	1.09	18	0	18	18	0
5	0.12	18	0	18	18	0
	0.21	18	0	18	18	0
	0.39	18	0	18	18	0
	0.74	18	0	18	18	0
	1.09	24	0	24	24	0
10	0.12	18	0	18	18	0
	0.21	18	0	18	18	0
	0.39	18	0	18	18	1
	0.74	18	0	18	18	0
	1.09	18	0	18	18	0
15	0.12	18	0	18	18	13
	0.21	18	0	18	18	10
	0.39	18	0	18	18	22
	0.74	18	0	18	18	0
	1.09	18	0	18	18	5
20	0.12	18	**1**	18	**17**	0
	0.21	18	0	18	18	10
	0.39	18	0	18	**17**	4
	0.74	18	0	18	**17**	3
	1.09	18	0	18	18	0
25	0.12	18	0	18	**17**	4
	0.21	18	**2**	18	18	4
	0.39	18	0	18	**17**	0
	0.74	18	0	18	18	0
	1.09	36	0	36	**35**	28

5.3 Plant classification for crop/weed discrimination

After the conveyor-based potato classification based on the very descriptive scattering features had been successfully shown, first approaches toward the plant classification for crop/weed discrimination of the MWLP sensor data were conducted. This example complements the potato classification in different points. First, unlike for the potato classification here the system is mounted on a vehicle and moved itself while the objects are still during measurement. Second, from image inspection it seemed likely that the scattering features would not have the same relevance here. Thus, a different feature selection or even feature generation was needed. Third, in this case, field-based data was classified, including all the connected problems (lighting / variations etc.). Hence, this application would show different aspects and prove the descriptiveness of the MWLP sensor data from another point of view.

5.3.1 Problem description

- **Description**

An autonomous mechanical weed control system comprising a carrier vehicle, a contactless sensor system for sensing weed and crop plants, a classification system making the necessary decisions and a mechanical weeding tool for treating the identified weeds would be a very useful device. For conventional farming the use of chemical herbicides could be reduced. For organic farming selective weed control in dense crops is only possible by hand until today.

In particular, weed control in carrot cultivation in organic farming does require much manual labor [26]. The carrot as plant with relatively slow early growth stage development is subject to competing weeds and, thus, must be manually weeded to avoid major yield losses [68]. The manual weeding takes place by persons lying on vehicles removing the weeds by hand, as illustrated by Figure 5.45. This ergonomic and working conditions are harsh and the manual labor causes high costs. Further, the results of the manual weeding are usually not perfect [26]. Hence, there is a high demand for an automated solution.

Of the different required modules for an autonomous weeding system from navigation over sensing and mapping to classification and precision treatment the lack of a weed detection and classification system with practicable robustness was identified as the main limitation for development of robotic weed control systems [132].

Figure 5.45: Manual weed control in organic carrot cultivation.

- **Related work**

As it is has been identified as a key-enabling method for autonomous weed control, classification for plant/weed discrimination is a research topic with long and vast history. This subsection can only crudely strive through it. However, despite there being a long research history only very few approaches have found their way to commercial applications. Crop/weed discrimination ratios of 60 % to 95 % are frequently reported under somehow ideal conditions. However, robustness under varying conditions is oftentimes not reached, mostly due to occlusion/overlapping plants and poor segmentation under natural conditions [132].

Classical approaches commonly use camera systems, such as RGB cameras or Bi-/Multi-spectral cameras. There are also approaches using sensors like Spectral Imaging [55] [87] or 3D range sensors [159]. However, the use of cameras is far more common [46] [1] [93] [21] [155] [157] [106]. The image processing thereby is performed in 4 main steps [93] [158]:

1. **Image aquisition**
   The RGB or Bi-/Multi-spectral image is acquired from the camera. Sometimes some kinds of normalization may be applied.

2. **Segmentation / Generation of a binary biomass image**
   The soil and plant pixels are separated and grouped into segmented objects. Different algorithms can be used, such as thresholding or edge detection. For Bi-/Multi-spectral images the Normalized Difference Vegetation Index (NDVI) is frequently calculated to separate soil and vegetation.

3. **Feature extraction**
   The segments in the binary image are detected. Numeric features of the segmented objects are extracted, such as means/variances of the color/intensity values of the segment or form

features such as compactness, density with respect to convex hull, contour moments or others.

4. **Object (segment) classification**
   For a set of images the contained and up to step 3 processed images are manually labeled as ground truth. For creating a meaningful knowledge base for a classifier usually a minimum of approx. 500 to 1000 plants should be labeled into weed and crop plants. Based on the labeled data base a classifier (SVM, Neural Networks, Decision trees/forests, etc.) can then be generated that can classify segments detected other images automatically into crops and weeds.

Most of the classical camera approaches more-less follow these steps. However, these steps have drawbacks.

The first drawback is that the binary segmentation of overlapping/occluding plants into separate objects is practically not feasible [132]. This problem was tackled by Pastrana & Rath using Active Shape Models [98] [110]. Their approach is reported to work up to a certain extent of overlapping, i.e., as long as the plant organs are somehow identifiable in the binary image. For very high overlapping, the effectiveness drops, though [98]. Haug et al. go even one step further an present a "Plant classification system [...] without segmentation" [45]. Thereby, "without segmentation" indicates that the objects are not separated during classification and do not have to be separated in the biomass image. This avoids the necessity of separating objects in the binary biomass image and, thus, makes the problem of overlapping obsolete [45].

The second problem is that the camera based approaches are sensitive to poor segmentation under natural field conditions [132]. This is due to the commonly used form features (compactness / moments etc.) of the binary image for classification. As a consequence of using form features, the segmentation is of particular importance and - in the end - classification is mostly influenced by the weakly-defined pixels at the plant boundary while the pixels in the plant center do not influence the classification much. This is a common problem for practically all camera-based approaches. Even Haug et al. generate a binary biomass image and use form features (e.g. perimeter, compactness) for classification [45, p. 1144].

Likely, a fully pixel-based plant classification that does not take a binarization into account is not possible for camera data, as the pixel information generated is not descriptive enough. I.e., form features of the segmented binary plant shape are likely indispensable for plant classification in camera image data. However, this might be different for the MWLP system. Using the descriptive pixel data of the MWLP system including different intensity and scattering features along with 3D data a pixel-based plant classification might be possible. This would have a further advantage over the camera-based approaches with classification at object level. As the classification happens at pixel-level it is no longer required to label min. 500 - 1000 plant for generating a classifier but only a couple of marks are required. This opens the possibility crop/weed classification with In-Field-Labeling (cf. Section 5.1.1).

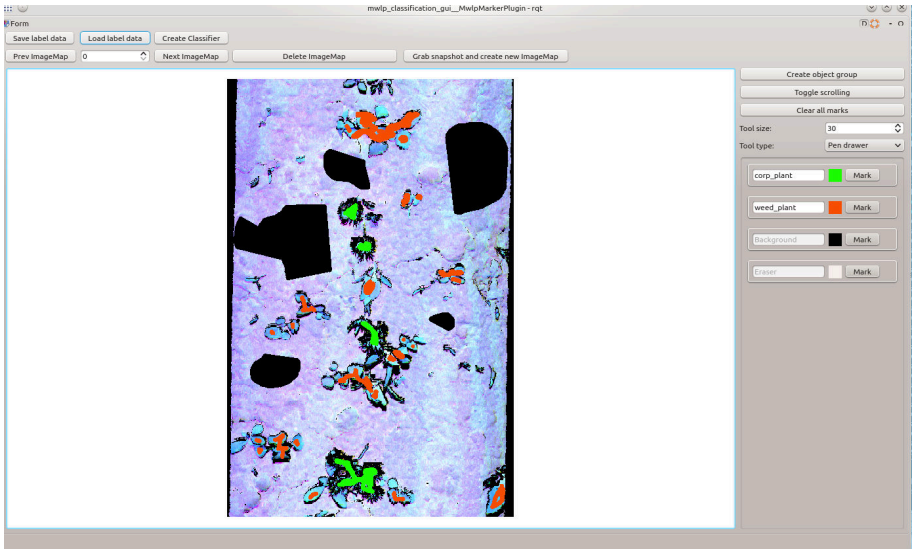


Figure 5.46: Field-acquired MWLP data with labeling marks.

5.3.2 Feature relevance, selection and generation for background removal

Before going to deep into the foreground / background classification, note that this step is not a classical segmentation / binarization. This is due to the following points. First, it is done on the pixel level, meaning a pixel assigned to the background can be surrounded by foreground pixels without issues, et vice versa. Second, it is done by classification rather than index thresholding or edge detection, i.e., is adaptable. Third, - and this is the main point - pixels identified as background pixels are only skipped for the classification into crop or weed pixels. However, the classified foreground pixels can still access their entire neighborhood and do not 'know' for feature generation whether their neighbors are foreground or background pixels. This means that there are no features for crop/weed classification generated from the foreground/background shapes. In turn, this makes the entire processing chain more robust toward poor segmentation.

The separate classification of foreground and background pixels is done to speed up the slightly slower crop/weed classification by skipping the background pixels. Further, it is necessary because the feature relevance for foreground/background (i.e., soil/plant) classification differs from the feature relevance for crop/weed classification as will be shown here.

For the labeling and classification tests the field-acquired data from carrot and weed plants scanned with the MWLP system mounted into BoniRob (cf. Section 4.5, cf. Figure 4.37) was used. It was loaded into the labeling GUI and annotated with some marks for background (soil) and crop/weed plants. A screen shot of the labeled marks is given in Figure 5.46.

The SAD-based measures of the relevance of the feature channels extracted by the MWLP system for the labeled data of Figure 5.46 are given in Table 5.6. Recall that these are scaled between 0.0 and 1.0 with 1.0 indicating the maximum relevance (cf. Section 5.1.4). Apparently,

Table 5.6: SADs of all channels extracted by the MWLP system for binary classification of plants and soil pixels.

Channel name	Background / Foreground SAD
LineWidth_Green	0.372884
IntenMax_Green	0.914066
IntensitySum_Green	0.265194
Scatter20Section_Green	0.593229
Scatter40Section_Green	0.687041
Scatter40Sum_Green	0.709044
Scatter60Section_Green	0.614667
Scatter60Sum_Green	0.74825
Scatter80Section_Green	0.418099
Scatter80Sum_Green	0.7494
LineWidth_Red	0.638308
IntenMax_Red	0.948752
IntensitySum_Red	0.75848
Scatter20Section_Red	0.310547
Scatter40Section_Red	0.352592
Scatter40Sum_Red	0.199827

Channel name	Background / Foreground SAD
... continued ...	
Scatter60Section_Red	0.354747
Scatter60Sum_Red	0.140995
Scatter80Section_Red	0.353578
Scatter80Sum_Red	0.114397
LineWidth_NIR	0.810751
IntenMax_NIR	0.93219
IntensitySum_NIR	0.700092
Scatter20Section_NIR	0.868723
Scatter40Section_NIR	0.901565
Scatter40Sum_NIR	0.912924
Scatter60Section_NIR	0.850403
Scatter60Sum_NIR	0.920386
Scatter80Section_NIR	0.654316
Scatter80Sum_NIR	0.920731

the *IntenMax* features of all lasers are of particular high relevance as well as the scattering features of the NIR laser. For the *IntenMax* there was no real explanation for this behavior. Inspecting the images they all looked very similar to the one depicted for the red laser in Figure 5.47. Apparently, the *IntenMax* values of the soil are systematically higher than those of the plants. Maybe the less dense leaf surface structure does not provide such a high reflection peak like the dense soil does. However, it has to be added that the dynamic range of these features was very small in comparison to other extracted features. Therefore, and in order to avoid redundancies between the features it was decided to use only the *IntenMax* features of one laser for the classification.

The scattering features of the NIR laser are also of high relevance. This can be explained by the light being scattered inside the leaf tissue. In particular, the light of the NIR laser can enter the tissue and, thus, is subject to scattering. Hence, one of scattering features of the NIR laser should be included for the foreground / background classification, but not multiple to avoid redundancy.

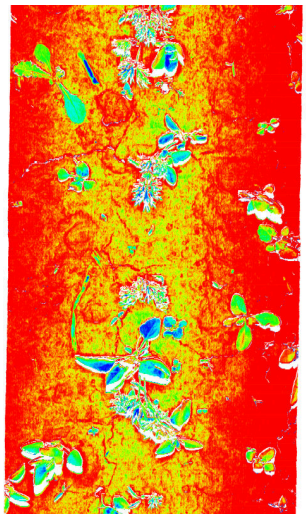


Figure 5.47: Heat map visualization of *IntenMax_Green*.

- **Feature generation**

Despite already having some good features, feature generation was also tested. The NDVI is known for being a good descriptor for soil and canopy discrimination. Therefore, it was also generated based on the *IntensitySum* features of the red laser at 650 nm and of the NIR laser at 850 nm. The NDVI is calculated according to Formula 5.12.

Formula 5.12: Calculation of the NDVI.

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

The gray-scaled *IntensitySum_Red* and *IntensitySum_NIR* values serving as input for NDVI calculation are depicted in Figures 5.48 and 5.49, respectively. Invalid pixels of the MWLP system, i.e., not scanable, are colored red in these images. For bi-/multi-spectral camera images it is usual that the reflection of soil exceeds the reflection of plants for the red waveband. For NIR the opposite case can be observed. This effect is also valid for the MWLP system. The calculation of the NDVI then creates an image with enhanced contrast by combining both pixel information. The resulting gray-scaled NDVI image for the field-based MWLP data is shown in Figure 5.50.

Further, the histograms for obtaining the SAD-based feature relevance measures mentioned in Table 5.6 are given for the channels *IntensitySum_Red* and *IntensitySum_NIR* as well as for the newly generated NDVI channel in Figures 5.51, 5.52 and 5.53. Note, by combining both channels with moderate relevances of 0.700 and 0.758 a new channel with very high relevance of 0.925 can be generated.

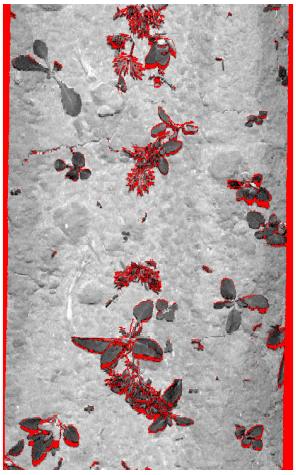


Figure 5.48: Gray-scaled *IntensitySum_Red*.

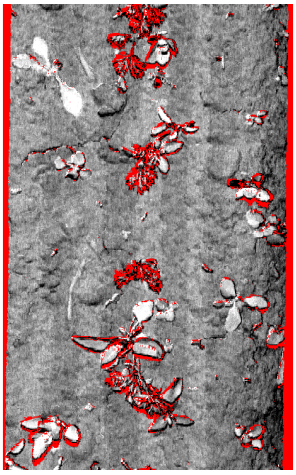


Figure 5.49: Gray-scaled *IntensitySum_NIR*.

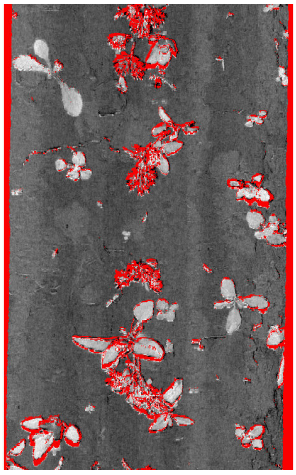


Figure 5.50: Gray-scaled NDVI.

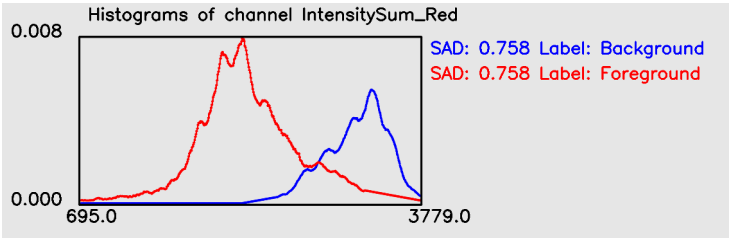


Figure 5.51: Foreground/Background histogram for *IntensitySum_Red*.

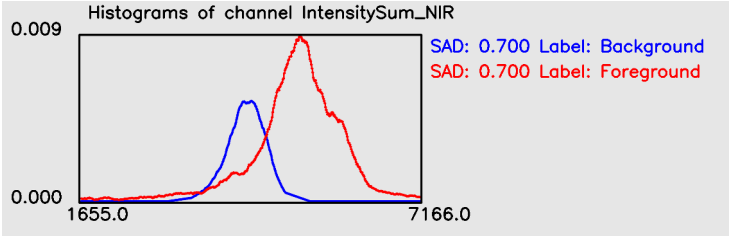


Figure 5.52: Foreground/Background histogram for *IntensitySum_NIR*.

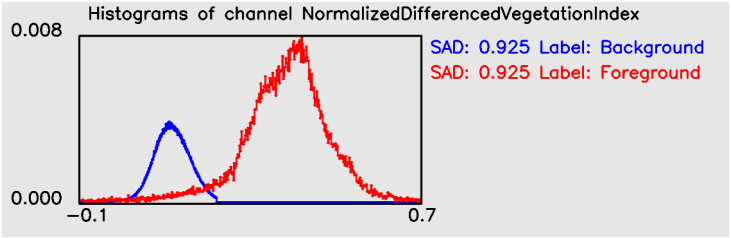


Figure 5.53: Foreground/Background histogram for NDVI.

Based on these observations, it was decided to generate the NDVI as additional feature for foreground / background classification. Further, the *Scatter60Sum_NIR* feature was selected as a relevant NIR scattering feature to be passed on to the binary classifier. Additionally, the *IntenMax_Green* was selected. It represents similar relevance like *IntenMax_Red* and *IntenMax_NIR*. However, the red and NIR lasers already influence the classifier via NDVI. I.e., for the green laser's value less redundancy was expected. For the calculation of the NDVI a filter class was implemented, such that could be plugged into the classification pipeline as background preprocessor (cf. Section 5.1.5).

5.3.3 Feature generation for crop/weed discrimination

The SAD-based relevance measures of the spectral channels for crop/weed classification are given in Table 5.7. Clearly, none of the spectral features provided by the MWLP system showed significant relevance for crop/weed classification. Hence, feature generation using the 3D data provided by the MWLP system was necessary.

Table 5.7: SADs of the MWLP system's spectral feature channels crop/weed classification.

Channel name	Crop plant / Weed plant SAD
LineWidth_Green	0.122
IntenMax_Green	0.243
IntensitySum_Green	0.142
Scatter20Section_Green	0.182
Scatter40Section_Green	0.331
Scatter60Section_Green	0.332
Scatter80Section_Green	0.060
LineWidth_Red	0.127
IntenMax_Red	0.223
IntensitySum_Red	0.149
Scatter20Section_Red	0.104

Channel name	Crop plant / Weed plant SAD
... continued ...	
Scatter40Section_Red	0.504
Scatter60Section_Red	0.100
Scatter80Section_Red	0.072
LineWidth_NIR	0.343
IntenMax_NIR	0.254
IntensitySum_NIR	0.350
Scatter20Section_NIR	0.124
Scatter40Section_NIR	0.192
Scatter60Section_NIR	0.341
Scatter80Section_NIR	0.076

- **Runtime analysis of Point Cloud Library (PCL) surface descriptors**

As shown, for crop/weed discrimination of carrots and weed plants the generation of features from the 3D data of the MWLP system was required. For this task the Point Cloud Library (PCL) offers a manifold of surface descriptors that describe objects surfaces based on the 3D surface point and normals estimated for each point. In particular, the Point Feature Histograms (PFH) and Fast Point Feature Histograms (FPFH) estimators are commonly used surface descriptors developed by Rusu et al. [121] [120] [122] [99]. The PFH features fully interconnect the neighborhood of a point in the 3D point cloud. The transformation from one point's normal to the normal of another neighborhood point is described using three angular features and a distance feature. This is done for each point pair in the neighborhood of a point. Next, in order

to describe the surface point itself the derived values for the point pairs in the neighborhood are aggregated to a histogram serving as descriptor [121] [125]. The FPFH features do not fully interconnect the neighborhood but work in 2 steps. First, the same three angular features as for the PFH are calculated for the point itself and the points in its neighborhood. However, here only the feature values for point pairs including the point itself are calculated. Further, the distance feature is neglected because oftentimes it does not provide much information about the objects as it highly depends on the sensor setup. In particular, for organized clouds the distances of the points in a neighborhood mostly depend on the sensor resolution and, thus, do not provide information about the sensed objects. The angular features obtained for the points and the points in their respective neighborhoods are aggregated to histograms for each point called Simplified Point Feature Histograms (SPFH) [120]. Second, these histograms are combined in the neighborhood of a point by a weighted average depending on the distance the point the SPFH was calculated for [120] [124]. The FPFH estimator is of particular interest as it has already been used for an agricultural application using data of a line profiling sensor. Paulus et al. have applied it for plant organ classification in phenotyping [99]. A different feature descriptor further provided by the PCL is the Radius-based Surface Descriptor (RSD) developed by Marton et al. This descriptor calculates the angles between the normals of the point and adjacent point to describe and its neighborhood. It returns the minimum and maximum local surface radius as describing numeric features. Using these values they can distinguish between different geometric surface classes [83] [82].

For phenotyping applications the data processing can happen offline and only the data collection must be performed online on the field. Therefore, the data processing algorithms for phenotyping are not subject to serious runtime restrictions. However, having in mind automated weeding the processing time plays an important role. The processing must happen online in order to provide the actuator with the information to treat or not to treat a grid cell before it is reached. Of course, during the test described here there was not yet an actuator involved. However, with the application in mind runtime is an issue. In order to assure the possibility of online processing it should at least be possible to process the incoming data in less time than the acquisition took. Fulfilling this constraint, the timing of the processing chain can then be adjusted to be ready before an actuator position is reached by determining the chunk size processed through it. Without fulfillment of this constraint online processing for automated actuation is impossible regardless of configuration, as internal buffers would increase over time.

With these bounding conditions in mind runtime tests of the PCL algorithms for processing MWLP sensor data have been performed. For this, a data set of field-acquired MWLP sensor data from scanning carrots and weed plant at 0.1 m/s with a frame rate of approx. 75 Hz and full camera resolution scanned within 5 seconds was used. The used data set contained 370 MWLP scan lines of each 2048 points each. Prior to the processing tests the data set was stuffed to with interpolated scans and partly padded on the sides to a size of 2074 x 4580 pixels. This stuffing with interpolated scans is always done prior to chunk-wise passing the MWLP data through the image classification pipeline (cf. Sections 5.1.5 and 5.1.6). The image classification pipeline may also include filtering of the data by image processing filter algorithms, such as e.g. erosion, dilation or blur. These algorithms only provide meaningful results if the

Table 5.8: Key features of the PC the performance tests have been conducted on

Component	Type	Remarks
CPU	Intel(R) Xeon(R) E5645	6 cores @ 2.40 GHz
RAM	DDR 3	12 GB @ 1333 MHz Front Side Bus
Graphics card	NVIDIA Quadro 4000	256 Stream Processing Units @ 950 MHz 2048 MB GDDR5 RAM

positioning of the pixels in the image is correct and if the pixel sizes in x- and y-direction are equidistant. Just pushing the MWLP scans line-by-line into images without padding and stuffing with interpolated scans would - depending on the movement - result in improper positioning and scaling of the image and, thereby, rule out the use of image-filters on the MWLP data. Hence, the stuffing with interpolated scans and padding is done prior to passing the chunks to the pipeline. Thus, use of image-filters in the pipeline is safe. For point cloud processing algorithms, however, this would not be required but still does not cause functional problems. The only drawback of processing the data including interpolated pixels is that more points/pixels must be processed to the filter. This causes longer processing times. Therefore, for performance critical processing steps downsampling is done before the data is processed through the filter. E.g. for a downsampling grid of 5-by-5 only every fifth pixel in x-direction and every fifth pixel in y-direction is passed on to the filter, i.e., the number of points/pixels is reduced by 25 with respect to the interpolated data. In principle, this downsampling could also be done immediately after the stuffing with interpolated scans. However, for several image processing such as e.g. erosion, dilation or blur and for the pixel classifiers processing 2074 x 4580 pixels in a much shorter time than 5 sec is possible without problems. Consequently, there is no general need to downsample the interpolated data. Hence, the data is only downsampled for performance critical filters. After the respective filtering operation for these performance-critical filters, their sparse results are interpolated back to the size of the stuffed data to assure same data dimensions.

The performance tests were conducted using a workstation PC. The key features of this PC are listed in Table 5.8. The performance test have been conducted for the different feature estimators at different levels of downsampling of the equidistant stuffed image data. It has been tested for the mentioned feature estimators PFH, FPFH and Radius-based Surface Descriptor (RSD). Further, all of these surface descriptors require the point cloud and normals estimated for each point as input. Therefore, the default normals estimator of PCL and the `pcl:IntegralImageNormalEstimation` [51] [50] were additionally included in the test. The `pcl:IntegralImageNormalEstimation` only works on organized point cloud data. However, the point clouds derived from the MWLP sensor data are organized. Hence, it can be used. For all other tested normals and feature estimators a search method for conducting the neighborhood search is required. It was set to `pcl::search::OrganizedNeighbor`. This search method is also optimized for organized point cloud data and, therefore, faster than `pcl::search::KdTree` or `pcl::search::OctTree`. All estimators were configured to perform a radius search. The search radii were set to values of 10 mm for the normals estimators and 7.5 mm for the feature estimators. These are reasonable values due to the observed plant sizes.

The results of the performance tests with the PCL algorithms are listed in Table 5.9. The grid sizes for downsampling of the stuffed interpolated data are given in the first table row. The second row states the image/organized point cloud dimensions after downsampling. Third row indicates the number of pixels effectively being processed by the respective algorithm at the specific level of downsampling. In the fourth row mentions the (constant) number of samples originally scanned by the MWLP system prior to stuffing/interpolation for proper scaling. The fifth row notes the ratio of samples passed on to the estimators (after to stuffing/interpolation/scaling and downsampling) over scanned samples (prior to stuffing/interpolation/scaling and downsampling) at this level of downsampling.

Following these heading rows in Table 5.9, the results of the performance tests at the different levels of downsampling and for the different feature estimators are given. Each cell states the number of repetitions, the mean runtime in seconds and the standard deviation for the respective case. The processing was performed single threaded in all cases. Of course, all of these algorithms can be speeded up by multi-core parallelization. However, when running in actual application filtering in the image pipeline and other tasks, such as piping, interpolation and classification, must be performed in parallel. Consequently, a single filter cannot use up the entire processing power. Further, the classification pipeline itself implements multi-threading by processing multiple data-chunks in parallel threads through the pipeline. Hence, use of the single threaded version for the individual filters is appropriate.

The mean processing times over the number of processed samples for the different algorithms are further drawn in Figure 5.54. Please mind the logarithmic axes scales. The blue horizontal line represents the data acquisition time of the set (5 seconds, as mentioned). In order of allowing online processing runtimes higher than this are not feasible. The vertical dashed orange line corresponds to a downsampling grid of 10-by-10. At this level of downsampling the number of samples passed on to the estimators is already been reduced by a factor of 0.125 (cf. Table 5.9). I.e., for higher numbers of downsampling less than 10 % of the information gathered is actually used. The use of high resolution sensors such as the MWLP system is then somehow obsolete as only a very minor part of the gathered information is actually been interpreted. Thus, higher numbers of downsampling should be avoided. Hence, the feasible and useful region for online processing of the high resolution data of the MWLP system is the bottom right part of Figure 5.54 separated by the blue horizontal and orange vertical line.

As consequence of the information shown in Figure 5.54, online normal estimation based on the organized sensor data of the MWLP system is not a problem. The `pcl::IntegralImageNormal Estimation` is fast enough to estimate normals even for full-resolution data of the interpolated and scaled data in less time than required for acquisition. The `pcl::NormalEstimation` with organized neighbor search would also be feasible with some downsampling but is still much slower. As no notable differences in the resulting normals of both methods were observed, `pcl::IntegralImageNormalEstimation` was chosen for all normal estimations in the following.

For feature estimation and surface description the PFH estimation takes way to long to be

Table 5.9: Runtime measurement statistics of PCL surface description algorithms on MWLP sensor data with different levels of downsampling.

Downsampling grid	50	15	12	10	8	5	3	1
Image size	41x91	138x305	172x381	207x458	259x572	414x916	691x1526	2074x4580
Processed samples	3731	42090	65532	94806	148148	379224	1054466	9498920
Scanned samples	757760	757760	757760	757760	757760	757760	757760	757760
Ratio	4.92E-3	0.056	0.086	0.125	0.196	0.500	1.392	12.54
Runtime statistics	N tests **Mean runtime [sec]** Standard deviation [sec]							
PCL - IntegralImage NormalEstimation	20 **3.82E-4** 7.14E-6	20 **1.42E-2** 1.81E-4	20 **2.13E-2** 2.79E-4	20 **3.41E-2** 2.91E-4	20 **4.87E-2** 3.13E-3	20 **0.115** 6.11E-4	20 **0.314** 2.01E-2	20 **2.829** 8.03E-2
PCL - FPFHEstimation	20 **1.02E-2** 2.47E-4	20 **0.922** 4.83E-2	20 **2.670** 6.13E-2	20 **4.608** 0.113	15 **10.908** 0.113	4 **71.291** 1.501		
PCL - PFHEstimation	20 **1.12E-2** 1.62E-4	15 **14.345** 6.29E-2	4 **56.083** 3.79E-2					
PCL - RSDEstimation	20 **4.20E-3** 5.13E-4	20 **0.241** 2.13E-3	20 **0.584** 2.87E-2	20 **1.132** 3.83E-2	20 **2.588** 5.60E-2	15 **15.111** 2.46E-2	2 **108.268** 4.55E-2	
PCL - NormalEstimation	20 **2.68E-3** 4.60E-4	20 **0.151** 1.09E-3	20 **0.366** 2.13E-2	20 **0.659** 2.84E-3	20 **1.635** 7.94E-3	20 **9.666** 2.31E-2	4 **70.728** 2.38E-2	

taken into account. The FPFH estimation cuts the feasible region at a very small part. At a downsampling grid of 10-by-10 it takes approx. 4.6 seconds on average, thus going below the 5 seconds of acquisition time. However, there is very few computational reserve. In case the robot moves a bit faster the buffers would immediately start to grow. Therefore, it was also not taken into account in the first place. The RSD estimation, in contrast, showed processing times significantly less than the acquisition time of 5 seconds on this data set. For a downsampling grid of 10-by-10 it was at approx. 1.1 sec, for 8-by-8 it was still at 2.6 seconds. Again, mind the logarithmic scale in Figure 5.54. Despite the apparently small difference the RSD estimation is actually 4-5 times faster than the FPFH estimation. Due to this good performance the PCL RSD estimation developed by Marton et al. [83] [82] was used as starting point for 3D feature generation.

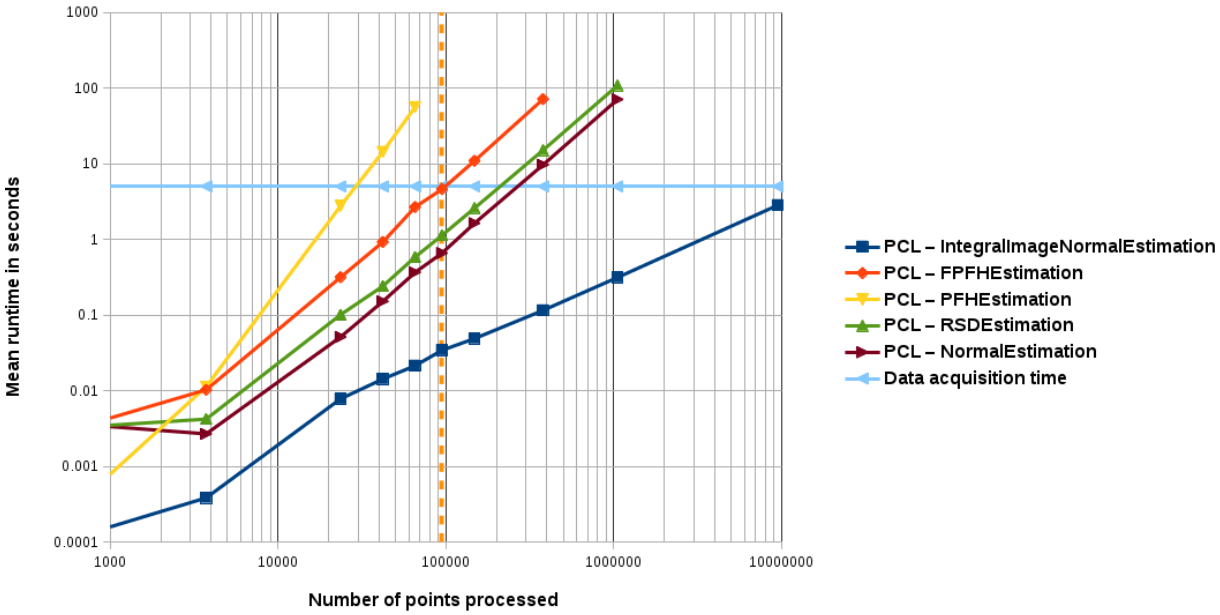


Figure 5.54: Runtime of PCL surface description algorithms on MWLP sensor data with different levels of downsampling.

- **Radius-based Surface Descriptor (RSD)**

The RSD estimation takes point normals as input and calculates the minimum and maximum radius that would be required for surface reconstruction at the given point. For reconstruction purposes, this value has the advantage of being directly connected with a physical meaning [83]. I.e., for surface estimation there is no statistical classification required - in contrast to FPFH features without physical meaning. As Figures 5.55 and 5.56 show, the returned values can directly be mapped to surface classes [82] [83]. Further, the calculation is significantly faster than for FPFH because only a single angular feature is calculated per point pair rather than three.

The procedure for the calculation of the minimum and maximum radius for an individual surface point is as follows [83]:

1. Calculate the angle α (cf. left-hand side of Figure 5.55) between the normals of a point pair for each point pair including the point itself and any of its neighbors within the maximum search distance d_{search}.

2. Group the obtained angles values depending on the distance of the point pair into distance bins. By default, five bins are used. For each bin determine the minimum and maximum angular value.

3. The relation between the angle α, the distance d and the radius r in the left-hand side of Figure 5.55 is given by $d = \sqrt{2r}\sqrt{1 - cos(\alpha)}$ [83]. Using Taylor series decomposition Marton et al. showed that for the relevant angle range this can be approximated by

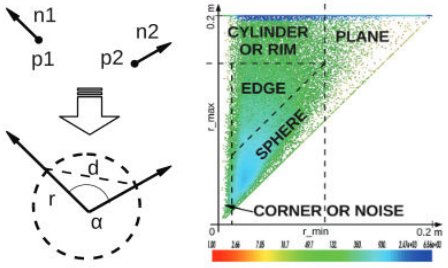


Figure 5.55: RSD feature meaning and mapping. Source: [82].

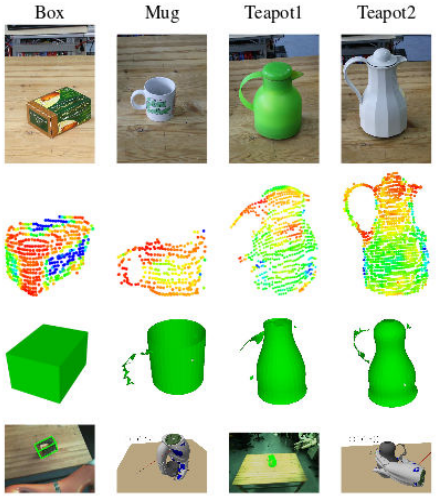


Figure 5.56: Object surface classification based on RSD. Source: [83].

$d \approx r\alpha$. Hence, the minimum and maximum radius features can be approximated from the values in the distance bins by linear regression [83]. The estimation of the final feature value based on linear regression is denoted in Formula 5.13.

Formula 5.13: Linear regression of radii from min/max angles of the distance bins.

$$r_{min} = \frac{\sum \alpha_{min_i} d_i}{\sum \alpha_{min_i}^2}$$

$$r_{max} = \frac{\sum \alpha_{max_i} d_i}{\sum \alpha_{max_i}^2}$$

, *where* α_{min_i} *and* α_{max_i} *are min/max angles of the distance bin*

, *and* d_i *is the center distance of the distance bin*

- **ExtendedRSD**

Before being used for plant classification of MWLP data the concept of the descriptive RSD was partly simplified and partly extended. Its calculation was simplified. As Formula 5.13 states, the finally resulting feature radii depend on the respective angles and on the distances of the distance bins. However, the distances of the bins are pre-configured values depending on the search radius and the number of bins. Hence, the descriptiveness of the features can only derive from the angles. The distance value inducts normalization as the angle between the normals of e.g. a sphere with constant radius does depend on the distance of the point pair. However, if calculated as described in Formula 5.13, the angles of the distance bin with the highest value contribute the most to the final result as their values are multiplied with the highest distance. This leads to the thought just to calculate the angular features for the outer distance bin.

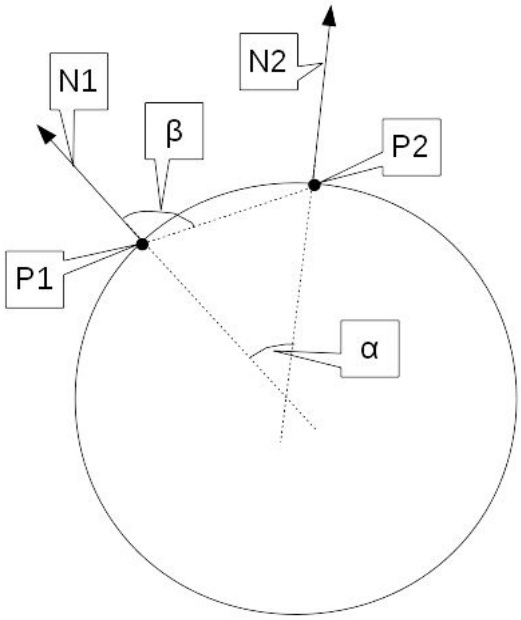


Figure 5.57: Angular features for extended RSD.

Further, if doing so, the angular values can directly be returned as applying a constant factor does not add new information.

Following this thought, the extended RSD implemented as part of this work introduces a minimum distance. Only point pairs with distances between the point to describe and the pairing point ranging between this minimum distance and the maximum search distance are evaluated. Further, the angular features are returned directly. This is similar to using only the (most influential) outer bin of the RSD estimation. Moreover, the other (less relevant) angles are not calculated at all. To avoid crude outliers the extended RSD thereby do not return the minimum and maximum angle values but the 5 % and 95 % percentile values of the samples monitored for a point. Just for testing purposes mean and variances of the angles calculated around a point are additionally returned.

Further, the angular features are extended by another angle. For a pair of two points P1 and P2 with their respective normals N1 and N2 not only the angle α between their normals is calculated but also the angle β spanned by the normal N1 and the vector from P1 to P2. This is depicted in Figure 5.57. The angles observed for this feature within the mentioned distance window around the point are again described by the 5 % and 95 % percentile values as well as mean and variance. These features can be interpreted as measure of concavity/convexity.

A class inheriting from `PipedGenericOverlayedImageFilterBase` (cf. Section 5.1.6) implements the calculation of the extended RSD features. Hence, it can be plugged into the image classification pipeline as foreground preprocessor and can use context of prior and following data

chunks while processing the data. The `pcl::IntegralImageNormalsEstimation` is included in this filter before the feature estimation is conducted as normals for the points to describe are required. The feature channels added to the MWLP data by this filter are listed in Table 5.10. The feature estimator was implemented on CPU and GPU.

Table 5.10: Numerical features appended to the MWLP data by the extended RSD estimation.

Feature	**Description**
MwlpCloudFeature_ Curvature	Curvature observed during normal estimation. Passed from the `pcl::IntegralImageNormalsEstimation`.
MwlpCloudFeature_ AngleNormalsMax	Upper percentile value (e.g. 95 %) of the monitored angles α between the normals. If the percentile is configured 100 % this value essentially contains the same information like the outer distance bin angle value (i.e., most influential angle value) of the RSD estimation.
MwlpCloudFeature_ AngleNormalsMin	Lower percentile value (e.g. 5 %) of the monitored angles α between the normals. If the percentile is configured 0 % this value essentially contains the same information like the outer distance bin min angle value (i.e., most influential angle value) of the RSD estimation.
MwlpCloudFeature_ AngleNormalsMean	Mean of the monitored angles α between the normals.
MwlpCloudFeature_ AngleNormalsVariance	Variance of the monitored angles α between the normals.
MwlpCloudFeature_ AngleNeighborPointMax	Upper percentile value (e.g. 95 %) of the monitored angles β between the normal of the point to describe and the connecting vectors to it pairing points.
MwlpCloudFeature_ AngleNeighborPointMin	Lower percentile value (e.g. 5 %) of the monitored angles β between the normal of the point to describe and the connecting vectors to it pairing points.
MwlpCloudFeature_ AngleNeighborPointMean	Mean of the monitored angles β between the normal of the point to describe and the connecting vectors to it pairing points.
MwlpCloudFeature_ AngleNeighborPointVariance	Variance of the monitored angles β between the normal of the point to describe and the connecting vectors to it pairing points.
MwlpCloudFeature_ AngleNeighborCount0	Count of valid points inside the circle-like 2-D search window around the monitored pixel (z-coordinate is not taken into account).
MwlpCloudFeature_ AngleNeighborCount1	Count of valid point found inside the search radius in 3D (z-coordinate is taken into account).
MwlpCloudFeature_ AngleNeighborCount2	Count of valid points that are closer than the minimum distance, i.e., their angles are not taken into account for feature estimation as they are to close.
MwlpCloudFeature_ AngleNeighborCount3	Count of valid points that are farer than the minimum distance but inside the search radius. I.e., the angles α and β for the pairing of these points with the monitored point are actually taken into account for feature estimation.

- **Line flicker**

The line flicker feature estimator was implemented due to an observation made when inspecting the distance map for carrots and weeds shown in Figure 4.37. The fine pinnate leafs of the carrots induce a kind of flicker in the distance image with many abrupt changes of the distance values. In contrast, oval leafs of weed plants have smoother leaf surfaces or grass-like weed plants only induce a single abrupt distance change. Therefore, the line flicker feature estimator was implemented. It analyses a kernel around the point to describe in the distance image. Thereby, not all pixels of the kernel are taken into account. It only runs along 4 axes in the kernel patch: x-direction, y-direction, 45° descending diagonal, 45° ascending diagonal. While running along these axes it sums up the absolute changes in the distance values (z-direction) between the current value and the previous value.

All features appended by the line flicker estimation are described in Table 5.11. Same as for the extended RSD estimation the line flicker estimation is also implemented on both CPU and GPU.

Table 5.11: Numerical features appended to the MWLP data by the line flicker feature estimation.

Feature	**Description**
FlickerFeature_MagicToValid ChangeCountFor_DistMap	Number of changes along the line axes between adjacent pixels with valid values and 'magic', i.e., non-measureable invalid, values.
FlickerFeature_ValueRange For_DistMap	Difference between the minimum and maximum observed distance values.
FlickerFeature_ValueCount For_DistMap	Number of valid, i.e., not 'magic', values on the analyzed line axes.
FlickerFeature_DirectionChange CountFor_DistMap	Number of sign changes of the distance values between adjacent pixels while running along the line axes.
FlickerFeature_ChangeSum For_DistMap	Summed up absolute changes of distance values of adjacent pixels while running the line axes.
FlickerFeature_NormalizedChange SumFor_DistMap	Change sum divided by direction change count.

- **Runtime analysis for here implemented feature estimators**

The processing time statistics for the here implemented feature estimators are given in Table 5.12. The data set, computer and boundary conditions are the same as for the tests conducted with the PCL estimators (cf. Table 5.9). The mean runtimes for the here implemented feature estimators are depicted in Figure 5.58. Further, Figure 5.58 draws the performances of the PCL FPFH estimation and of the PCL RSD estimation, for orientation. The horizontal line plotting the data acquisition time and the vertical line indicating the downsampling grid of 10-by-10 are added to Figure 5.58, same as done for Figure 5.54. Again, the region right of the vertical line and below the horizontal line is the part useful and feasible for online processing.

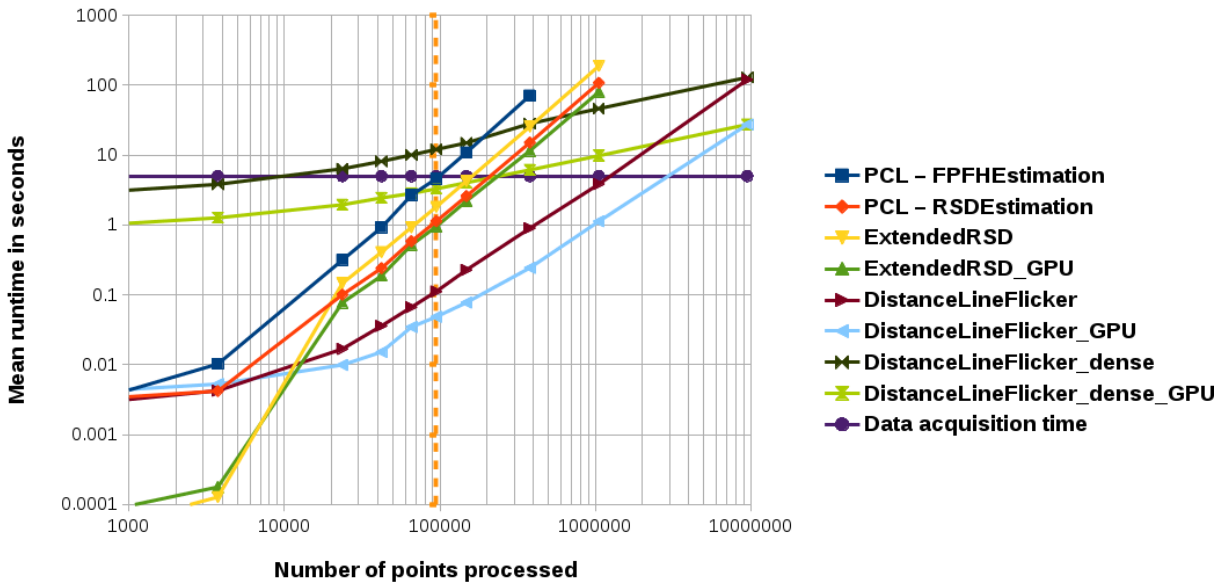


Figure 5.58: Mean runtimes of the here implemented feature estimators.

The extended RSD features are calculated on both CPU and GPU. The introduced minimum point distance for the test was set to 3.5 mm. The search radius was set to 7.5 mm, same as for the feature estimators tested before. The CPU version of the extended RSD is a bit slower than the RSD estimation for the PCL also running on CPU (1.82 sec vs. 1.13 sec @ downsampling grid of 10-by-10). This can be explained by the different changes influencing the performance. On one hand, the estimation is speeded up by skipping the calculation of angles for point pairs with small distances. On the other hand, it is slowed down because two angle values (α and β in Figure 5.57) are calculated per point pair rather than just one. Further, the angle set around one point is not only described by minimum and maximum value but by the upper and lower percentile as well as mean and variance. This implies that some more calculations have to be conducted while the increase in processing time is moderate. Still, the extended RSD features still significantly outperform the FPFH estimation in terms of processing time (1.82 sec vs. 4.60 sec @ downsampling grid of 10-by-10). The GPU-based version of the extended RSD further allows speeding up the processing time by a factor of approx. 2, thereby even outperforming the RSD estimation for the PCL.

The distance line flicker features were tested on CPU and GPU in two different modes. The default mode (labeled 'DistanceLineFlicker' and 'DistanceLineFlicker_GPU' in Table 5.12 and Figure 5.58) corresponds to the evaluations conducted before with the other feature estimators. The point cloud was downsampled completely and then passed on to the feature estimator. Hence, the estimator does only 'see' the downsampled data. Evidently, the estimation of the distance line flicker features is considerably faster than all other feature estimators tested. Even the CPU-based implementation is approx. 10 times faster than PCL RSD estimation. The GPU-based implementation induces another speed up by a factor of approx. 4. This is because

Table 5.12: Runtime measurement statistics of here implemented surface description algorithms on MWLP sensor data with different levels of downsampling for CPU and GPU.

Downsampling grid	50	15	12	10	8	5	3	1
Processed samples	3731	42090	65532	94806	148148	379224	1054466	9498920
Runtime statistics	N tests **Mean runtime [sec]** Standard deviation [sec]							
ExtendedRSD	20 **1.28E-4** 4.32E-6	20 **0.409** 2.65E-2	20 **0.925** 3.73E-2	20 **1.820** 6.62E-3	20 **4.253** 7.12E-2	15 **25.635** 9.70E-2	2 **189.087** 0.144	
ExtendedRSD_ GPU	20 **1.78E-4** 4.89E-6	20 **0.187** 1.20E-3	20 **0.518** 1.85E-4	20 **0.950** 2.01E-4	20 **2.204** 3.83E-4	15 **11.564** 7.26E-4	4 **80.202** 0.351	
DistanceLine Flicker	20 **4.26E-3** 1.35E-4	20 **3.59E-2** 2.57E-4	20 **6.65E-2** 3.87E-4	20 **0.112** 3.65E-4	20 **0.227** 7.40E-4	20 **0.901** 1.81E-2	20 **3.847** 8.01E-2	2 **120.320** 0.326
DistanceLine Flicker_ GPU	20 **5.28E-3** 1.57E-4	20 **1.52E-2** 6.69E-4	20 **3.46E-2** 2.52E-4	20 **4.85E-2** 3.78E-4	20 **7.75E-2** 9.42E-4	20 **0.244** 9.48E-4	20 **1.129** 1.69E-3	15 **27.401** 9.19E-3
DistanceLine Flicker_dense	20 **3.818** 7.94E-2	20 **8.142** 0.102	15 **10.048** 7.98E-2	15 **12.079** 5.98E-2	15 **14.954** 4.62E-2	15 **28.239** 0.432	6 **46.261** 0.175	2 **129.432** 0.175
DistanceLine Flicker_dense_ GPU	20 **1.267** 4.24E-2	20 **2.440** 4.31E-2	20 **2.831** 3.85E-3	20 **3.297** 3.78E-3	20 **4.036** 3.01E-3	20 **6.144** 3.56E-3	20 **9.811** 3.33E-3	15 **27.395** 1.31E-2

the line flicker feature estimation does not fully analyze all points/pixels of the kernel around the point/pixel to describe but only uses the pixels that are part of the lines it analyses. Namely, only points situated on the horizontal line centered in the kernel, the vertical line in centered the kernel or one of the diagonals are taken into account. Consequently, the complexity of this estimator is linear with respect to the kernel size. For the other estimators the processing time grows with the 2nd or 3rd polynomial of the kernel size. Hence, the calculation is significantly faster for high resolution data and reasonable kernel dimensions.

The analyzed performance statistics series in Table 5.12 and Figure 5.58 labeled 'DistanceLineFlicker_dense' and 'DistanceLineFlicker_dense_GPU' represent a different downsampling mode for the distance line flicker estimation. This was reasonably only possible, because the estimator is faster than the others. Here, the data is passed on to the feature estimator without downsampling. However, the estimator does only calculate the features for the samples that are part of the downsampling grid. This means that the number of resulting point samples does not increase, but these are estimated using a more qualified data base. If the downsampling grid is

1-by-1, i.e., no downsampling is applied both, modes are essentially the same. Consequently, at this point the runtimes do practically not differ (cf. Figure 5.58). However, if downsampling is applied the runtimes of the 'dense' modes are much higher as much more points are analyzed. But still in this mode on GPU the distance line flicker estimation is possible within 3.30 seconds for a downsampling grid of 10-by-10 which is still feasible for online processing of the data acquired in 5 seconds.

- **Relevances of the generated features for crop /weed classification**

Based on the runtime analyses of the different feature generators the decision was taken to use the extended RSD and distance line flicker feature estimators both running on GPU. The downsampling grid for both estimators was set to 10-by-10, but the distance line flicker estimation was running in 'dense' mode. In this mode the computing times of both estimators were found as 0.95 seconds and 3.30 seconds for the above mentioned data set acquired in 5 sec. Even assuming both blocked the entire GPU resources over that times, there would still be some reserve. Further, this allows off-boarding the feature estimation to GPU. I.e., the interpolation, piping, classification and additional normal estimation using `pcl::IntegralImageNormalEstimation` as input for extended RSD etc. happening on the CPU in parallel are not effected as much as they would be if the feature estimation were also conducted on CPU.

The SAD-based relevance measures of the numeric features generated by the two estimators for pixel-based classification of crop plants and weed plants are listed in Table 5.13. For feature descriptions please note Tables 5.11 and 5.10. Further, the NDVI is given as it is calculated anyway for soil/plant classification. However, same as the other spectral features listed in Table 5.7 the NDVI also has no relevance here. This again approves the necessity of feature generation from the 3D data.

As Table 5.13 shows, there are a couple of 3D-features having moderately high relevances for crop/weed classification, clearly outperforming the spectral features listed in Table 5.7. Very high relevances in the range of 0.9 and higher like observed for potato/residual classification or soil/plant separation were not found here, though. However, by combining a set of features with moderately high relevances and low redundancies a classification with a reasonable level of accuracy might still be possible.
Of the features generated by the distance line flicker estimation (labeled *FlickerFeature_ * For_DistMap*) the features *MagicToValidChangeCount*, *ValueRange*, *ValueCount*, *ChangeSum* and *NormalizedChangeSum* show relevances for the desired classification. Of these, *MagicToValidChangeCount*, *ValueCount* and *ChangeSum* were selected to be passed on to the classifier. The *ValueRange* and *NormalizedChangeSum* have some informational connections to the *ChangeSum*. Therefore, of these three only the latter was picked for classification as it showed the highest relevancy while the other were skipped to avoid redundancies.

The curvature calculated during normal estimation by `pcl::IntegralImageNormalEstimation` has a relatively low relevance measured here as 0.449 (cf. Table 5.13, labeled *MwlpCloudFeature_Curvature*). Consequently, it was not passed on to the classifier.

Analyzing the relevancies of the angular features provided by the extended RSD estimation it can be stated that for both angle sets described - the angles between the normals of the point pairs (i.e., angle α in Figure 5.57) and the angles between the normal and the connecting vector of the point pair (i.e., angle β in Figure 5.57) - the minimum and maximum values are more relevant descriptors than the mean and standard deviation. Hence, the decision of Marton et al. for RSD to use the min/max values rather than mean [83] can be proven correct also for the data set analyzed here. However, for this data set the min/max values of the neighborpoint angle β have higher relevances than those of the normals angle α. Particularly, the *MwlpCloudFeature_AngleNeighborPointMin* shows a relevancy of 0.602. It was therefore used for classification and passed on to the classifier. Its complementing value *MwlpCloudFeature_AngleNeighborPointMax* was also passed on to the classified. Further, the counters of valid neighboring pixels show some relevancies. However, only the *MwlpCloudFeature_ValidNeighborCount1* feature is used for classification as it is the most relevant counter and the other counters provide somehow similar information.

For comparison and validation purposes, the SAD-based feature relevance measures were further calculated for the PCL implementations of the RSD and FPFH estimation neglecting that these are slower in processing than the here implemented feature estimators. The principle radii features obtained by running the RSD estimation showed relevances of 0.432 and 0.424 for the minimum and maximum radius, respectively. These relevance values are similar to those of the minimum and maximum normals angles mentioned in Table 5.13, thus showing that the major part of the descriptiveness of the RSD features derives from the angle values and the outer bin is the most influential. The 33 elements of histograms descriptors obtained by running the FPFH estimation of PCL on the tested data set on average showed low relevances with a mean of 0.357. The standard deviation of the 33 relevances from the mean was 0.114. Only two of the 33 the numeric entries of the FPFH histograms showed relevances higher than 0.6. These were entry number 27 with a relevancy of 0.602 and entry number 28 with 0.699. This again proves an observation made with the angle features for normals and angle points mentioned in Table 5.13: The angle between normal and the connection vector neighbor point, i.e., angle β in Figure 5.57, is angular feature most sensitive for crop/weed classification. The FPFH estimation returns a joint histogram for the three angular features it monitors. Each angular feature histogram occupies a chunk of 11 of the total 33 entries. The entries 23 to 33 thereby represent the third angular feature. In turn, this feature is the angle between normal and the connection vector neighbor point [31] [119], i.e., angle β in Figure 5.57. However, because only 2 of the 33 histogram entries obtained by FPFH estimation show moderate (not high) relevances and the runtime is much longer than that of the extended RSD estimation, the decision to exclude it was fixed.

Table 5.13: SADs of the generated feature channels crop/weed classification.

Channel name	Crop plant / Weed plant SAD
NormalizedDifference VegetationIndex	0.317
FlickerFeature_ MagicToValid ChangeCountFor_ DistMap	0.753
FlickerFeature_ ValueRangeFor_ DistMap	0.631
FlickerFeature_ ValueCountFor_ DistMap	0.681
FlickerFeature_ DirectionChange CountFor_ DistMap	0.240
FlickerFeature_ ChangeSumFor_ DistMap	0.711
FlickerFeature_ NormalizedChange SumFor_DistMap	0.617
MwlpCloudFeature_ Curvature	0.449
MwlpCloudFeature_ AngleNormalsMax	0.456
MwlpCloudFeature_ AngleNormalsMin	0.436
MwlpCloudFeature_ AngleNormalsMean	0.218

Channel name	Crop plant / Weed plant SAD
... continued ...	
MwlpCloudFeature_ AngleNormalsVariance	0.388
MwlpCloudFeature_ AngleNeighborPointMax	0.486
MwlpCloudFeature_ AngleNeighborPointMin	0.602
MwlpCloudFeature_ AngleNeighborPoint Mean	0.300
MwlpCloudFeature_ AngleNeighborPoint Variance	0.258
MwlpCloudFeature_ ValidNeighborCount0	0.598
MwlpCloudFeature_ ValidNeighborCount1	0.599
MwlpCloudFeature_ ValidNeighborCount2	0.489
MwlpCloudFeature_ ValidNeighborCount3	0.469

- **Summary of classifier configuration for plant classification experiments**

Summarizing the observations made during the tests described in this Section 5.3.3 and the prior Section 5.3.2 the configuration for the plant classification was fixed. Table 5.14 shows the configuration of the plant classification for the experiments described in Sections 5.3.4 and 5.3.5. Note that the ROS node running the plant classification also uses (initializes and finalizes) the CUDA memory pool mentioned in Section 3.2.4 - same as the `/line_detection_node` of the MWLP system. Again, this allows distributed and memory management also for the estimators of the extended RSD and the line flicker features without causing implicit synchronization of the CUDA streams. The used lens, the camera sampling mode and frame rate as well as the driving speed of the carrying BoniRob are not mentioned in Table 5.14. These were varied during / between the experiments and therefore will be stated along with the respective results in Sections 5.3.4 and 5.3.5.

Table 5.14: Configuration of MWLP system and classification pipeline for plant classification.

MWLP system	
Lasers	• Green (#4 @532 nm) • Red (#6 @650 nm) • NIR (#8 @850 nm)
Image classification pipeline	
Common preprocessor	• none
Background preprocessor	• Filter for calculation of NDVI
Foreground preprocessor	Stacked filter containing • Normal estimation using `pcl::IntegralImageNormalsEstimation` • Extended RSD estimation on GPU • Distance line flicker estimation on GPU
Segment filter	• none
Features binary classifier	• *IntenMax_Green* • *Scatter60Sum_NIR* • *NormalizedDifferenceVegetationIndex*
Features object group classifier	• *FlickerFeature_MagicToValidChangeCountFor_ DistMap* • *FlickerFeature_ValueCountFor_ DistMap* • *FlickerFeature_ChangeSumFor_ DistMap* • *MwlpCloudFeature_AngleNeighborPointMin* • *MwlpCloudFeature_AngleNeighborPointMax* • *MwlpCloudFeature_ValidNeighborCount1*

5.3.4 Results from classifying field data of carrots

The so-configured classification pipeline was feed with the field-acquired sensor data collected during the field tests with the MWLP system (cf. Section 4.5). Sample results of this processing are shown in Figures 5.59 to 5.62.

Figure 5.59 shows a color representation of a subset of the collected MWLP sensor data scanned with the RG_IR lasers. The reflection of the NIR laser is again visualized in blue, i.e., same color configuration like in the left side of Figure 4.37 in Section 4.5. Figure 5.60 shows the binary image generated by the binary soil/plant classifier. White pixels indicate plants, black pixel the remaining background, i.e., soil. As the figure shows, the image does not provide a perfect plant / soil segmentation. As mentioned, this was never intended, though. However, it shows a good filter image for filtering out the vast majority of irrelevant soil pixels before plant classification without filtering to many plant pixels.

Figure 5.61 shows the probability image as result of the object group classifier including its pre-processors for generation of features from the 3D data. In this case the colorization of the image is done - analog to Formula 5.11 for potato classification - according to the following Formula 5.14.

Formula 5.14: Colorization of the probability image.

$$ColorPixel_k = \sum_{i=1}^{N} P(ObjectGroup_i|k) * Color_{ObjectGroup_i}$$

$$, where\ N\ is\ the\ number\ of\ ObjectGroups\ to\ classify\ into\ and$$

$$Color_{crop\_plant} = RGB(0, 255, 0)$$

$$Color_{weed\_plant} = RGB(255, 0, 0)$$

Obviously, the pixel-based plant classification of the crop/weed plants illustrated in Figure 5.61 works good for the weed plants. For the crop plant it also works relatively good in most cases. However, some plants show misclassified pixels, particularly at the leaf boundaries. To evaluate how this would influence a weeding process the treatment grid was further generated, as seen before for potato classification. The default configuration of the treatment grid with included protection of certainly classified crop plant cells was used for this. The parameters used in this case for the grid aggregation are given in Figure 5.25 in Section 5.1.7. The resulting treatment grid is shown as overlay over the original image in Figure 5.62. Inspection of this image shows that most of the grid cells containing weed plants are treated while most of the grid cells containing crop plants (carrots) stay untreated.

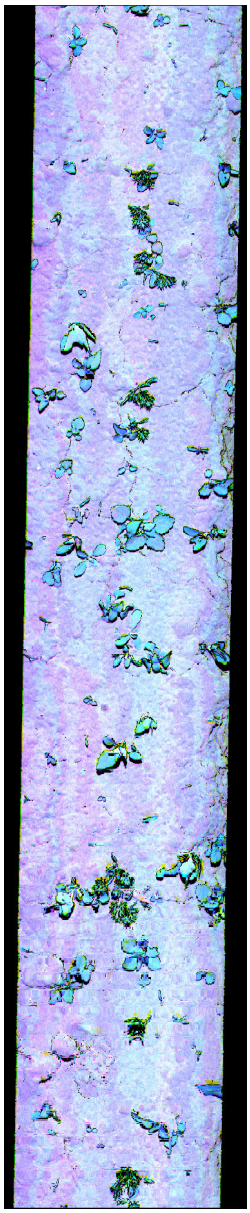


Figure 5.59: Color visualization of the original MWLP sensor data as input for the plant classification.

Figure 5.60: Binary image generated by the soil / plant classifier for filtering soil pixels.

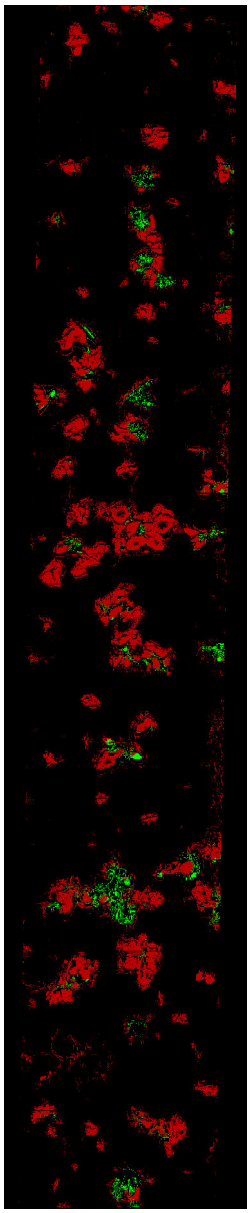


Figure 5.61: Probability image of the pixel-based crop / weed classification.

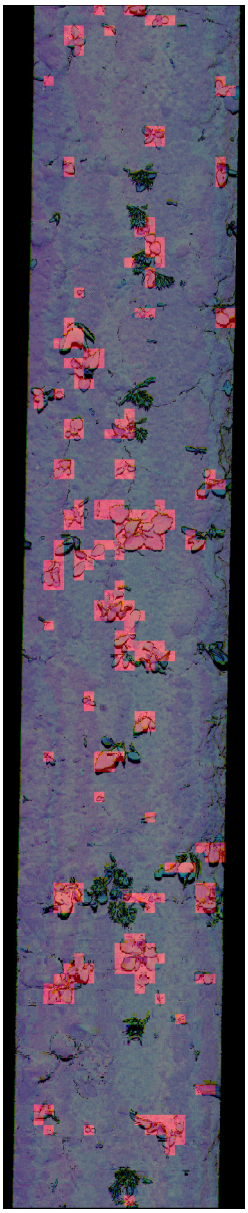


Figure 5.62: Overlay of the original input image and the grid with the final treatment decision. Red shaded grid cells are to be treated.

- **Validation procedure**

In order to validate the treatment grid generated based on the plant classification a reference grid was created by manually assigning references for each grid cell by a human user. The user referenced all grid cells while inspecting the color visualization of the MWLP system as cells of crop plants, cells of weed plants or background cells (i.e., soil). During the manual referencing user did not see the auto-generated grid nor any other result created by the classification pipeline but only the color image representation of the MWLP sensor data. Hence, he/she had to reference the grid from scratch without being influenced by the automatic result. The manually created reference grid is shown in Figure 5.65 as overlay of the original image visualization next to Figures 5.63 and 5.64 just repeating the original data and the automatically generated grid for orientation.

The manually referenced grid was then compared with the automatically generated treatment grid. The comparison of both grids as overlay over the original data is given by Figure 5.66. For the comparison there are six cases that can occur:

- Referenced as background, automatically not treated
  –> dark shade in Figure 5.66
- Referenced as background, automatically treated
  –> red shade in Figure 5.66
- Referenced as crop plant, automatically not treated
  –> green shade in Figure 5.66
- Referenced as crop plant, automatically treated
  –> yellow shade in Figure 5.66
- Referenced as weed plant, automatically not treated
  –> blue shade in Figure 5.66
- Referenced as weed plant, automatically treated
  –> purple shade in Figure 5.66

Hence, dark, green and purple are the preferred colored of the overlaying shade in Figure 5.66 as these indicate the correct treatment according to the references of the cells.

For the major part of the grid cells in Figure 5.66 the automatic result and the manual reference match. However, inspecting Figure 5.66 it can be seen that misclassification does occur. There are three different kinds of misclassification with different potential impact in a practical application. The following kinds or misclassification can occur:

- Referenced as background (soil)), automatically treated
  Here the actuator tool would hit the ground at a point where there is neither a crop plant nor a weed plant, i.e., blank soil. This kind of misclassification is uncritical as it does not affect the yield. It might cause additional wear of the actuator tool or have other minor effects, but these are tolerable.

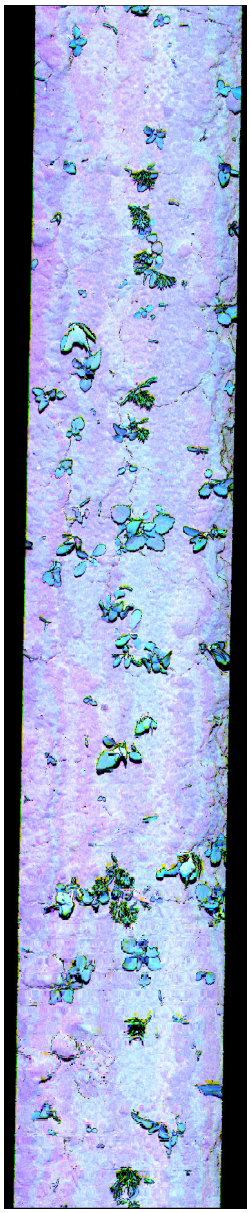


Figure 5.63: Color visualization of the original MWLP sensor data as input for the plant classification.

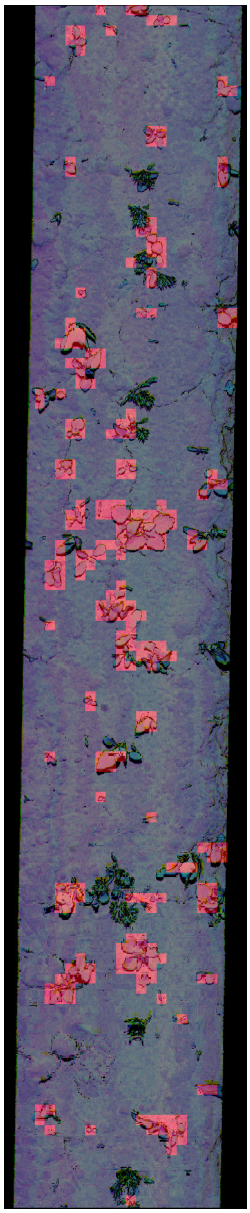


Figure 5.64: Overlay of the original input image and the grid with the final treatment decision. Red shaded grid cells are to be treated.

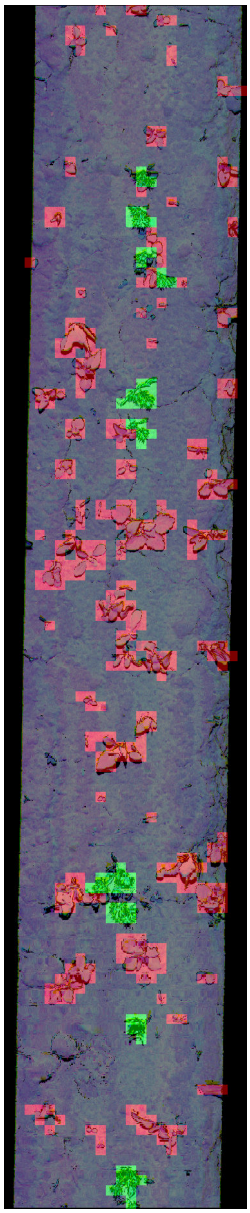


Figure 5.65: Manually created reference grid.

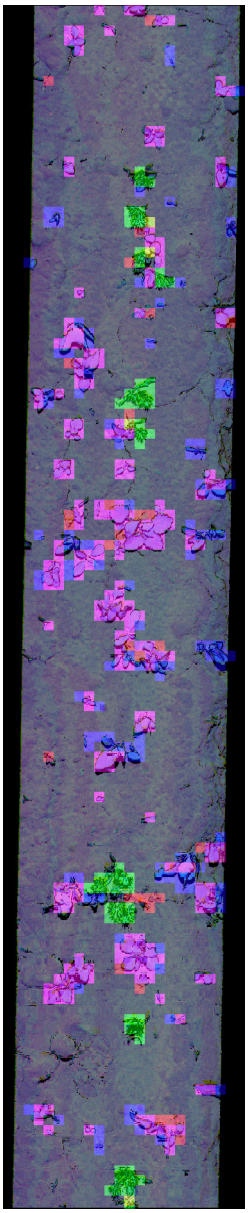


Figure 5.66: Comparison of the manually created reference grid with the automatically generated treatment grid.

Table 5.15: Result statistics of the grid comparison shown in Figure 5.66.

	Manual background	Manual good	Manual bad	N grid cells
Auto treated	1.25 %	**6.78 %**	65.67 %	262
Auto not treated	98.75 %	93.22 %	23.41 %* / **10.98 %****	2618
N grid cells	2475 = 100 %	59 = 100 %	346 = 100 %	2880

- Referenced as good (crop plant), automatically treated
  Here the actuator tool would hit the ground at a position covered by a crop plant. This is **critical**. The crop plant must not be damaged, no matter where and how. Hence, this is only tolerable for a low percentage of crop plants.

- Referenced as bad (weed plant), automatically not treated
  Here the actuator would not treat a grid cell assigned as belonging to a weed plant. In principle, this should be avoided. However, inspecting Figures 5.64 to 5.66 it is notable that the user labeled the plants mostly including boundary while the automatic grid generation - particularly for large weed plants - often skips the boundary. If such a weed plant is treated with a stamp-like actuator all over the center and the boundaries are skipped, it will still be severely damaged, thus allowing the crop plant to advance and overtake the weed plant. Therefore, for this kind of misclassification a case distinction is done. If an untreated grid cell referenced as belonging to a weed plant has an adjacent cell, which is actually treated, it will be assumed that the plant is damaged there, thus not absolutely requiring this cell to be treated. Hence, in this case, the misclassification can be assumed uncritical. If an untreated grid cell referenced as belonging to a weed plant has no adjacent treated cell, it will be assumed that the entire plant is left untreated and, thus, the misclassification is **critical**.

Following these assumptions, the statistics comparing the auto generated grid with the manually referenced grid are summed up. Table 5.15 shows an example of the used pattern with the result for the comparison shown in Figure 5.66. In the table cell 'Manual bad' / 'Auto not treated' the described case distinction is done. The value marked (*) indicates the percentage of cells there with adjacent treated cells, i.e., the uncritical part. The value marked (**) indicates the percentage of cells there without adjacent treated cells, i.e., the critical part. The misclassification identified as **critical** are marked bold in Table 5.15. These critical values should ideally be zero.

Table 5.15 only shows sample result statistics to illustrate the comparison of automatically generated grid cells with the manually created reference grid at the example of the data shown in Figures 5.59 to 5.66. In this case, 6.78 % of the crop plant cells are illegally damaged and 10.98 % of the weed plant cells stay unfortunately untreated.

The grid cell size was - as mentioned - configured to be approx. 1 cm by 1 cm for this and the following evaluations. In the example data show in Figures 5.59 to 5.66 there were 24 lateral grid cells. Hence, the 2880 grid cells mentioned in Table 5.15 represent a section of approx. 1.2 m length of the respective ridge top.

Table 5.16: Misclassification excerpt of Table 5.15.

Data set	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
Example set	1.25 % of 2475	**6.78 %** of 59	23.41 % / **10.98 %** of 346

Table 5.15 just shows the results for a sample set of data. However, its pattern is used through the following parts for describing the performance of the automatic classification within different field situations and with different system configurations. To show it a bit more compact, only the misclassification percentages and the number of grid cells they relate to are given in the tables throughout the following validation parts. Table 5.16 serves as an example of these following tables. It only lists the misclassification figures of all the numbers stated in Table 5.15. The remaining numbers given in Table 5.15 but not mentioned in Table 5.16 can be calculated trivially given the content of Table 5.16. Clearly, all percentage values in Table 5.16 would ideally be zero. However, differences from zero are uncritical for the not bold numbers, while they are **critical** for the bold figures.

- **Validation statistics for different field situations and configurations**

The first field tests with the MWLP system mounted into BoniRob for scanning carrots and weed plants were conducted on 2014/06/15. Due to some issues with the camera driver at that day the camera was operating at only approx. 50 Hz rather than its maximum frame rate of approx. 100 Hz for full frame images. Further, despite that the Schneider lens outperformed the Pentax lens in several tests in Sections 4.2 and 4.3 the Schneider lens was not available at that day. Hence, the Pentax lens had to be used.

The measurements were conducted on 2014/06/15 at different driving speeds of BoniRob of 25 mm/s, 50 mm/s and 100 mm/s. The result statistics for the processing results of the different data sets are shown in Table 5.17. Thereby, data subsets, which had been touched for labeling, algorithm development and parameter tweaking, were not included in the summarizing statistics to avoid to optimism.

As Table 5.17 shows, the critical error rates are at quite good levels of less than 10 % throughout

Table 5.17: Result statistics for data collected on 2014/06/15.

Driving speed	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
25 mm/s	4.15 % of 8775	**6.00 %** of 483	31.35 % / **9.62 %** of 1694
50 mm/s	2.09 % of 3534	**3.70 %** of 162	32.28 % / **6.60 %** of 697
100 mm/s	4.27 % of 7500	**8.93 %** of 224	34.22 % / **6.01 %** of 895

Table 5.18: Result statistics for data collected on 2015/06/10 with full camera resolution.

Driving speed	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
25 mm/s	2.40 % of 6649	**12.44 %** of 217	29.51 % / **7.10 %** of 183
50 mm/s	4.65 % of 26292	**10.08 %** of 833	33.52 % / **17.10 %** of 883
100 mm/s	1.06 % of 29252	**11.73 %** of 733	27.08 % / **10.97 %** of 720

the tests conducted on the data of 2014/06/15. Significant dependencies on the driven speed cannot be recognized in the tested speed range.

The classification algorithm was developed in winter 2014/15 and spring 2015 based on subsets of the data collected on 2014/06/15. As it showed good results for the other data collected on that day it was further tested on carrot scans of the next year 2015.

The carrot scan data of the next season was collected on 2015/06/10. It was also gathered using BoniRob for carrying the MWLP system over the carrot ridges and providing odometry data. Unfortunately, while conducting these field tests there were different issues with the camera driver causing the camera frame rate for full resolution to be limited at approx. 75 Hz. However, in meantime the subsampling mode of the camera had been implemented and was also tested at an increased frame rate of approx. 360 Hz. Again, for these tests only the Pentax lens was available and was used.

For the classification tests, a small part of the data was accessed for labeling, same as it would be done by a user following the In-Field-Labeling concept (cf. Section 5.1.1). Further, another small set of data was accessed for tweaking the parameters of the treatment grid aggregation (cf. Section 5.1.7). These are the modification options the classification pipeline offers to the user in order to adapt the system for a specific field situation. Hence, these are modifications that do not break with the concept as any user could perform them. Further modification in the parameterization or even programming were not done for adapting the system for the new field situation. Moreover, the data subsets accessed for labeling or adjustment of the grid aggregation parameters were excluded from the following statistics.

At full camera resolution and 75 Hz camera frame rate the scanning was conducted on 2015/06/10 at driving speeds of 25 mm/s, 50 mm/s and 100 mm/s. The resulting statistics of the classifier performance are given in Table 5.18. In this configuration the automatic generation for the treatment grid still worked moderately good with the critical misclassification rates in the range of approx. 10 to 15 %. For field application, these are still acceptable rates compared both with values of other classifiers from the literature (cf. [132] and [45]) as well as compared with practical monitoring of manual weeding (cf. [26]). Again, a correlation between the driving speed and the classification accuracy can not be seen in the monitored range.

Further, on 2015/06/10 scans were collected applying the subsampling mode of the camera with

Table 5.19: Result statistics for data collected on 2015/06/10 with camera in subsampling mode.

Driving speed	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
50 mm/s	6.96 % of 23808	**15.35 %** of 593	26.56 % / **18.13 %** of 720
100 mm/s	2.66 % of 11411	**14.61 %** of 394	31.07 % / **19.82 %** of 338
150 mm/s	3.17 % of 44098	**14.54 %** of 1417	24.24 % / **12.36 %** of 1060
200 mm/s	3.83 % of 31039	**33.03 %** of 1562	27.61 % / **10.33 %** of 1094

a higher frame rate of approx. 360 Hz. The higher frame rate of the camera seems to allow higher speeds. Hence, the driving speed of the BoniRob was (partly) increased for these tests with values of 50 mm/s, 100 mm/s, 150 mm/s and 200 mm/s. The result statistics for the classification tests conducted on the data of these tests are given in Table 5.19.

As the result statistics show, for the data acquired in subsampling mode of the camera the critical misclassification rates at all speeds do increase to a level of approx. 15 to 20 %. Particularly, for the speed of 200 mm /s 33 % of carrot cells are illegally treated. This cannot be tolerated. Seemingly, for the classification of carrot plants with the fine feathered, pinnate leafs the full camera resolution is required. The higher density of scans thanks to the higher scan rate in subsampling mode does apparently not matter as much as the reduced resolution of the incoming camera images.

Nevertheless, on both days for full camera resolution acceptable to good results for the classification were achieved at driving speeds of up to 100 mm/s. On both days the frame rate was reduced to 50 and 75 Hz. However, the camera is able to provide full resolution images at 100 Hz with optimal camera setup. Consequently, the same scan density, which provided these results, can be achieved at driving speeds of at least 133 mm/s, maybe even up to 200 mm/s. Driving speeds higher than this are seemingly not possible for carrot classification, though, as higher frame rates cannot be achieved without use of the subsampling mode. However, driving speeds in the range of 100 mm/s or even little more are still a good practicable level for mechanical weed control in carrots as the state-of-the-art manual weeding is also conducted at this level of speed or even slower [26].

- **Up-to-date label data / up-to-date classifier knowledge base**

As shown, using following the In-Field-Labeling concept and generating a classifier based on a couple of labeled marks into sensor data acquired from the same field situation some quite good classification results could be achieved. The In-Field-Labeling concept, thus, provides a practicable solution for keeping the knowledge base of the classifier up-to-date with the respective field situation.

To prove the significance of the up-to-date knowledge base for correct classification experiments using subsets of data of both days were conducted. Both used subsets had been gathered with

Table 5.20: Comparing up-to-date classifiers with wrong classifiers trained with data of the respective other date.

Date of label data for classifier	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
Processing data subset of 2014/06/15:			
2014/06/15 (correct classifier)	1.40 % of 1576	**4.44** of 90	35.52 % / **7.76** of 335
2015/06/10 (wrong classifier)	12.37 % of 1576	**41.11** of 90	37.01 % / **13.43** of 335
Processing data subset of 2015/06/10:			
2015/06/10 (correct classifier)	0.45 % of 2225	**9.62 %** of 52	39.13 % / **2.17 %** of 46
2014/06/15 (wrong classifier)	35.19 % of 2225	**59.62 %** of 52	23.91 % / **8.70 %** of 46

full camera resolution. The subsets were processed through the classification pipeline twice. First, they were processed using the correct classifier, which was trained with label data of the same day. The second processing was conducted using a wrong classifier, which was trained with the label data of the respective other day. For all these processing tests the result statistics were calculated.

Table 5.20 shows the resulting statistics for these tests. First, a data subset collected on 2014/06/15 was processed with the correct classifier created using sample data of that date. Further, it was processed with a wrong classifier derived based on the carrot/weed labelings in data gathered on 2015/06/10. The results of this test can be found at top of Table 5.20. Further, a data subset collected on 2015/06/10 was processed with the correct classifier of that date and with the wrong classifier of 2014/06/15. The results of these processing runs are listed at bottom of Table 5.20.

As the data given in Table 5.20 shows, clearly, on both dates the classifiers trained based on labeling marks in the data of the respective same date significantly outperformed the classifiers of the respective other date - in spite of both classifiers being trained for the same task, i.e., carrot/weed discrimination. Particularly, the rate of misclassified, i.e., treated, carrot cells drastically increases. Contrarily, the number of not treated weed plants does not decrease but increase. This proves the importance of up-to-date classifier knowledge for pixel-based classification of complex classification problems, such as carrot/weed discrimination.

5.3.5 Results from classifying field data of corn salad

As mentioned, the plant classification described here was developed with focus on mechanical weed control for organic cultivation of carrots. However, in October 2015 there was the oppor-

Figure 5.67: Impressions of field situation and tests of corn salad scans. Bottom/left: Sensing head of the MWLP system mounted into BoniRob with lasers turned on (NIR not visible).

tunity to collect field data from scanning corn salad (*Valerianella locusta*) using the MWLP system and BoniRob. Mechanical weed control in corn salad is an issue for both organic and conventional farming. For conventional farming there are only very few herbicides available for this culture. Oftentimes, these do not cover the entire range of weed kinds. The field trials were finally conducted on 2015/10/19.

Figure 5.67 shows some photo impressions of the test for scanning corn salad data conducted that day. The scans were performed again using the red, green and NIR lasers. The camera was operated with optimal setup at its maximum frame rates of approx. 100 Hz for full resolution and approx. 410 Hz in subsampling mode. Further, for these tests the Schneider lens was available and used. Again, different driving speeds have been tested. Figure 5.68 depicts an example of the collected data.

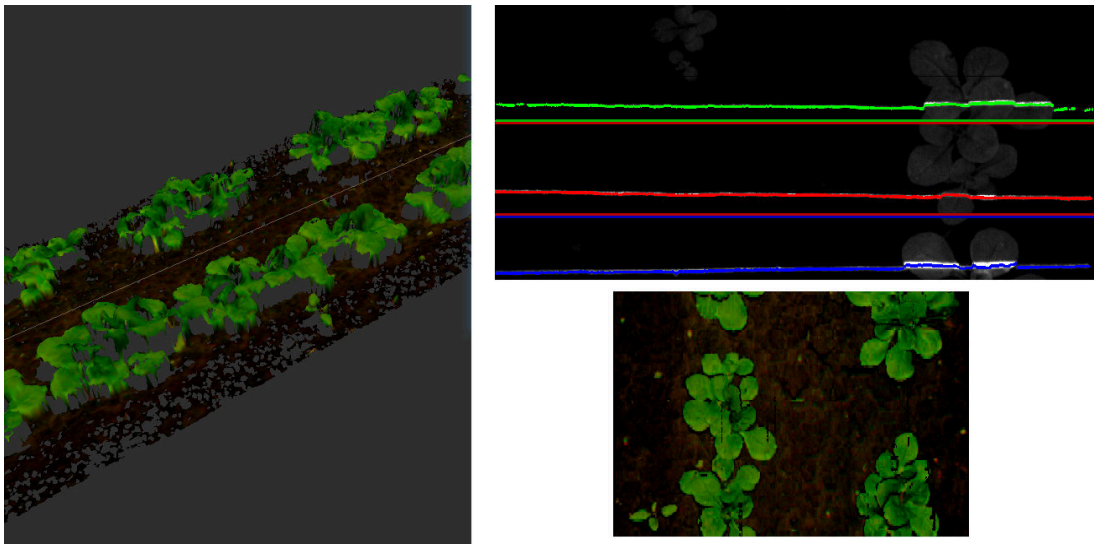


Figure 5.68: Impressions of the collected data. Top/right: line detection in original camera imaged. Bottom/right: Color image visualization of the *IntensitySum* features of the red and green lasers. Left: Meshed 3D visualization of the data.

- **First classification test with corn salad**

Unfortunately[1], the field situation for conducting the tests showed only very few weeds. Therefore, the data was not highly interesting for classification tests. Nevertheless, some simple tests were conducted to give the classifier a shot for this crop it was not specifically geared toward.

Following the In-Field-Labeling concept first a couple of plant segments were annotated with labeling marks. Figure 5.69 depicts a screen shot of the labeling.

The marks and data shown in Figure 5.69 were essentially everything the classifier 'knew' about corn salad when the classifier was feed with the corn salad data in the following test. No parameter changes or even programming changes were performed with respect to the tests on carrot data described in Section 5.3.4. Not even the parameters of the treatment grid aggregation were modified via the respective GUI. The treatment grid aggregation was left at the default configuration for plant classification with crop plant protection shown in Figure 5.25. Figure 5.70 shows the resulting image for this first shot corn salad classification. The original data (colored like in Figure 5.68) is thereby overlayed with the treatment grid finally generated by the classification pipeline. Grid cells to be treated are - again - shaded red.

As Figure 5.70 shows, the classifier exactly hits the two weed plants contained in the data while there are no false-positively treated crop plant cells. Hence, in this case, the automatically obtained classification result is perfect. Despite that there is no statistical significance of

[1] Unfortunately from the point of view of classification tests; fortunately from the point of view of the farmer.

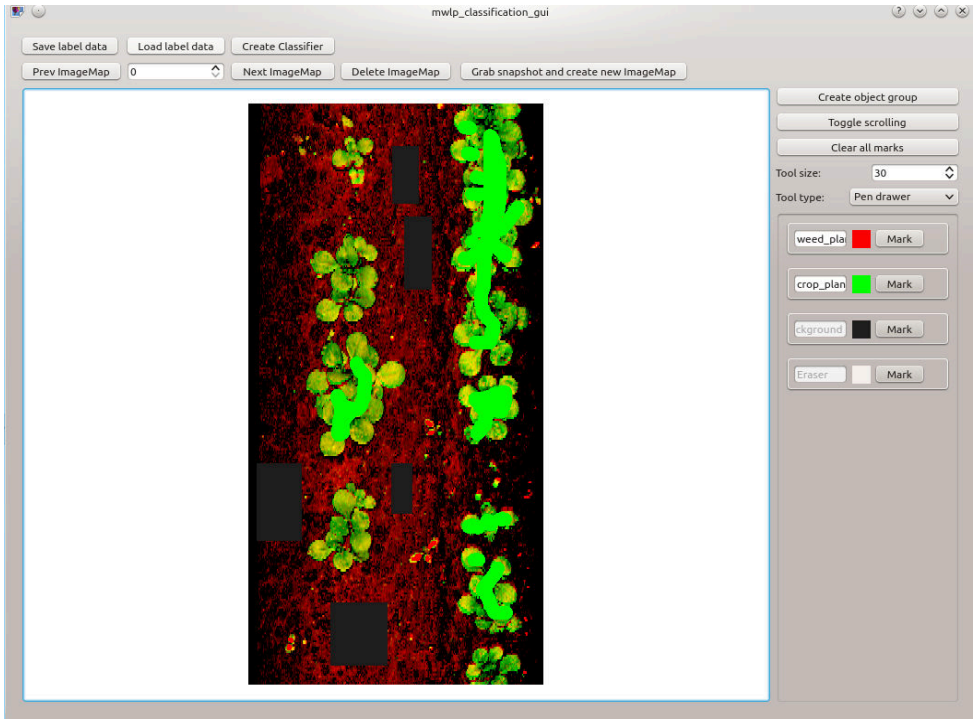


Figure 5.69: The provided label data for the corn salad classification.

such single test, this is still remarkable having in mind the very low effort for adaptation of the classifier to the new crop corn salad. Again, note that the labels in Figure 5.69 represent the entire knowledge the classifier had about corn salad.

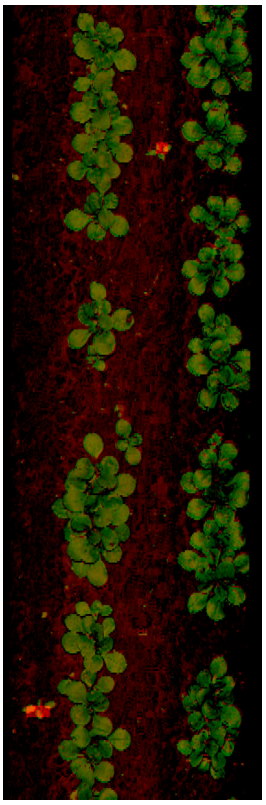


Figure 5.70: Result of the 1st shot of the corn salad classification.

- **Classification results**

Due to the remarkably good result of the first classification test it was probed, whether the good performance of the classifier for the new crop corn salad could further be reproduced on the entire data collected on 2015/10/19.

Tables 5.21 shows the classification results of the data sets acquired. The first two rows indicate the results for scanning at full camera resolution with driving speeds of 50 mm/s and 100 mm/s, respectively. Again, parts of the data that have been used for labeling are excluded from the resulting statistics to avoid optimism. The two bottom rows list the classification results for the data gathered in subsampling mode driving speeds of 150 mm/s and 400 mm/s, respectively. As the subsampling modes allowed higher driving speeds a longer distance was covered in these modes, hence the higher number of samples.

Clearly, in both modes and with driving speeds of up to 400 mm/s the classifier performs remarkably good. Apparently, the corn salad with leafs less-fine structured than those of carrots allows operating the camera in subsampling mode with reduced resolution. Hence, the frame rate can be increased to 410 Hz and allowing even driving speed of 400 mm/s with quite good classification results. Of course, the number of weed plants evaluated for these tests is relatively

Table 5.21: Result statistics for data collected on 2015/10/19.

Driving speed mode	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
50 mm/s / full resolution	0.00 % of 1719	**0.20 %** of 500	66.67 % / **0.00 %** of 3
100 mm/s / full resolution	0.00 % of 4737	**0.00 %** of 2643	50.00 % / **0.00 %** of 12
150 mm/s / subsampling	1.64 % of 16571	**0.72 %** of 6385	21.21 % / **9.09 %** of 66
400 mm/s / subsampling	1.90 % of 30602	**1.55 %** of 14281	16.07 % / **2.35 %** of 85

low due to the low weed pressure on the scanned field. Thus, the statistical significance of the result shown at top of Table 5.21 can partly be doubted. However, this data sets given, the classifier worked very good also for the new crop corn salad, it was not specially designed for.

- **The effect of the In-Field-Labeling concept**

As the classification pipeline worked out even for corn salad, thus a kind of crop different from carrots, it was tested which part of this was due to the In-Field-Labeling concept. For this experiment a data subset collected from corn salad was processed with the correct classifier generated from corn salad labels and with a wrong classifier generated from carrot labels. Vice versa, a data set of carrots was processed with the correct carrot classifier and with the wrong corn salad classifier.

Table 5.22 shows the results of these tests. At its top the results for classifying corn salad data with the correct classifier and with the classifier generated for carrots are compared. Here, the corn salad classifier outperforms. The carrot classifier just finds nothing safe to treat. Thus, the weed control action would have no effect using this classifier. At bottom of Table 5.22 the results of the carrot classification of the tested subset with the correct classifier and with the wrong corn salad classifier are compared. Again, the use of the classifier for the wrong crop results in a drastic increase of weed plants left untreated.

As a consequence of these evaluations, again, the importance of the In-Field-Labeling concept particularly for complex tasks, such as crop/weed classification, is shown. However, looking at the results of the background classification, i.e., soil/biomass discrimination, in Table 5.22 the data also shows that for relatively simple classification problems like soil/plant an up-to-date knowledge-base is not always required.

Based on the tests conducted here and in Section 5.3.4 it can be stated that the combination of the descriptive spectral + 3D data of the MWLP system and the In-Field-Labeling concept provides a powerful and very flexible approach to address the complex classification problem of crop/weed discrimination even using a relatively simple Naive Bayes classifier. Thereby, segmentation and detection of plant objects as well as 'handshaking' each plant object being major error sources in field due to lighting changes and overlapping could completely be avoided. Further, the In-Field-Labeling concept does not only provide flexibility but also gives the user

Table 5.22: Comparing classifiers of correct crop with wrong classifiers trained with data of the respective other crop.

Date/crop of label data for classifier	Misclassification [% of N grid cells]		
	Manual background	Manual good	Manual bad
Processing data subset collected on 2015/10/19 from corn salad:			
2015/10/19 - corn salad (correct classifier)	0.00 % of 1794	**0.00 %** of 1215	40.00 % / **0.00 %** of 5
2014/06/15 - carrots (wrong classifier)	0.00 % of 1794	**0.00 %** of 1215	0.00 % / **100.00 %** of 5
Processing data subset collected on 2014/06/15 from carrots:			
2014/06/15 - carrots (correct classifier)	1.40 % of 1576	**4.44** of 90	35.52 % / **7.76** of 335
2015/10/19 - corn salad (wrong classifier)	0.19 % of 1576	**4.44 %** of 90	10.75 % / **87.46 %** of 335

a tool to cope with situations, in which the initial knowledge of the classifier does not provide sufficient results.

Chapter 6

Discussion and conclusion

Finalizing the dissertation, this chapter is intended to review and summarize the observations made during the realization of the described concepts. Further, an outlook on further actions and investigations to be conducted complementing the results described here is given. New research questions brought up by this work are drafted.

- **Review of the results**

The seed crystal of the work conducted for this dissertation was my idea of extending the LP concept to a system with multiple lasers at different wavelengths and a single monochrome imager, hence the MWLP concept. As shown in Chapter 2, the MWLP concept has a unique set of advantages and disadvantages as compared with other competing and complementing sensor concepts. Hence, under some specific application restrictions and requirements it may outperform other sensor concepts in certain respects. Starting from this draft of the MWLP concept, it was intended to show the feasibility of the concept by setting up a prototype system. Further, this prototype had to be characterized for assessing its accuracies. Finally, in order to demonstrate the use of the sensor data outcome a classification system was set up and connected with the sensor, thus showing the sensor data can help to solve real world classification problems. The following explanations give an overview of the results from these parts of the dissertation (cf. Section 1.2.1).

Realization of the MWLP prototype: Described in Chapter 3, the MWLP system was realized as prototype as part of this work. The prototype allows capturing the camera data of different line lasers and processing the data at 100 Hz in full resolution mode. In subsampling mode of the camera even images at 410 Hz can be processed. Initial thoughts of implementing image processing for line detection on a Field-programmable Gate Array (FPGA) have not been further pursued as the GPU-based implementation - so far - provided sufficient performance. Further, an FPGA implementation would not contribute novel aspects from the point of view of showing the feasibility of the approach. A limitation is the restriction of the search space for matching to x- and y-shifts. I.e., only linear or planar x-/y-movements between sensor and objects are modeled. However, for a wide range of thinkable applications this is sufficient.

Characterization for the MWLP prototype: Described in Chapter 4, the implemented prototype was thoroughly tested. The critical accuracies of line assembly and distance precision are safely in the sub-millimeter range through a wide scope of configurations and operating conditions. Movement speeds of the objects of more than 1.0 m/s have shown feasible. The predicted differences in the scattering behavior at different wavelengths and different materials can be monitored in an image-based manner for multiple wavelengths. Finally, field application of the sensor mounted into a field robot was demonstrated.

Classification of MWLP sensor data: As stated in Section 1.2, the goal of this thesis was not just to implement the sensor concept to show some nicely colored images or point clouds but also to show that the descriptive sensor data can help to solve real world classification problems. Hence, the implemented, connected and tested classification pipeline is described in Chapter 5. For the chosen example applications of potato and plant classification this was shown to be feasible. Thereby, a pixel-based classification was used, thus providing high flexibility for adaptation by the In-Field-Labeling concept. Treatment grids that are ready to serve as input for actuator tools can be generated while object detection and 'hand-shaking' each object (i.e., potato / stone, crop plant / weed plant) are intentionally avoided as these are major error sources for field applications. The classification results have been validated for both applications. Finally, the adaptation of the plant classification from carrots to corn salad by just adding a couple of marks in example data remarks on the flexibility of the concept.

Summarizing the results of this thesis, it can be stated that the MWLP concept is feasible and is a promising approach for solving problems in real world agricultural applications.

- **Future potentials**

As the feasibility of the MWLP approach was shown in this dissertation, a series of future potentials are unlocked. These include possible applications of the sensor in combination with actuators or without actuators.

For applications without actuators the data of the sensor itself is seen as valuable output. For example for quality assurance in industrial processes the indicated results of the sensor are interpreted for process management but do not necessarily cause an immediate reaction.

Examples for applications of the sensor without automatic coupling of an actuator are - for instance - assessment of fruit quality in food processing or plant phenotyping. For the assessment of fruit quality first very simple tests were conducted during the work on this thesis as explained in Section 4.4, e.g. the detection of damages in the apple tissue beneath the surface (cf. Figures 4.28 - 4.30). These showed the potential of fruit quality assessment particularly using the obtained scattering features. However, these tests can only be seen as very simple initial tests. For realistic application correction methods taking into account inclination angles and object color have to be implemented to provide robust results. The input for these correction steps is already provided by the sensor, though. Thus, no additional sensor for inclination correction is required - in contrast to hyperspectral imaging [8].

A further application example without actuator is plant phenotyping for crop plant breeding. In this field no explicit tests have been conducted. However, scanning plants with high resolution under field condition was shown feasible as part of the plant classification tests. Thus, the data scanned from plants could further serve for other purposes. Likely, it is possible to determine a variety of plant parameters based on the 3D data along with reflection and backscattering data at multiple wavelengths, which can be gathered using the MWLP system. In combination with other sensors or fused with a-priori data its usage could help to widen the phenotyping bottleneck.

For applications with actuators there was a wide elaboration on the two classification applications potato sorting and weed control. Further, other sorting applications are possible. The experiments of the two classification applications were validated at the level of the actuator input, as this dissertation is focused on sensor and showing the feasibility of the classification. The finally generated treatment grids, which could serve as input for an actuator, were validated against manually referenced grids. Despite the limited number of manually referenceable grids, the results are promising. To provide higher trust in the system further field tests are required, though. However, the next step toward practical adoption in these fields is to combine the system with an actuator. Conducting more field tests without actuator would be possible, too. However, for these tests a manual referencing of the data for validation would be required and this is only possible up to a certain amount of scanned field data and monitored situations. Thus, an actuator is required, that does an immediate and 'hard' validation of the decisions taken by the sensor and classification pipeline. Consequently, a combination with an actuator would allow agronomic field tests to validate the conclusions made in this dissertation for the observed modules based on the scan data.

Concluding this dissertation, it can be stated that the introduced MWLP approach has been shown to be feasible and to offer many potentials for applications, particularly in the agricultural domain. The flexibility of the sensor in terms of addressing particular wavelengths, FOVs and resolutions allows gearing the system toward the specific application. It enables system-level engineering considering both, the sensor and the data processing chain. Hence, the MWLP approach can provide a fresh impetus to marketable applications of image-based sensor systems in many potential agricultural fields.

Bibliography

[1] M. J. Aitkenhead, I. A. Dalgetty, C. E. Mullins, A. J. S. McDonald, and N. J. C. Strachan. Weed and crop discrimination using image analysis and artificial intelligence methods. *Computers and Electronics in Agriculture*, 39(3):157–171, 2003.

[2] P. Alt. Optisch-elektronische Sortiersysteme in der kartoffelverarbeitenden Industrie. In *Bericht über die 13. Kartoffeltagung*, pages 68–74, Detmold, 1991. Arbeitsgemeinschaft der Kartoffelforschung e. V., Granum-Verlag.

[3] Friedrich.-J. Baartz. Der ColorRanger: hohe Datenraten für 3D und Farbe dank on-chip-processing. `http://spectronet.de/portals/visqua/story_docs/vortraege_2011/110929_farbworkshop_konstanz/110930_14_baartz_chronos_vision.pdf` 6/19/2012.

[4] Waldemar Bangert, Arnd Kielhorn, Florian Rahe, Amos Albert, Peter Biber, Slawomir Grzonka, Sebastian Haug, Andreas Michaels, Daniel Mentrup, Martin Hänsel, Daniel Kinski, Kim Möller, Arno Ruckelshausen, Christian Scholz, Fabian Sellmann, Wolfram Strothmann, and Dieter Trautz. Field-Robot-Based Agriculture: "RemoteFarming.1" and "BoniRob-Apps". In *71th conference LAND.TECHNIK-AgEng 2013*, pages 439–446, Düsseldorf, 2013. VDI Verlag GmbH.

[5] Norbert Bauer. *Leitfaden zu Grundlagen und Anwendungen der optischen 3-D-Messtechnik*. Fraunhofer Allianz Vision, Erlangen, 2003.

[6] Norbert Bauer. *Leitfaden zu praktischen Anwendungen der Bildverarbeitung*. Fraunhofer Allianz Vision, Erlangen, 2003.

[7] Bryce E. Bayer. Color imaging array, July 20 1976. US Patent 3,971,065.

[8] Jan Behmann, Anne-Katrin Mahlein, Stefan Paulus, Jan Dupuis, Heiner Kuhlmann, Erich-Christian Oerke, and Lutz Plümer. Generation and application of hyperspectral 3D plant models: methods and challenges. *Machine Vision and Applications*, pages 1–14, 2015.

[9] P. Berghmans, C. Fivez, and J. Speybrouck. Method for sorting potato products and sorting apparatus for potato products, January 26 2012. US Patent App. 13/258,276.

[10] P. Berghmans and M. Ruymen. Method and device for sorting products, May 19 2010. EP Patent App. EP20,080,447,046.

[11] David I. Bevan. Distributed garbage collection using reference counting. In *PARLE Parallel Architectures and Languages Europe*, pages 176–187. Springer, 1987.

[12] François Blais, John Taylor, Luc Cournoyer, Michel Picard, Louis Borgeat, Louis-Guy Dicaire, Marc Rioux, Jean-Angelo Beraldin, Guy Godin, C Lahnanier, et al. Ultra-high resolution imaging at 50μm using a portable XYZ-RGB color laser scanner. 2005. `http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=5765101` 6/26/2015.

[13] A. Bligh, D. Meagher, and M. Moynihan. An air spring pneumatic product rejection system, March 25 2009. EP Patent 1,605,170.

[14] H. Bösch. Optoelektronisches Ausleseverfahren - Problemstellung, Auswahlkriterien und Anwendung. In *Bericht über die 16. Kartoffeltagung*, pages 79–90, Detmold, 1994. Arbeitsgemeinschaft der Kartoffelforschung e. V., Granum-Verlag.

[15] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[17] Xavier P. Burgos-Artizzu, Angela Ribeiro, Alberto Tellaeche, Gonzalo Pajares, and Cesar Fernández-Quintanilla. Improving weed pressure assessment using digital images from an experience-based reasoning approach. *Computers and electronics in agriculture*, 65(2):176–185, 2009.

[18] Lucas Busemeyer, Daniel Mentrup, Kim Möller, Erik Wunder, Katharina Alheit, Volker Hahn, Hans Peter Maurer, Jochen C. Reif, Tobias Würschum, Joachim Müller, Florian Rahe, and Arno Ruckelshausen. Breedvision - A multi-sensor platform for non-destructive field-based phenotyping in plant breeding. *Sensors*, 13(3):2830–2847, 2013.

[19] Grimme Landmaschinenfabrik GmbH & Co.KG. Neuheit: AirSep - Steine und feuchte Kluten per Luftstrom trennen, 2013. `http://www.grimme.com/de/news/neuartiges-beimengentrenngeraet-airsep-fuer-kartoffelroder.94.html` 10/08/2015.

[20] Teledyne Dalsa. Genie Cameras. `http://www.teledynedalsa.com/imaging/products/cameras/area-scan/genie/` 12/22/2015.

[21] François-Michel De Rainville, Audrey Durand, Félix-Antoine Fortin, Kevin Tanguy, Xavier Maldague, Bernard Panneton, and Marie-Josée Simard. Bayesian classification and unsupervised learning for isolating weeds in row crops. *Pattern Analysis and Applications*, 17(2):401–414, 2014.

[22] Jan Dupuis, Stefan Paulus, Anne-Kathrin Mahlein, Heiner Kuhlmann, and Thomas Eichert. The impact of different leaf surface tissues on active 3D laser triangulation measurements. *Photogrammetrie - Fernerkundung - Geoinformation*, 6:437–447, 2015.

[23] Martin Egbers, Christoph-Frederick Kronsbein, and Arno Ruckelshausen. Der Reifegrad bestimmt die Häcksellänge. *Landtechnik*, 61(3):136–137, 2006.

[24] Quality Engineering. Schlüsselfunktion der 3D-Lasermesstechnik: Berührungslose Geometrie- und Konturkontrolle mit Laserlichtschnitt-Verfahren, 2004. http://www.qe-online.de/artikelarchiv/-/journal_content/56/12275/424708/Schl%C3%BCsselfunktion-der-3D%E2%80%93Lasermesstechnik/ 7/7/2015.

[25] Michael Ferguson, Paul Bouchier, and Mike Purvis. Package Summary: rosserial, 2015. http://wiki.ros.org/rosserial 7/13/2015.

[26] Susanne Fittje, Martin Hänsel, Frederik Langsenkamp, Arnd Kielhorn, Maik Kohlbrecher, Maria Vergara, and Dieter Trautz. Praxiserhebungen zu Aufwand und Erfolg der Handjäte in Möhren unter ökologischer Bewirtschaftung. In *Am Mut hängt der Erfolg - Rückblicke und Ausblicke auf die ökologische Landbewirtschaftung. Beiträge zur 13. Wissenschaftstagung Ökologischer Landbau*. Verlag Dr. Köster, Berlin, 2015.

[27] Tully Foote. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6, April 2013.

[28] Tully Foote, Eitan Marder-Eppstein, and Wim Meeussen. Package Summary: tf, 2015. wiki.ros.org/tf 10/02/2015.

[29] Tully Foote, Eitan Marder-Eppstein, and Wim Meeussen. tf2, 2015. wiki.ros.org/tf2 10/02/2015.

[30] Tully Foote, Radu Bogdan Rusu, and Esteve Fernandez. Package Summary: nodelet, 2015. http://wiki.ros.org/nodelet 7/15/2015.

[31] Open Perception Foundation. pcl::PFHEstimation< PointInT, PointNT, PointOutT > Class Template Reference. http://docs.pointclouds.org/1.7.2/a02406.html 01/08/2016.

[32] Thomas Fricke, Felix Richter, and Michael Wachendorf. Assessment of forage mass from grassland swards by height measurement using an ultrasonic sensor. *Computers and electronics in agriculture*, 79(2):142–152, 2011.

[33] Fachhochschule Osnabrück Fakultät für Ingenieurwissenschaften und Informatik. Forschungsbericht 2006- 2007, 2007. https://www.ecs.hs-osnabrueck.de/uploads/media/iui_forschungsbericht_2006_07_web.pdf 10/08/2015.

[34] Baumer Optotronic GmbH. Technical Data HXG20NIR, 2012. http://ftp.elvitec.fr/Baumer/MANUELS/HXG/TDS_HXG20NIR_v10e_120201.pdf 7/10/2015.

[35] Baumer Optotronic GmbH. "Intelligente Softwareintegration" Das neue Baumer GAPI SDK, 2014. http://www.baumer.com/de-de/produkte/identifikation-bildverarbeitung/software-und-starter-kits/baumer-gapi-sdk/ 7/13/2015.

[36] DAVID Vision Systems GmbH. David SLS-2 Structured Light Scanner, 2015. http://www.david-3d.com/de/products/sls-2 6/26/2015.

[37] Raytrix GmbH. Raytrix Lightfield Camera, 2012. http://raytrix.de/tl_files/downloads/Raytrix_Slides.pdf 6/26/2015.

[38] Tilo Gockel. *Interaktive 3D-Modellerfassung mittels One-Shot-Musterprojektion und Schneller Registrierung*. Universitätsverlag Karlsruhe, Karlsruhe, 2006.

[39] Dennis H. Goldstein, David B. Chenault, and J. L. Pezzaniti. Polarimetric characterization of Spectralon. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 126–136. International Society for Optics and Photonics, 1999.

[40] The PostgreSQL Global Development Group. PostgreSQL, 2015. http://www.postgresql.org/ 12/23/2015.

[41] H. U. Haase. Die instrumentelle Farbmessung in der kartoffelverarbeitenden Industrie. In *Bericht über die 17. Kartoffeltagung*, pages 68–75, Detmold, 1995. Arbeitsgemeinschaft der Kartoffelforschung e. V., Granum-Verlag.

[42] Nathan Hagen and Michael W. Kudenov. Review of snapshot spectral imaging technologies. *Optical Engineering*, 52(9):090901–090901, 2013.

[43] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[44] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[45] Sebastian Haug, Andreas Michaels, Peter Biber, and Jorn Ostermann. Plant classification system for crop/weed discrimination without segmentation. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 1142–1149. IEEE, 2014.

[46] Jochen Hemming and Thomas Rath. PA - Precision Agriculture: Computer-Vision-based Weed Identification under Field Conditions using Controlled Lighting. *Journal of agricultural engineering research*, 78(3):233–243, 2001.

[47] Dave Hershberger, David Gossow, Josh Faust, and William Woodall. Package Summary: rviz, 2015. http://wiki.ros.org/rviz 12/23/2015.

[48] Hitachi. Nippon Signal MEMS 3D laser sensor. http://www.hitachi-hightech.com/us/product_detail/?pn=ind-hta-ele-002 12/21/2015.

[49] Dirk Holz and Sven Behnke. Fast Range Image Segmentation and Smoothing using Approximate Surface Reconstruction and Region Growing. In *Proceedings of the 12th International Conference on Intelligent Autonomous Systems (IAS), Jeju Island, Korea, June 26-29 2012*, 2012.

[50] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using RGB-D cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer, 2012.

[51] Stefan Holzer, Radu Bogdan Rusu, M. Dixon, Suat Gedikli, and Nassir Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2684–2689. IEEE, 2012.

[52] Grégoire Hummel, Maximiliano Pin, Nico Wagnitz, and Uladzimir Zhokhavets. PlantEye - ein kommerzieller 3D Laser Scanner für automatisiertes Screening von Pflanzen. *Bornimer Agrartechnische Berichte*, 81:235–240, 2013.

[53] IFM. OD3200 | PMD 3D Sensor. `http://www.ifm.com/products/de/ds/O3D200.htm` 12/21/2015.

[54] IMEC. IMEC HYPERSPECTRAL MOSAIC SNAPSHOT IMAGER. `http://www2.imec.be/content/user/File/Brochures/2015/5_flyer_HSI_Snapshot%20mosaic%20sensor.pdf` 6/24/2015.

[55] Sascha In der Stroth, Bernd Ramler, Andreas Linz, and Arno Ruckelshausen. Weed detection based on spectral imaging systems with CMOS-cameras. In *4th European Conference on Precision Agriculture ECPA*. 2003.

[56] Indiamart. Micro Epsilon Laser Line Profile Sensors, 2015. `http://www.indiamart.com/microepsilonmesstechnikgmbh/laser-line-profile-sensors.html` 7/7/2015.

[57] Quest Innovations. Multispectral Cameras. `http://www.quest-innovations.com/cameras/multispectral-cameras` 6/25/2015.

[58] András Jung, René Michels, and Rainer Graser. Nich-scannende hyperspektrale Kamera für UAS Plattformen. *Bornimer Agrartechnische Berichte*, 81:141–147, 2013.

[59] András Jung, René Michels, and Rainer Graser. Räumlich und spektral auflösende Hyperspektralkamera und Verfahren, November 18 2015. EP Patent App. EP20,140,001,857.

[60] Klemens Kalverkamp, Franz-Josef Dettmer, and Christian Döhmann. Trennvorrichtung für eine Kartoffelerntemaschine oder eine Aufbereitungsmaschine, May 30 2014. WO Patent App. PCT/EP2013/002,447.

[61] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors, A Hands-on Approach*. Morgan Kaufmann Publishers, Amsterdam and other, 2010.

[62] Ralph Klose, Jaime Penlington, and Arno Ruckelshausen. Usability study of 3D time-of-flight cameras for automatic plant phenotyping. *Bornimer Agrartechnische Berichte*, 69:93–105, 2009.

[63] Andreas Kolb, Erhardt Barth, and Reinhard Koch. ToF-sensors: New dimensions for realism and interactivity. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–6. IEEE, 2008.

[64] Peter Kraemmer, Hartmut Bartelt, Helmut Fischer, and Bernard Schmauss. Imaging in scattering media using the phase of modulated light sources. In *International Symposium on Biomedical Optics Europe'94*, pages 65–74. International Society for Optics and Photonics, 1995.

[65] Christoph-Frederik Kronsbein, Frank Volke, Daniel Schmitt, Martin Benecke, Bernd Schniederbruns, Andre Berghaus, Jaron Martinez, Henrik Hufendiek, and Arno Ruckelshausen. Magnetische Resonanz- und Nah-Infrarot- Sensorsysteme zur Online-Messung der Feuchte bei ErntegÃ$\frac{1}{4}$tern. In *Massendatenmanagement in der Agrar- und Ernährungswirtschaft, 33. GIL-Jahrestagung*, pages 175–178, Potsdam, 2013.

[66] David Kushner. The Making of Arduino, 2011. `http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino` 7/13/2015.

[67] Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on Parzen window. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(12):1667–1671, 2002.

[68] Hermann Laber and Hartmut Stützel. Ertragswirksamkeit der Restverunkrautung in Gemüsekulturen nach nichtchemischen Unkrautregulationsmaßnahmen. *Pflanzenbauwissenschaften*, 7(1):29–38, 2003.

[69] Frederik Langsenkamp, Fabian Sellmann, Maik Kohlbrecher, Arnd Kielhorn, Wolfram Strothmann, Andreas Michaels, Arno Ruckelshausen, and Dieter Trautz. Tube Stamp for mechanical intra-row individual Plant Weed Control, 2014. `http://http://www.ecs.hs-osnabrueck.de/fileadmin/groups/156/Veroeffentlichungen/2014-CIGR_2014_Tube_Stamp_for_mechanical_intra-row_individual_Plant_Weed_Control.pdf` 11/17/2015.

[70] Pierre-Jean Lapray, Xingbo Wang, Jean-Baptiste Thomas, and Pierre Gouton. Multispectral Filter Arrays: Recent Advances and Practical Implementation. *Sensors*, 14(11):21626, 2014.

[71] Leuze. Light-section sensors, 2015. `http://www.leuze.com/en/deutschland/produkte/industrielle_bildverarbeitung/lichtschnittsensoren_3/index.php` 7/7/2015.

[72] LMI3D. Gocator 2300 Series Datasheet, 2015. `http://downloads.lmi3d.com/system/files/Gocator/documents/Gocator%202300%20Series/DATASHEET_Gocator_2300_WEB_EN.pdf` 7/7/2015.

[73] LMI3D. Gocator Profile Sensors, 2015. `http://www.lmi3d.com/products/gocator/profile-sensor#` 7/7/2015.

[74] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[75] Delia Lorente, Manuela Zude, Christine Idler, J. Gómez-Sanchis, and J. Blasco. Laser-light backscattering imaging for early decay detection in citrus fruit using both a statistical and a physical model. *Journal of Food Engineering*, 154:76–85, 2015.

[76] Delia Lorente, Manuela Zude, C. Regen, L. Palou, J. Gómez-Sanchis, and J. Blasco. Early decay detection in citrus fruit using laser-light backscattering imaging. *Postharvest Biology and Technology*, 86:424–430, 2013.

[77] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[78] Justin Luitjens. CUDA STREAMS. BEST PRACTICES AND COMMON PITFALLS, 2014. `http://on-demand.gputechconf.com/gtc/2014/presentations/S4158-cuda-streams-best-practices-common-pitfalls.pdf` 7/13/2015.

[79] Khalid Makeen, Simon Kerssen, Daniel Mentrup, Bengt Oelmann, and Arno Ruckelshausen. Multiple Reflection Ultrasonic Sensor System for Morphological Plant Parameters. *Bornimer Agrartechnische Berichte*, 78:110–116, 2012.

[80] Richard Mander. Visualant's ChromaId Technology, 2013. `http://www.visualant.net/resources/Visualant_ChromaID_Technical_White_Paper.pdf` 3/5/2014.

[81] Melvin Earl Maron. Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417, 1961.

[82] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. Combined 2D–3D categorization and classification for multimodal perception systems. *The International Journal of Robotics Research*, 30(11):1378–1402, 2011.

[83] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3D modelling of novel objects from a single view. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3700–3705. IEEE, 2010.

[84] Patrick Mihelich, James Bowman, and Vincent Rabaud. Package Summary: camera_calibration, 2015. `http://wiki.ros.org/camera_calibration` 7/28/2015.

[85] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[86] H. B. Mitchell. *Multi-Sensor Data Fusion*. Springer, Berlin, 2007.

[87] D. Moshou, D. Kateris, X. E. Pantazi, and I. Gravalos. Crop and weed species recognition based on hyperspectral sensing and active learning. In *Precision agriculture*, volume 13, pages 555–561. Springer, 2013.

[88] Thomas Mueller, Benjamin Langmann, and Eduard Reithmeier. Development of a measurement system for the online inspection of microstructured surfaces in harsh industrial conditions. In *SPIE Photonics Europe*, pages 91411F–91411F. International Society for Optics and Photonics, 2014.

[89] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008.

[90] Albrecht Nitsch. *Praxishandbuch Kartoffelanbau*. ERLING Verlag GmbH & Co.KG, 2013 edition, 2013.

[91] Hyun Kwon Noh and Renfu Lu. Hyperspectral laser-induced fluorescence imaging for assessing apple fruit quality. *Postharvest Biology and Technology*, 43(2):193–201, 2007.

[92] NVIDIA. NVIDIA CUDA Runtime API, 2015. `http://docs.nvidia.com/cuda/cuda-runtime-api/` 7/13/2015.

[93] Horst Oebel. *Teilschlagspezifische Unkrautbekämpfung durch raumbezogene Bildverarbeitung im Offline-und (Online-) Verfahren (TURBO)*. Dissertation, Universität Hohenheim, 2006. `http://opus.uni-hohenheim.de/volltexte/2006/150/` 11/30/2015.

[94] David Ooms, Frédéric Lebeau, R Ruter, and M-F Destain. Measurements of the horizontal sprayer boom movements by sensor data fusion. *Computers and electronics in agriculture*, 33(2):139–162, 2002.

[95] Peter Opuchlik. Realisierung und Analyse eines 3D Laser Scanners, 2009. `http://www.informatik.uni-konstanz.de/fileadmin/informatik/ag-saupe/Webpages/lehre/ws08-master-bachelor-projekt/BachelorPeterOpuchlik.pdf.lic` 4/7/2015.

[96] Vincent C. Paquit, Kenneth W. Tobin, Jeffery R. Price, and Fabrice Mériaudeau. 3D and multispectral imaging for subcutaneous veins detection. *Optics express*, 17(14):11360–11365, 2009.

[97] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, pages 1065–1076, 1962.

[98] Julio C. Pastrana and Thomas Rath. Erkennung überlappender Pflanzen mit Hilfe von 'Active Shape Models'. In *17. und 18. Workshop Computer-Bildanalyse in der Landwirtschaft Computerised Image Analysis In Agriculture*, pages 117–123, 2012.

[99] Stefan Paulus, Jan Dupuis, Anne-Katrin Mahlein, and Heiner Kuhlmann. Surface feature based classification of plant organs from 3D laserscanned point clouds for plant phenotyping. *BMC bioinformatics*, 14(1):238, 2013.

[100] Stefan Paulus, Jan Dupuis, Sebastian Riedel, and Heiner Kuhlmann. Automated analysis of barley organs using 3D laser scanning: An approach for high throughput phenotyping. *Sensors*, 14(7):12670–12686, 2014.

[101] Stefan Paulus, Thomas Eichert, Heiner E Goldbach, and Heiner Kuhlmann. Limits of active laser triangulation as an instrument for high precision plant imaging. *Sensors*, 14(2):2489–2509, 2014.

[102] Stefan Paulus, Henrik Schumann, Heiner Kuhlmann, and Jens Léon. High-precision laser scanning system for capturing 3D plant architecture and analysing growth of cereal plants. *Biosystems Engineering*, 121:1–11, 2014.

[103] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

[104] Pepperl+Fuchs. LR300 - Laser Light Section Sensor for 3D Measurement, 2015. `http://www.pepperl-fuchs.us/usa/en/16230.htm` 7/7/2015.

[105] PerkinElmer. Data sheet: VariSpec Liquid Crystal Tunable Filters. `http://www.perkinelmer.com/CMSResources/Images/44-140156DTS_010053A_01_VariSpec_DTS.pdf` 12/22/2015.

[106] Maria Persson and Björn Åstrand. Classification of crops and weeds extracted by active shape models. *Biosystems Engineering*, 100(4):484–497, 2008.

[107] PIXELTEQ. SpectroCam | VIS-NIR. `http://www.pixelteq.com/product/spectral-cameras/` 6/24/2015.

[108] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[109] Jörg Rahnenführer. Statistical methods for model selection, 2012. `http://sfb876.tu-dortmund.de/SummerSchool2012/topics.html` 10/02/2015.

[110] Thomas Rath and Julio Pastrana. Mensch vs Computer: Bildverarbeitung zur Erkennung überlappender Pflanzen. *LANDTECHNIK*, 67(3):172–178, 2012.

[111] Stefan Reusch. Use of ultrasonic transducers for on-line biomass estimation in winter wheat. *JIAC2009 Book of Abstracts; Lokhorst, CJFM, Huijsmans, RdL, Eds*, pages 169–175, 2009.

[112] U. Riese and J. Poller. Optoelektronische Sortiereinrichtung für automatische Qualitätsverlesung. In *Bericht über die 16. Kartoffeltagung*, pages 69–78, Detmold, 1994. Arbeitsgemeinschaft der Kartoffelforschung e. V., Granum-Verlag.

[113] Bosch Deepfield Robotics. Weeding - Automated Weed Control For Better Environmental Protection, 2015. `http://www.deepfield-robotics.com/en/Weeding.html` 11/17/2015.

[114] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[115] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[116] Arno Ruckelshausen. Neue Sensorentwicklungen - ein technischer Blick auf Pflanzen, Sensoren und Daten. *Journal für Kulturpflanzen*, 66(2):73–79, 2014.

[117] Arno Ruckelshausen, Peter Biber, Michael Dorna, Holger Gremmes, Ralph Klose, Andreas Linz, Florian Rahe, Rainer Resch, Marius Thiel, Dieter Trautz, and Ulrich Weiss. BoniRob–an autonomous field robot platform for individual plant phenotyping. volume 9, pages 841–847, 2009.

[118] Arno Ruckelshausen and Lucas Busemeyer. Toward Digital and Image-Based Phenotyping. In *Phenomics in Crop Plants: Trends, Options and Limitations*, pages 41–60. Springer, 2015.

[119] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[120] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[121] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.

[122] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010.

[123] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[124] Radu Bogdan Rusu, Andreas Holzbach, Nico Blodow, and Michael Beetz. Fast geometric point labeling using conditional random fields. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 7–12. IEEE, 2009.

[125] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 643–650. IEEE, 2008.

[126] Yoshitaro Sakata, Kazufumi Sakai, and Kazuhiro Nonaka. Inspection technique of latent flaws on fine polished glass substrates using stress-induced light scattering method. In *SPIE Photonics Europe*, pages 913209–913209. International Society for Optics and Photonics, 2014.

[127] Jason Sanders and Edward Kandrot. *CUDA BY EXAMPLE. An Introduction to General-Purpose GPU Programming*, volume 2. Addison-Wesley, Upper Saddle River, NJ and other, 2010.

[128] Christian Scholz, Wolfram Strothmann, Fabian Lankenau, Daniel Schmunkamp, Fabian Sellmann, and Arno Ruckelshausen. Sensorteststand zur Evaluierung bildgebender Sensoren mit dynamisch-reproduzierbaren Störgrößen. In *22. Workshop Computer-Bildanalyse in der Landwirtschaft, 21.04.2016*, Werningerrode, 2016. To be published 2017.

[129] Ulrich Schwesinger, Cedric Pradalier, and Roland Siegwart. A novel approach for steering wheel synchronization with velocity/acceleration limits and mechanical constraints. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5360–5366. IEEE, 2012.

[130] Conrad Electronic SE. IR-Emitter 940 nm 40 ° 5 mm Kingbright BL0106-15-28. `https://www.conrad.de/de/ir-emitter-940-nm-40-5-mm-kingbright-bl0106-15-28-149241.html` 01/06/2016.

[131] Fabian Sellmann, Waldemar Bangert, Slawomir Grzonka, Martin Hänsel, Sebastian Haug, Arnd Kielhorn, Andreas Michaels, Kim Möller, Florian Rahe, Wolfram Strothmann, Dieter Trautz, and Arno Ruckelshausen. Human-machine interaction for a field-robot-based weed control application in organic farming. In *Proceedings of 4th Conference on Machine Control & Guidance*, pages 36–42, 2014.

[132] D. C. Slaughter, D. K. Giles, and D. Downey. Autonomous robotic weed control systems: A review. *Computers and electronics in agriculture*, 61(1):63–78, 2008.

[133] Wolfram Strothmann. Entwicklung einer bildverarbeitungsbasierten Methode zur Erfassung der Querverteilungsgenauigkeit von Zentrifugaldüngerstreuern. Master's thesis, University of Applied Sciences Osnabrück, October 2011.

[134] Wolfram Strothmann, Arnd Kielhorn, Fabian Sellmann, Kim Möller, Martin Hänsel, Dieter Trautz, and Arno Ruckelshausen. Mensch-Maschine-Schnittstelle zur Bildverarbeitung im RemoteFarming. *Bornimer Agrartechnische Berichte*, 81:41–49, 2013.

[135] Wolfram Strothmann, Arnd Kielhorn, Vadim Tsukor, Dieter Trautz, and Arno Ruckelshausen. Interactive Image Segmentation for Model Adaption and Decision Support. In *9th European Conference on Precision Agriculture, Book of Posters*, pages 95–96, 2013.

[136] Wolfram Strothmann, Karsten Morisse, and Arno Ruckelshausen. Smartphone-basierte Bildverarbeitung zur Erfassung der Querverteilungsgenauigkeit von Zentrifugaldüngerstreuern. *Bornimer Agrartechnische Berichte*, 78:124–132, 2012.

[137] Wolfram Strothmann, Arno Ruckelshausen, and Joachim Hertzberg. Multiwavelength laser line profile sensing for agricultural crop characterization. In *SPIE Photonics Europe*, page 91411K. International Society for Optics and Photonics, 2014.

[138] Wolfram Strothmann, Vadim Tsukor, Joachim Hertzberg, and Arno Ruckelshausen. Konfigurationsmöglichkeiten und Datenkonzepte des Multiwavelength Line Profiling (MWLP) Systems. In *Bornimer Agrartechnische Berichte*, volume 88, pages 42–52, May 2015.

[139] Wolfram Strothmann, Vadim Tsukor, and Arno Ruckelshausen. In-Field-Labeling-HMI für automatische Klassifizierung bei der Pflanzen- und Erntegutcharakterisierung mittels bildgebender Sensordaten. In *Informatik in der Land-, Forst-, und Ernährungswirtschaft, Referate der 35. GIL-Jahrestagung*, pages 177–180, Geisenheim, February 2015.

[140] Nicolaas Tack, Andy Lambrechts, Philippe Soussan, and Luc Haspeslagh. A compact, high-speed, and low-cost hyperspectral imager. In *SPIE OPTO*, pages 82660Q–82660Q. International Society for Optics and Photonics, 2012.

[141] OpenCV Dev Team. OpenCV 2.4.11.0 documentation, 2015. `http://docs.opencv.org/modules/gpu/doc/per_element_operations.html#gpu-add` 7/14/2015.

[142] OpenCV Dev Team. OpenCV 2.4.11.0 documentation, 2015. `docs.opencv.org/modules/gpu/doc/data_structures.html#gpu-gpumat` 7/14/2015.

[143] OpenCV Dev Team. OpenCV 2.4.11.0 documentation, 2015. `http://docs.opencv.org/modules/nonfree/doc/feature_detection.html?highlight=sift#sift` 7/16/2015.

[144] OpenCV Dev Team. OpenCV 2.4.11.0 documentation, 2015. `http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html#orb` 7/16/2015.

[145] OpenCV Dev Team. OpenCV 2.4.11.0 documentation, 2015. `http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html?highlight=fast` 7/16/2015.

[146] OpenCV Dev Team. OpenCV 2.4.11.0 documentation, 2015. `http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html?highlight=cvtcolor#cvtcolor` 7/17/2015.

[147] Marius Thiel, Thomas Rath, and Arno Ruckelshausen. Plant moisture measurement in field trials based on NIR spectral imaging - A feasability study. In *Proceedings of 2nd International Workshop on Computer Image Analysis in Agriculture*, pages 16–29, 2010.

[148] Dirk Thomas. Parameter server, 2015. `http://wiki.ros.org/Parameter%20Server` 12/23/2015.

[149] Dirk Thomas and Aaron Blasdel. Package Summary: rqt_image_view, 2015. `http://wiki.ros.org/rqt_image_view` 12/23/2015.

[150] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). 2005.

[151] Emanuele Trucco and Alessandro Verri. *Introductory techniques for 3-D computer vision*, volume 201. Prentice Hall Englewood Cliffs, 1998.

[152] Vadim Tsukor, Wolfram Strothmann, Witali Schwamm, and Arno Ruckelshausen. Contactless Sensor System for Row Navigation and Automatic Depth Control for a Sugar Beet Harvester using a 3D Time of Flight (ToF) Camera. *CIGR Proceedings*, 1(1), 2014.

[153] Vadim Tsukor, Wolfram Strothmann, Erik Wunder, Witali Schwamm, and Arno Ruckelshausen. Cost efficient surface condition determination system for sugar beets for a harvester cleaning control using an industrial RGB camera. In *18th World Congress of CIGR, CIGR2014, Beijing, China*, 2014. `http://www.ecs.hs-osnabrueck.de/fileadmin/groups/156/Veroeffentlichungen/2014_CIGR2014_Tsukor.pdf` 01/10/2016.

[154] Gong Wei, Song Shalei, Zhu Bo, Shi Shuo, Li Faquan, and Cheng Xuewu. Multi-wavelength canopy LiDAR for remote sensing of vegetation: Design and system performance. *ISPRS Journal of Photogrammetry and Remote Sensing*, 69:1–9, 2012.

[155] Martin Weis. *An image analysis and classification system for automatic weed species identification in different crops for precision weed management*. PhD thesis, University of Hohenheim, 2010.

[156] Martin Weis, D. Andújar, G. G. Peteinatos, and R. Gerhards. Improving the determination of plant characteristics by fusion of four different sensors. In *Precision agriculture*, volume 13, pages 63–69. Wageningen Academic Publishers, 2013.

[157] Martin Weis, Christoph Gutjahr, Victor Rueda Ayala, Roland Gerhards, Carina Ritter, and Florian Schölderle. Precision farming for weed management: techniques. *Gesunde Pflanzen*, 60(4):171–181, 2008.

[158] Martin Weis and Markus Sökefeld. Detection and identification of weeds. In *Precision Crop Protection-the Challenge and Use of Heterogeneity*, pages 119–134. Springer, 2010.

[159] Ulrich Weiss, Peter Biber, Stefan Laible, Karsten Bohlmann, and Andreas Zell. Plant species classification using a 3d lidar sensor and machine learning. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 339–345. IEEE, 2010.

[160] Thomas Wiemann, Hendrik Annuth, Kai Lingemann, and Joachim Hertzberg. An Extended Evaluation of Open Source Surface Reconstruction Software for Robotic Applications. *Journal of Intelligent & Robotic Systems*, 77(1):149–170, 2015.

[161] Thomas Wiemann, Andreas Nüchter, and Joachim Hertzberg. A toolkit for automatic generation of polygonal maps–las vegas reconstruction. *ROBOTIK 2012*, 2012.

[162] Erik Wunder, Arnd Kielhorn, Ralph Klose, Marius Thiel, and Arno Ruckelshausen. GIS- and sensor-based technologies for individual plant agriculture. *Landtechnik*, 67(1), 2012.

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Osnabrück, 07/22/2016