Karlsruhe Institute of Technology

Development of a Robotic Shared Autonomy Object Recognition Pipeline for Image Streams

Diplomarbeit of

Andreas Lars Wachaja

At the Department of Mechanical Engineering
Institute of Measurement and Control

Reviewer:	Prof. Dr.-Ing. Christoph Stiller
Advisor:	Sarah Osentoski, Ph. D.

Duration: December 3, 2012 – June 03, 2013

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 06/03/2013

.....

(**Andreas Lars Wachaja**)

Contents

1. Introduction **5**
- 1.1. Problem Definition . . . 6
- 1.2. Concept Overview . . . 9
  - 1.2.1. Weed Detection Pipeline . . . 9
  - 1.2.2. User Integration . . . 10
- 1.3. Thesis Structure . . . 11
- 1.4. Project Acknowledgements . . . 11

2. Background **13**
- 2.1. Shared Autonomy . . . 13
  - 2.1.1. Interfaces for Human-robot Interaction . . . 14
  - 2.1.2. Level and Behavior of Autonomy . . . 15
  - 2.1.3. Nature of Information Exchange . . . 16
  - 2.1.4. Shared Autonomy Applications . . . 17
- 2.2. Precision Agriculture . . . 19
  - 2.2.1. Methods . . . 19
  - 2.2.2. Trends . . . 20
- 2.3. Autonomous Weed Control . . . 20
  - 2.3.1. State of the Art . . . 20
  - 2.3.2. Commercial Products . . . 22
  - 2.3.3. Weed Detection and Identification . . . 23

3. Shared Autonomy Approach **25**
- 3.1. User Integration . . . 27
  - 3.1.1. Human Insertion Points . . . 27
  - 3.1.2. Realization . . . 28
- 3.2. User-classifier Interaction . . . 28
  - 3.2.1. Interaction Scenarios . . . 29
  - 3.2.2. Realization . . . 30
- 3.3. User Interface . . . 32
  - 3.3.1. Interface Concepts . . . 32
  - 3.3.2. Realization . . . 32

4. Concept of Tile Images **37**
- 4.1. Advantages . . . 37
- 4.2. Image and Tile Contours . . . 38
- 4.3. Dataflow . . . 39
- 4.4. Tile Creation . . . 41
  - 4.4.1. Transformation . . . 42
  - 4.4.2. Mosaicing Algorithm . . . 42
  - 4.4.3. Tile Deskewing . . . 43
  - 4.4.4. Error Handling . . . 45

4.4.5. Discussion of the Tile Creation Process . . . 45

5. Object Classification and Stem Localization 47
5.1. Image Preprocessing . . . 47
5.1.1. Optimization for Channel Subtraction . . . 47
5.2. Segmentation . . . 48
5.2.1. Segmentation Algorithm . . . 49
5.2.2. Evaluation of Image Subtraction and Segmentation . . . 49
5.2.3. Contour Overlapping Detection . . . 50
5.3. Extraction of Feature Values . . . 51
5.3.1. Feature Extraction . . . 51
5.3.2. Evaluation of Feature Values . . . 51
5.4. Object Classification . . . 52
5.4.1. Choice of Classes . . . 53
5.4.2. Classifier Requirements . . . 53
5.4.3. Evaluation of Classifier Types . . . 54
5.4.4. Realization . . . 55
5.5. Stem Localization . . . 57

6. Implementation 59
6.1. Middleware . . . 59
6.1.1. Code Structure . . . 59
6.1.2. Advantages . . . 59
6.2. Image Processing . . . 60
6.3. Graphical User Interface . . . 61
6.4. Thread Handling . . . 61

7. Evaluation and Analysis 63
7.1. Tile Approach . . . 63
7.1.1. Accuracy of Marker Placement . . . 63
7.1.2. Processing Efficiency . . . 65
7.2. Evaluation of Shared Autonomy Approaches . . . 67
7.2.1. User Study Setup . . . 67
7.2.2. Weed Detection Performance . . . 70
7.2.3. User Experience . . . 72
7.3. Impact of the Shared Autonomy Approach . . . 73

8. Summary and Future Work 77
8.1. Summary . . . 77
8.2. Conclusions . . . 79
8.3. Design Recommendations and Future Work . . . 79

Appendix 83
A. Feature Values . . . 83
B. Questionnaire for the User Study . . . 85

Bibliography 89

List of Figures 95

List of Tables 97

Nomenclature 99

Abstract

This thesis examines a Shared Autonomy approach for an image based weed detection framework in agricultural robotics. A key obstacle of the realization of commercially viable robots is that state-of-the-art autonomous systems are not sufficiently robust in unstructured environments. Our hypothesis is that a human user in the loop is able to support an automated process in difficult tasks and overcome these obstacles. This thesis examines how user input can be queried, integrated and effectively incorporated.

We first evaluate possible points of insertion for the human user and different integration concepts. Based on out-of-the-box solutions, we design a flexible, user-centered weed detection framework, which involves the human in the classification of plant contours. The user supports the autonomous system in its task to detect and eliminated weed through mobile manipulation. An image stream created out of a series of overlapping images received from the robot is employed for user interaction and the association of single images with the goal to optimize the cooperation between robot and user.

We evaluate different interaction scenarios between the human user and the classifier with varying levels of autonomy through an user study considering both the system performance and the user experience. We observe that it is difficult for a shared autonomy system with a high level of autonomy to compete against less sophisticated human-in-the-loop approaches due to the complexity of the weed detection process. Our results show that the user and the classifier influence each other in a symbiotic manner and that our tile image stream approach reduces the load on the user as well as the amount of data which has to be transmitted within the system significantly. Based upon our experiences we derive a set of design suggestions for future systems.

Keywords and Phrases

Shared Autonomy, Mixed Initiative Control, Human-Robot Cooperation, Human-Robot Interaction, Human-Robot Interface, Adjustable Autonomy, Precision Agriculture, Precision Farming, Weed Detection, Weed Identification, Active Weed Control, Shape-based Features, Machine Vision, Machine Learning, Classification, Query Strategy, Support Vector Machine, User Study, ROS, Robot Operating System, BoniRob

ACKNOWLEDGEMENTS

First, I would like to thank my reviewer Professor Stiller, who established the contact to the Bosch Research and Technology Center Palo Alto and supervised my thesis. Next, a very big thank you goes to Sarah Osentoski who advised me with this thesis and spend a large amount of time on our meetings, all my questions and the review of this work. It was always a great inspiration working with you!

I am incredibly grateful for the opportunity to spend some time in such an amazing work environment. I would like to thank Benjamin Pitzer, Philip Roan and Matthias Roland for sharing their knowledge with me and being available whenever I needed some advice. Dejan Pangercic and Kai Franke, I will not only miss your expertise, but also our late-night Antenne-Bayern work sessions and jogging with you in the hills of Arastradero Preserve.

Furthermore I would like to send some greetings to my intern colleagues in the robotics team: Nikolaus Demmel, Fadri Furrer, Karol Hausman, Ross Kidson, Johannes Kühn, Russell Toris and Thomas Witzig. It was a great inspiration and a lot of fun working with all of you—keep up the great work!

Finally, I would like to thank my parents, who support me in all of my carrier decisions. I know this is not always easy for you.

1. Introduction

A goal of autonomous robotic systems is to execute tedious or dangerous tasks, increase productivity, decrease production costs and preserve the environment. However, state-of-the-art systems are only able to solve a limited number of tasks in clearly defined environments robustly. These robots cannot handle complex tasks or unexpected events reliably and have problems adjusting to changes in an unstructured environment.

Robots are skilled at data acquisition, data storage, navigation and manipulation. They are able to execute tedious work which would bore a human quickly with a steady performance. On the other hand, many tasks that are highly challenging for robots such as perception, situation awareness and intelligent decision making, can be easily handled by humans. Therefore, it seems to be a very promising approach to combine both the skills of a robot and that of a human in order to overcome restrictions which arise from a system design that purely focuses on autonomy. The combination of human and machine is able to incorporate the strengths of both sides and therefore increase the robustness of a system, decrease the development costs and offer the possibility that one can learn from the other. This idea is often referred to as *Shared Autonomy*, *Human-Robot Cooperation*, *Adjustable Autonomy*, *Human in the Loop* or *Mixed Initiative Control*. We use the term shared autonomy in this thesis. As human input is cost-intensive, one optimization goal is to maximize the level of autonomy. Another goal is to obtain a system which is still reliable and robust. Furthermore, it is important to create an interface between human and robot which allows an efficient cooperation. This motivates the design of a system which focuses on an intelligent and user-customized communication between human and robot.

This thesis aims to improve the functionality and reliability of an agricultural robot by using feedback provided by a human. We describe the design of a shared autonomy system for the discrimination between crop and weed plants. The main goal of our work is to evaluate human-machine interaction scenarios in order to understand the effects of a human user in the loop, to identify suitable points of application for a human in an autonomous object detection process and to evaluate different interfaces enabling the human-robot interaction. Therefore, we design an integrated framework that is built with the shared autonomy approach in mind from the very first step.

The focus of this thesis is the optimization of the collaboration between the autonomous system and the user. As we are interested in an extensive evaluation of different interaction scenarios rather than the optimization of the overall system performance we employ out-of-the-box solutions for the autonomous processing pipeline whenever possible.

1.1. Problem Definition

The German organic farming industry is a growing sector. In 2012, 3.9 % of the food sales in Germany were organic products, this is an increase of 6 % compared to the previous year ([1] p. 16). The high request for organic products marked with the German "Bio" certificate results in the demand to establish more efficient production technologies, which increase the productivity and decrease product prices, making organic products affordable for everyone. Similar developments can be observed in other countries. One of the main challenges of farming organic field crops is the prohibition of synthetic herbicides regulating weed plants growing on the fields. However, it is especially important to ensure that young crop plants do not have to compete with weeds and to give these plants a growth advantage in the first month of their growth period. Therefore alternative weed control techniques have to be applied. Figure 1.1 shows the state-of-the-art solution for weed regulation on organic carrot fields in Germany. Workers lying flat on a trailer are dragged over the weed dams by a tractor and remove unwanted plants either with their bare hands or with simple tools. Working conditions on organic farms are often criticized[1] and do not always meet the expectations associated with a fully sustainable production. The weed regulation in organic carrot fields is a highly interesting domain of application because of the tedious and cost-intensive state-of-the-art weed control technique and because of the high share of area under cultivation for carrots compared to other organicly grown vegetables in Germany ([1], p. 8).

Figure 1.1.: State-of-the-art weed regulation technique for organic carrot fields in Germany

Our project takes place in the context of the motivation to create a robot that is able to automatically pick weeds with a high level of autonomy. Previous projects on agricultural robotics include the development of BoniRob [2], a robot employed for plant rating. While the main focus in the first BoniRob project was robot control, localization, navigation and perception, the goal of the follow up project Bonirob2 is to introduce an additional manipulation functionality that allows active weed control. This technique aims at the specific manipulation of weed plants only and therefore minimizes unwanted side effects such as crop damage in contrast to passive weed control. BoniRob2 is designed with an app-like concept that allows the attachment of different hardware 'apps' [3]. One app is built for weed manipulation and contains a delta kinematic manipulator which enables the robot either to punch or to grab and uproot weeds (see Figure 1.2).

[1] `http://grist.org/article/mark`

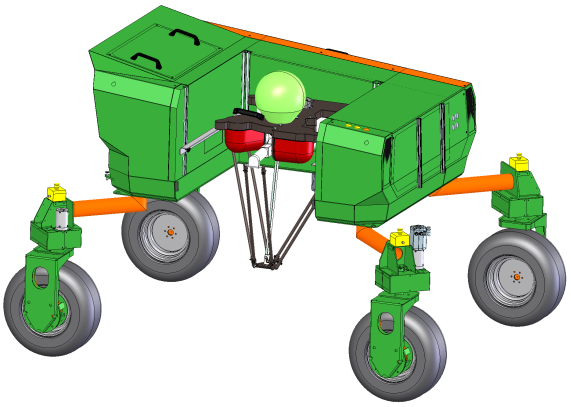


Figure 1.2.: Bonirob2 with delta kinematic for weed manipulation

Active weed control is a challenging problem touching the topics perception, cognition, manipulator control and hardware design. This thesis regards two parts of this process, perception and cognition, separated from the overall system in order to evaluate the application of shared autonomy concepts. The goal is to detect the stem positions of weed in an overlapping stream of camera images without depth information captured while the robot is moving linearly along the weed dams. We define the weed stem position as the point in the image, where the stem of a weed plant intersects with the approximated soil plane. Per detected weed plant, one position marker and an image of its surroundings are sent back to the robot for weed manipulation by Visual Servoing [4].

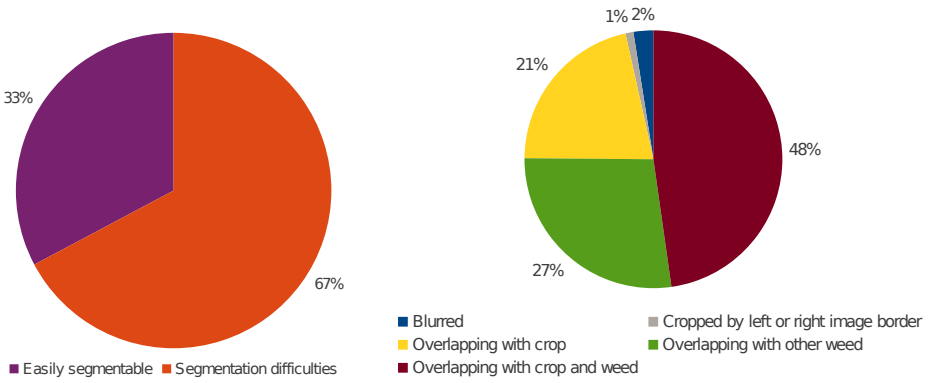
Challenges in the weed detection process

One of the main challenges is reliable and highly functional plant discrimination. Related research such as that of Åstrand and Baerveldt [5] and Weis and Gerhards [6] show that correct detection rates for completely autonomous weed detection range from 77 % to 98 % in the field. However, the latter value could only be achieved using a high a priori knowledge of expected weed plant species and not considering temporal and environmental variabilities.

It is important to identify the main challenges in the weed detection process in order to create a system which is designed to overcome these difficulties. There are three major problems:

Segmentation difficulties A system for weed detection must be able to distinguish between plants on the field in order to determine the plant type and the plant's stem position. This is difficult because of *occlusion*, the overlapping of multiple plants in regions with a high plant density. A statistical investigation by visual observation based on 298 images of one of our datasets shows, that 65 % of weed plants are overlapping with other plants (Figure 1.3). As no depth information is available, these overlapping contours will cause undersegmentation. Åstrand and Baerveldt [5] sum this problem up in their conlcusion: "Increasing weed pressure also brings on more overlapping plants that lower the performance of the system, especially when weed is merged with crops."

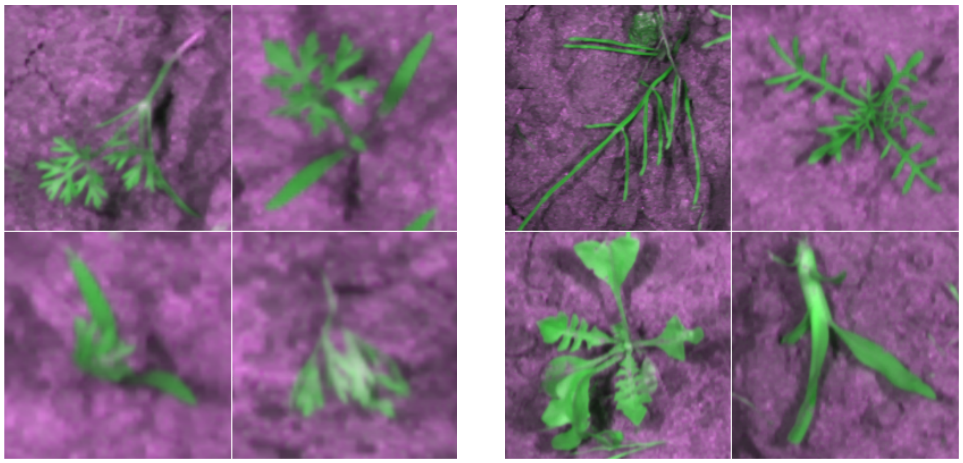
Furthermore, fine plant structures such as stems are hard for cameras to resolve and can be the cause of oversegmentation. These segmentation problems demand higher requirements regarding sensing techniques and image segmentation algorithms for an autonomous system. Additionally, motion blur and imperfect lighting conditions decrease the image quality.



(a) Overview (b) Reasons for segmentation difficulties

Figure 1.3.: Evaluation of difficulties in the weed segmentation process based on visual observation

Variable plant appearance There is no a priori knowledge about expected types of weed plants on the field. Moreover, the outer appearance of plants is highly variable over their growth stage and influenced by environmental conditions such as winds. Figure 1.4 provides an overview of crop and weed plants encountered in our dataset and the variability of their appearance. A good system has to be flexible enough to recognize formerly unknown weed types and adapt to changes of the plant appearance. Slaughter et al. state in their review on shape-based plant classification [7]: "While a large number of shape based methods for machine vision recognition of plants have demonstrated good potential under ideal conditions, a lack of robust methods for resolving occlusion, leaf damage or other visual 'defects' (e.g., insect or hail damage, leaves twisted in the wind, or splashed with soil) commonly found in farms remains a major challenge to commercialization of the technique."



(a) Crop plants (b) Different types of weed plants

Figure 1.4.: Examples for plants encountered in our dataset. The soil background is purple because the images are acquired with a multispectral camera. The red channel of the camera is mapped to the red and blue channel of the image whereas a second Near Infrared channel is mapped to the green channel.

Difficulties localizing the stem of a plant Our use case of active weed manipulation by gripping or punching results in high demands regarding the localization of the stem of a weed plant. This is the point of application for the manipulation tool. Figure 1.4b

shows that the stem position of a weed plant varies depending on the plant type and pose. To illustrate this we describe the difference between prostrate and upright weed. The prostrate weed grows flat on the soil surface, so the stem is close to the border of its shape, whereas upright weed has its stem position in the center. The proper detection of the stem position requires a distinct knowledge about the plant pose and structure.

In their review on autonomous robotic weed control systems [7] Slaughter et al. conclude: "Robust weed detection and identification, remains as the primary obstacle toward commercial development and industry acceptance of robotic weed control technology."

1.2. Concept Overview

Due to this challenges, there is no industrial system for active weed regulation available until today which is based on the detection of crop as well as weed plants. Current research focuses on the improvement of sensing techniques [8] and better weed detection algorithms [9]. This increases the hardware and development costs of a product as well as the complexity of the overall system. Often, the effort to increase the robustness of a system is disproportionate to the amount of autonomy gained additionally (compare e.g. Bohren et al. [10]).

Other fields of applications where robots operate in complex environments, for example explore a cluttered area (Fong et al. [11]), show a different approach to create reliable solutions. Robots are provided with the ability to interact with a human user in the case of uncertainties. This idea is known as shared autonomy. It bridges the gap between full teleoperation and a completely autonomous system. Shared autonomy enables the realization of industrial solutions for robots operating in unstructured environments[2] and can be seen as an intermediate step in the direction of fully autonomous systems.

We create a shared autonomy system in order to overcome difficulties in the weed detection process. It is expected that this approach increases the system functionality, reliability and robustness. The goal of this thesis is to design and evaluate an user-centered shared autonomy approach for weed discrimination in carrot fields. We aim to create an integrated framework which naturally allows the users to interact. Our focus is to determine different scenarios where the user can help the system, to evaluate several scenarios for human-robot interaction in an user study and to examine, how the level of autonomy affects user and pipeline performance.

The system is evaluated on a preliminary test dataset of 2D-field images captured by a multispectral camera. This dataset was acquired on an organic carrot farm in Germany with a camera mounted on a small vehicle representing the weed manipulation robot. Due to the advanced growth stage and the resulting high occlusion of carrot and weed plants the dataset can be considered as difficult and cannot be compared to datasets in previous work such as that of Åstrand and Baerveldt [5].

One of the main challenges of our approach is the interaction with the human user. The basic concept is illustrated in Figure 1.5. The weed detection pipeline receives images from the robot, processes them with the option to request additional user feedback and finally returns the position of weed stems and an image of their surrounding for Visual Servoing.

1.2.1. Weed Detection Pipeline

We describe an overview of the autonomous portion of our proposed system. Our design approach is presented in Figure 1.6. In a first step, published images are received and

[2] `http://goo.gl/0YBmV`

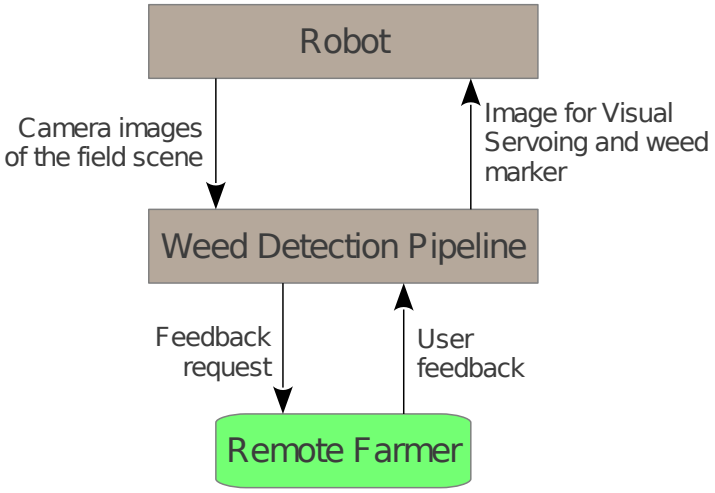


Figure 1.5.: Overview of the system design with the shared autonomy approach

preprocessed. Next, the images are segmented into contours of plants. We use plant contours, as they can not only be employed in the classification step, but also provide an excellent basis for the pipeline-user interaction as a field image overlayed with the detected plant contours is a good visualization of the segmentation results. After the segmentation step, feature values for each plant are extracted out of the detected shapes and labels assigned by the classifier. For all weed plant objects, a stem marker describing the stem location is determined and per marker, one cropped image containing its surrounding environment is created for Visual Servoing. Both the marker and this image are sent back to the robot.

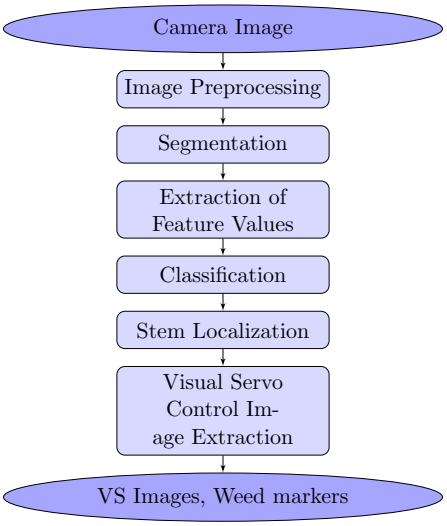


Figure 1.6.: Autonomous weed detection process

1.2.2. User Integration

Based on this pipeline architecture, we are able to integrate the user, also called *Remote Farmer*, in our detection process. Possible points of application and interaction concepts are presented and evaluated in Chapter 3.

We develop an user interface based on a scrolling image visualizing a cut-out of the current ridge of the field in order to increase the situation awareness of the human user. Therefore,

we employ the overlapping characteristic of the images received from the robot in order to create an image mosaic. This mosaic consists of non-overlapping images, the so-called *tiles*, which can be assembled at their image border to create an image stream. The graphical user interface displays the image stream and interaction requests to the user. The stream does not only improve the user performance and experience as it provides a high scene context, but is also important to create associations between the overlapping images received from the robot and to reduce the amount of data which has to be transmitted between GUI and processing pipeline.

1.3. Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2 reviews state-of-the-art approaches for human-robot interaction, provides an overview of precision agriculture developments and presents research projects for autonomous weed control.

Chapter 3 introduces the proposed shared autonomy system for weed detection. Furthermore, we describe the user interface developed in the scope of our thesis.

Chapter 4 presents the tile approach employed to create an image stream and examines the dataflow in the shared autonomy pipeline.

Chapter 5 describes the autonomous part of the processing pipeline which enables the shared autonomy system.

Chapter 6 provides implementation details of our system.

Chapter 7 evaluates and analyzes the overall system performance.

Chapter 8 contains on overview of the thesis, our conclusions and design recommendations for further development.

1.4. Project Acknowledgements

This work is carried out as part of a public funded project by BMELV (German Federal Ministry of Food, Agriculture and Consumer Protection). The project partners are: AMAZONE, Robert Bosch GmbH and the Hochschule Osnabrück.

2. Background

In this chapter, we provide an overview of previous work. Our approach to evaluate different human-robot interaction concepts for the detection of weed plants is related to several veins of research. Therefore, this chapter is divided up into three parts: At first, we examine existing shared autonomy literature. Although our work touches only a partial system of an agricultural robot, we want to provide a quick overview of modern agricultural techniques summed up under the term Precision Agriculture. Finally methods and the state of the art for autonomous weed detection systems are described. As the focus of our thesis is how a human and a robot can achieve a common goal together and not the implementation of a custom-made fully autonomous system, image processing and object recognition techniques are not included in this chapter and will be introduced whenever required.

2.1. Shared Autonomy

Conventional robotic design approaches focus either on the development of fully autonomous systems or robots which can be teleoperated from a human user. Both of these approaches are challenging. Fully autonomous systems often require restrictive assumptions about the environment in order to operate. Teleoperation systems can be cumbersome due to the need to control a high number of degrees of freedom, the requirement that large amounts of data must be transmitted in a short time, and the high level of human involvement. While teleoperation is acceptable for robots that work for example in environments which are dangerous for human users, it is not desirable for the goal of robots reducing workload and increasing overall efficiency.

Shared autonomy approaches try to bridge the gap between the two described concepts by combining the strengths of a robot such as data acquisition and storage, manipulator control, localization and planning with that of a human for example perception, reasoning and context awareness (compare Pitzer et al. [12]). There are two interfering goals:

- Reduce the user input and its complexity to a minimum.
- Maximize the overall system performance, robustness and reliability.

We define Shared Autonomy as the involvement of a human user in an otherwise autonomous process.

2.1.1. Interfaces for Human-robot Interaction

One important research question in the field of shared autonomy systems is the efficient interaction between robot and user. This depends in large parts on the design of suitable interfaces for collaboration. In their survey on human-robot interaction, Goodrich and Schultz [13] define the Human Robot Interaction (HRI) problem as "to understand and shape the interactions between one or more humans and one or more robots". They name five attributes which a designer can affect:

Level and behavior of autonomy There are numerous definitions for the level of autonomy. A straightforward one is "the amount of time that the robot can be neglected" [14]. We will examine the level of autonomy further in one of the following subsections.

Nature of information exchange Defines the manner how information is transmitted between human and robot. Goodrich and Schultz distinguish between the medium and the format of information exchange. Whereas there is a wide variety of formats, the following media are commonly employed for HRI tasks and can also be combined:

- Visual displays such as that of Leeper et al. [15] for the evaluation of shared autonomy grasping approaches.
- Gestures such as the interface of Marge et al. [16] which uses operator following and gestures for the control of an unmanned ground vehicle.
- Speech and natural language, one example in the field of robotics is the work of Tellex et al. [17] who examine natural language commands for robotic navigation and manipulation systems.
- Non-speech audio e.g. audio signals alerting an user about critical system conditions as employed by Dixon et al. [18] in the field of unmanned aerial vehicle flight control.
- Physical interaction and haptics such as the force feedback system of Chotiprayanakul et al. [19] who use a force field for collision avoidance in a teleoperation scenario for robotic grit blasting.

Furthermore, a combination of different interaction modes is also possible. One example is the multimodal interface of Ubeda et al. [20] which combines haptic feedback, a graphical interface and electrooculography for the control of a robot arm.

In their survey, Goodrich and Schultz cite four different metrics for the measurement of interaction effectiveness: The *interaction time*, the *cognitive* or *mental workload* of an interaction, the amount of *situation awareness* produced by the interaction and the *amount of shared understanding of common ground between humans and robots.*

Structure of the team The structure of the team describes the amount of human users and robots involved in the HRI task as well as their authorities and roles.

Adaption, learning, and training of people and the robot Both the user and the robot have to adapt to variability in the interaction scenario. They can learn from the other and might require training in order to optimize the overall system performance.

Shape of the task "Task-shaping is a term that emphasizes the importance of considering how the task should be done and will be done when new technology is introduced" [13].

Our thesis is interested in all of these aspects, but we put a special focus on the variation of the level autonomy as one important question is to what extend an autonomous system can enable a human user in the weed detection process. Furthermore, our work examines how to best display information from the system to a human user so that he can intuitively help the robot.

Goodfellow et al. [21] show, that it is not necessary to have one perfect implementation of an interface for information exchange, but rather different task-oriented interface types which enable to overcome restrictions in the communication between user and robot. One example is a robot which has to pick a distinct bottle. It is easier to send images of possible bottles to the user and ask him for the correct one than trying to determine the available bottle types and contents and translate this information into natural language. Goodfellow et al. describe the design and implementation of three different mobile user interfaces for the domains navigation, perception, learning and manipulation.

2.1.2. Level and Behavior of Autonomy

One important research question is to find an optimal level of autonomy. In their research about a model for types and levels of human interaction with automation [22], Parasuraman, Sheridan et al. start with the consideration of a task that is manually executed by a human. They create a framework which allows to evaluate which parts of this task can be automated by what extent. Based on a four-stage model for human information processing they define ten levels of automation of decision and action selection as displayed in Figure 2.1. They underline that "automation does not simply supplant human activity but rather changes it, often in ways unintended and unanticipated by the designers of automation" and demand "human-centered automation" [23].

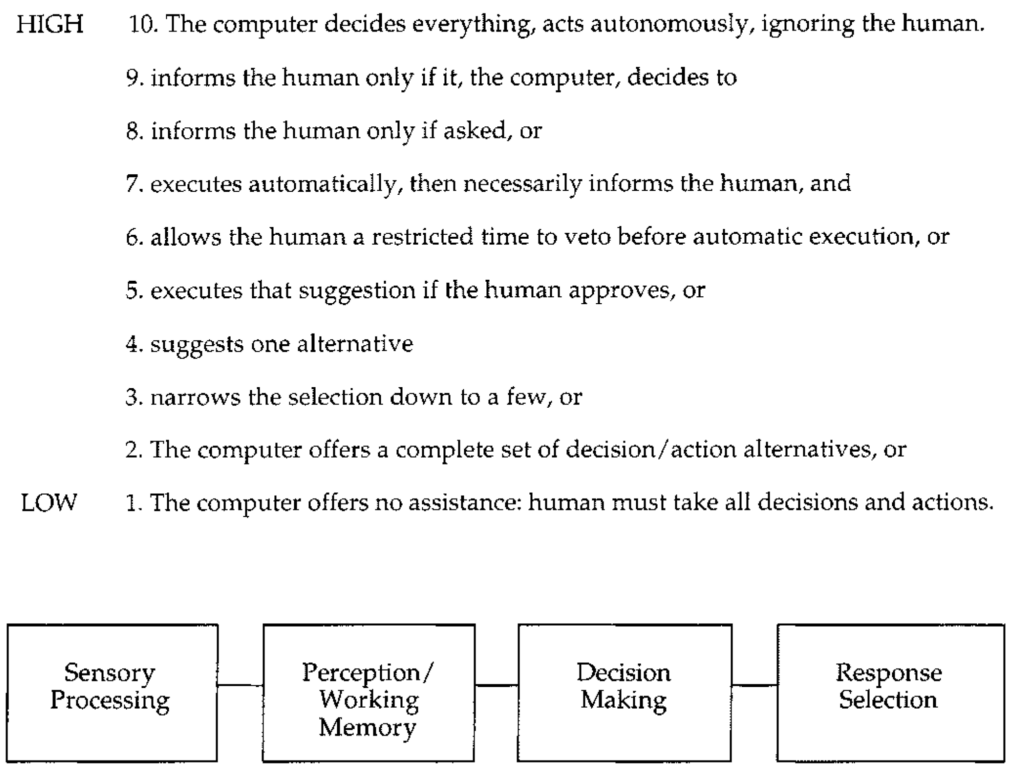


Figure 2.1.: Levels of automation and four-stage model of human information processing as defined by Parasuraman, Sheridan et al. [22]

The term *Adjustable Autonomy* is often utilized to underline the variable ratio between the amount of user input and the level of autonomy of a robot. One important research

question is to figure out the effectiveness of a human-robot team depending on this ratio. An effectiveness measure can relate the amount of user input to the system performance. In general, it is desirable to minimize the user feedback and maximize the robot's performance. Kaupp and Makarenko [24] define the level of autonomy of a robot as the cost value for user feedback. Their robot fulfills a navigation task and requests user feedback, when the expected information-gain justifies the costs. The *team effectiveness* is measured as a function of the autonomy level. It is a weighted sum of the success rate, the task fulfillment time and the required user queries. Kaupp and Makarenko show that the optimal choice of the level of autonomy of a robot highly depends on the metric employed to measure the team effectiveness. For three different scenarios, the optimal level of autonomy varies between fully autonomous operation and continuously queried user feedback. All of the scenarios achieve relatively good results for a high level of autonomy combined with a small amount of user input.

In general, it is important to determine where to incorporate user input. It can be distinguished between low- and high-level tasks. Low-level tasks are often easy to solve for a human and therefore have the potential to be outsourced, for example as a crowdsourcing application. A classical use case is the solution of captchas. Montoyama et al. [25] state, that a human-provided solution for one captcha costs around 0.001 $ and has a median response time between 9 and 22 seconds. This information enables us to estimate the value of human user input for simple, not time critical tasks. Our thesis aims at integrating the user at such a low-level task as well—critical processing steps in the weed detection pipeline are solved with additional human input. High-level tasks are commonly employed in human-robot interaction scenarios and include beneath others robot surveillance, navigation and failure recovery (Sankaran et al. [26]).

2.1.3. Nature of Information Exchange

As we are examining an image based weed detection approach and are interested to create an user interface which is compatible with standard hardware, we will employ a visual display as media of information exchange. In this subsection, a quick overview of state-of-the-art visual interfaces for robots is provided.

Chen et al. [27] review research papers which examine the efficiency of user interface designs for robot teleoperation. They name the main influencing factors which decrease the human performance in teleoperation tasks: A limited field of view, limited depth information and possible bandwidth restrictions. "As a result, the operator's situation awareness of the remote environment can be compromised and the mission effectiveness can suffer." They underline that one of the main design goals for an interface is to maximize the situation awareness of the user.

It is important considering which data has to be visualized in an user interface. Therefore they divide the information for teleoperation control stations up into five categories:

- Sensor view and/or data transmitted from the robots
- Plans and commands issued to the robots
- Health status of the robots
- Status of the tasks
- Map displays

Although our use case is different, this list provides a good example for the variety of data which has to be considered in the GUI design process.

Keskinpala et al. [28] share their experiences on the development of a GUI for the teleoperation of a mobile robot. They point out it is important to provide "rapid but meaningful information". Especially when the GUI is designed for inexperienced users robot internal data has to be translated into a format that "provide[s] insight into a robot's environmental view". Often, images are a good base for the visualization of information, as they resemble the human visual system and contain "a large amount of information that is not easily obtained with other sensory modalities". These images can than be enhanced in the fashion of an Augmented Reality approach with the overlay of additional sensor data, for example that of a laser range-finder. Keskinpala et al. expect this overlay method induces higher workload levels.

A high amount of interface research for human-robot interaction is carried out in the scope of Urban Search and Rescue (USAR) applications. One example for this is the work of Baker et al. [29]. They provide guidelines, which focus on the development of USAR teleoperation interfaces but are also highly relevant for our thesis:

Enhance awareness Provide a map indicating where the robot has been. Provide more spatial information about the robot in the environment to make operators more aware of their robots' immediate surroundings.

Lower cognitive load Provide fused sensor information rather than make the user mentally combine data from multiple sources.

Increase efficiency Minimize the use of multiple windows, and provide user interfaces that support multiple robots in a single window, if possible.

Provide help in choosing robot modality Provide the operator assistance in determining the most appropriate level of robotic autonomy at any given time.

Baker et al. [29] demonstrate their design guidelines by the improvement of an existing interface displayed in Figure 2.2a. They observed that the users focused mainly on the video display in the interface. Thus, this is the new central element in their GUI. Furthermore, they visualized information more effectively and got rid of "dead space" in the main view. The resulting interface is shown in Figure 2.2b.

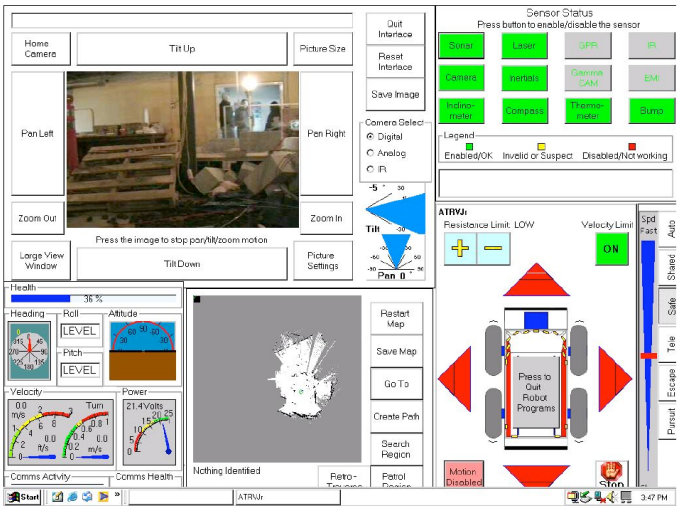
2.1.4. Shared Autonomy Applications

There are a wide range of applications for shared autonomy approaches in robotics. In the following part, we will present some selected example applications that are especially interesting.

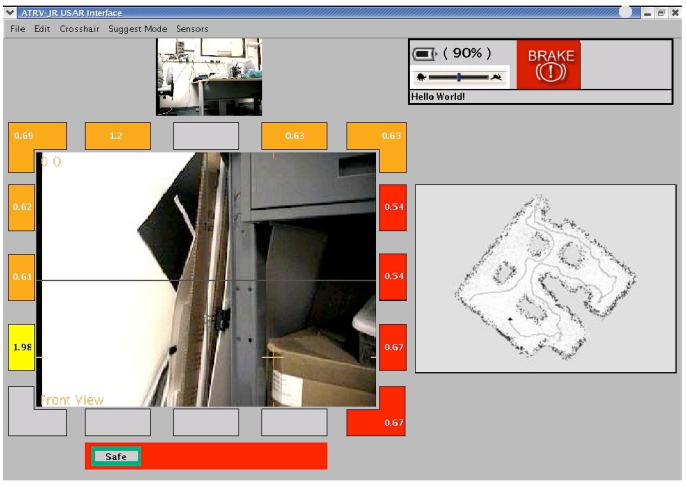
Object Manipulation

Pitzer et al. [12] employ the skills of a human user in order to assist a robot with a complex perception task: To segment an object which might be partially occluded correctly. They are able to show that significant robustness improvements are achieved based a shared autonomy approach. Pitzer et al. point out, "that a human-robot team can work together effectively solving a typical object manipulation task."

In an user study on human-in-the-loop robotic grasping [15], Leeper et al. evaluate four different user interfaces with varying levels of autonomy in different environments. Inexperienced users had the task to grasp as many objects as possible while avoiding collisions. Leeper et al. show, that interfaces with a higher level of autonomy perform better compared to interfaces which allow the user to control the robot arm on a lower level of control. However, the users tended to trust high level autonomous systems too much. Therefore,



(a) Old existing interface



(b) Design of the new interface

Figure 2.2.: Redesign of an user interface for an USAR application [29]

one important conclusion of their work is that "autonomous components must establish an appropriate level of trust with the operator in order to provide significant benefits, and communicate their limitations in an appropriate way".

In a related study, Witzig et al. [30] collected context information regarding an object grasping task from a human user with the aim to improve the ranking of autonomously created grasp suggestions based on the decision of a Bayesian Network. They demonstrate, that their approach has the highest grasping success rates and produces less collisions compared to other concepts with higher and lower levels of autonomy. On the other hand, their user study shows tendencies that the participants preferred methods with a lower level of autonomy which provides them with more control over the robot even if those methods produce worse quantitatively measured results.

Mobile robotics

A good example for a high level shared autonomy task is the telepresence bot QB of the company anybots[1]. While the user navigates the robot with keyboard commands, the platform supports its supervisor in his navigation task and avoids obstacles.

[1] `https://www.anybots.com`

The application of shared autonomy integrating the user into high level tasks enables new scenarios in mobile robotics such as one user controlling several autonomous robots at the same time. Fong et al. [31] introduce the system model of *collaborative control*, where human and robot are considered to be active partners and communicate dialogue-based. The human user is not only able to send high level commands to mobile robots, but can also ask the robot simple questions regarding its task and system state. On the other hand, the robot has the functionality to ask the user questions—"human and robot collaborate in order to compensate for limitations of autonomy". This architecture makes the interaction between human and robot focus on the most important information and therefore increases the efficiency.

Urban search and rescue

Typical Urban Search and Rescue scenarios involve a small robot, which is able to explore areas of collapsed buildings that are inaccessible or too dangerous for human members of a rescue team. Optionally, the robot is able to transport emergency utilities to trapped persons or manipulate its environment. The employments of robots for USAR tasks gained increasing publicity after the collapse of the World Trade Center and has high demands as the robots execute possibly life-saving measures and are operated directly from end-users. Murphy [32] provides a good overview of human-robot interaction in rescue robotics. He states, that shared autonomy approaches in USAR mainly focus on the reduction of the human to robot ratio and assisted failure recovery.

2.2. Precision Agriculture

On one hand, a growing world population, the employment of biomass as substitute for other resources and increasing food prices boost the demand for agricultural production in larger scales and with a higher efficiency. On the other hand, farming methods are expected to have less environmental impact and sometimes also to obey extended restrictions regarding the employment of fertilizers and herbicides. Precision Agriculture (PA) tries to moderate between these two demands by the employment of methods enabled by advances in information technology.

Autonomous systems for active weed control are a typical Precision Agriculture application. Pierce and Nowak define Precision Agriculture as "the application of technologies and principles to manage spatial and temporal variability associated with all aspects of agricultural production for the purpose of improving crop performance and environmental quality" [33].

2.2.1. Methods

In their worldwide overview of precision agriculture, Zhang et al. [34] distinguish between six different variabilities:

Yield variability Historical and present yield distributions.

Field variability The topology of the field, for example elevation and slope.

Soil variability Geometrical, chemical and physical characteristics of the soil, e.g. the soil fertility or its water-holding capacity.

Crop variability Properties of the crops such as their height, density or stress for water and nutrient matter.

Variability in anomalous factors Different influences which lower the crop performance such as weed occurrence, insect infestation and wind damage.

Management variability

In general, there are two methods to manage the named variabilities [34]: The map-based approach: Data or samples are collected at distinct locations and then employed for the offline creation of a site-specific map which is used for the final application. The sensor-based approach: Application decisions are performed online based on synchronously captured sensor data.

Whereas studies mainly focused on the management of soil fertility in former years, advanced sensing, computation and guidance technologies enable the application of PA technologies in new fields such as weed manipulation or plant phenotyping. Our concept of an autonomous system for the active manipulation of weed plants aims at the management of the variability of one anomalous factor: Weed infestation. The attempt to distinguish between weeds and plants is strongly influenced by crop variability.

2.2.2. Trends

A recent trend in precision agriculture which is currently also present in modern media is the employment of unmanned aerial vehicles (UAVs), also known as "farming drones" (see for example Figure 2.3). In their review on the application of UAVs for precision agriculture, Zhang and Kovacs name the main use cases: "Yield mapping, chemical content measurement, vigor mapping, vegetation stress monitoring and assessment of impacts of fertilizing on crop growth." The advantage of UAVs for those monitoring tasks are before all relatively low costs and the flexible availability. However, Yhang and Kovacs list also the shortcomes such as high initial costs, low reliability and sensor capability, strict aviation regulations and problems with the acceptance by farmers.

Figure 2.3.: Aeryon scout quadrocopter for image acquisition

2.3. Autonomous Weed Control

Robots for autonomous weed control do not only enable a higher productivity, but also reduce the amount of required herbicides. The technique is also highly interesting for organic farming, where current weed control methods rely on cost- and labour intensive human work. It is distinguished between inter-row and intra-row weeding. The latter one is more difficult because crop damage has to be avoided, but also more relevant as inter-row weeds can be managed by passive control techniques such as chemical or mechanical solutions.

2.3.1. State of the Art

Slaughter et al. [7] provide an overview of autonomous weed control systems. Four core technologies are identified:

- Guidance
- Weed detection and identification
- Precision in-row weed control
- Mapping

Figure 2.4 provides an overview of a choice of autonomous vehicles employed for scientific studies of active weed control techniques or related research projects. All vehicle are equipped with row detection algorithms for row navigation and are, except of the platform in Figure 2.4b, designed with a flexible four wheel steering concept .

(a) BoniRob1

(b) Mobile robot of Åstrand and Baerveldt

(c) Platform of the Danish Institute of Agricultural Sciences

(d) Platform of Bakker et al.

Figure 2.4.: Vehicles for autonomous weed control and related precision agriculture research applications

BoniRob1 (Figure 2.4a, Rahe et al. [2]) is a mobile platform with adjustable track width created in the predecessor project. In contrast to its successor BoniRob2, it has no plant manipulation functionality. The main purpose of BoniRob1 is plant phenotyping. Therefore it is equipped with a variety of optical sensors for the measurement of plant morphology.

The mobile robot of Åstrand and Baerveldt [35] in Figure 2.4b is equipped with two cameras for row detection and a simple approach for crop identification, which resulted in

accuracies of up to 97 % in a test trial with sugar beets. However, the plant segmentation was manually supported. A height-adjustable wheel rotating around an axis parallel to the row-line is employed for intra-row weed control.

Figure 2.4c shows the platform of the Danish Institute of Agricultural Sciences. It does not contain any weed manipulation tools but is equipped with a real time kinematic global position system (RTK-GPS) for precise localization. The main focus of this research project is on robot control, guidance and the mapping of crop positions (Bak and Jakobsen [36]).

The platform of Bakker et al. [37] is built for the evaluation of intra-row weed detection and manipulation concepts. In their paper, they provide an excellent overview of the design process of the vehicle, possible design choices and their decisions. Currently, there is no information about field experiments for weed control is available.

To sum it up, all of the four presented concepts are similar, but focus on different disciplines. The system of Åstrand and Baerveldt is the only one with a functional weed control mechanism. Our shared autonomy approach for weed detection is the first one of its kind. Except for manual data preprocessing, no attempts to increase the functionality and robustness of plant detection algorithms by involving the user in the critical processing step have been made so far to the author's best knowledge.

2.3.2. Commercial Products

There is a manageable amount of commercial products for autonomous weed control. Existing solutions focus mainly on the combination of robust techniques out of autonomous weed detection systems with traditional agricultural methods. An overview of available technologies is provided in Figure 2.5.

(a) WeedSeeker

(b) WEEDit

(c) CLASS CAM PILOT

(d) Robocrop inrow

(e) Blue River vision

Figure 2.5.: Commercial products related to autonomous weed detection

One example is the *WeedSeeker*[2] (Figure 2.5a), a combination of a nozzle for herbicide application and a plant sensor based on spectroscopic techniques which use the different reflectance characteristics of plants and stones/soil. The system is able to distinguish between plants and ground without vegetation and sprays herbicides only when plants

[2] `http://www.ntechindustries.com/weedseeker-home.html`

are below the nozzle. This reduces the amount of required chemicals with the effect of decreasing costs and a lower environmental impact. The system is not able to distinguish between weed and crop plants, it is a pure plant detection technology.

A similar system on a larger scale is offered by *WEEDit*[3]. WEEDit sensors are mounted on a spraying vehicle (see Figure 2.5b) and include a red light source which illuminates a line on the field. They detect NIR light emitted from the plants as chlorophyl converts red light partially to NIR light. One sensor is able to control 5 nozzles with a distance of 200 mm. Array widths of up to 36 m cover can be realized by default.

The *CLAAS CAM PILOT*[4] (Figure 2.5c) is an example for the commercial availability of guidance systems. A 3D camera system navigates tractors beneath other through crop rows or ridges and therefore avoids crop damage and operator fatigue.

Probably the most advanced product for autonomous weed control is the *garford robocrop inrow*[5] shown in Figure 2.5d. This is a farming tool for inter- and intra-row weeding which can handle up to 6 m field width at once. One or several cameras detect the crop positions, which must be the dominant plants in the image. Sewing pattern are additionally employed for a robust detection. The intra-row weed control is executed by weeding discs rotating with variable angular speed around an axis orthogonal to the soil plane. The speed is controlled in such a way, so that the resulting cycloid curve of the tool on the soil plane does not damage crop plants but covers the soil region between the crops of one row. Additional fixed tines accomplish mechanical inter-row weeding. Although a human driver is still required for the system the active weed manipulations works autonomously.

A promising approach for fully autonomous systems for crop thinning as well as organic weed control is the Silicon Valley startup company *Blue River*[6] (see Figure 2.5e). Furthermore, the Danish startup *RoboWeed*[7] is working on an autonomous GPS-based system with an adjustable rotor weeding tool for active weed control [38].

2.3.3. Weed Detection and Identification

In their review on autonomous weed control systems, Slaughter et al. [7] conclude that the greatest remaining challenge is plant detection and identification. Approaches to solve this problem are divided into three groups based on the employed information: Biological morphology, texture and spectral characteristics. In general, most of the plant detection studies do not consider temporal variabilities over several growing seasons.

Biological morphology

The usage of morphological features such as the plant shape or also the shapes of single leaves are a widely spread approach with promising results. Yang et al. [39] provide a general overview of the enormous amount of shape feature extraction techniques and their specific characteristics. They underline in their conclusion, that the selection of shape features depends on the task.

Weis and Gerhards [6] name three different kind of features for the discrimination between weed species in multispectral images:

- Region-based features such as size, compactness and Hu moments [40]

[3] http://www.weedit.com.au
[4] http://www.claas.com/cl-pw/en/products/easy/on_field/optische_ls/cam_pilot
[5] http://www.garford.com/PDF/robocrop%20inrow%20en.pdf
[6] http://bluerivert.com
[7] http://www.venturecup.dk/competition/roboweed-winner-cleantech-and-technology

- Contour-based features such as fourier descriptors and curvature scale space representation [41]
- Skeleton-based features, which describe the plant structures

Weis and Gerhards make clear that it is important to reduce the dimensionality of the feature space to approximately 15 features with suitable methods. Especially skeleton and region-based features are selected by rating methods. They assign weed plants to four classes depending on their sensitivity to herbicides. A classifier evaluation shows, that different classifier types can all perform higher than 95 % correct classification rate and the choice of the classifier type is a minor influencing factor in the weed identification process.

Åstrand and Baerveldt [5] show, that not only geometric information about the plant itself but also sewing patterns of crops can be enabled for crop-weed discrimination. In field experiments, their algorithm is able to "identify 99 % of the crops and remove about half of the intra-row weeds".

Other sensing techniques employ ultrasonic sensors for plant height measurements with the goal to detect areas with a high weed density (Andújar et al. [42]) or a 3D LIDAR sensor which measures morphological information in combination with reflectance values (Weiss et al. [43]).

Texture

Texture features are rarely used for plant classification and most of the studies are executed under lab conditions. Burks et al. [44] are able to achieve an overall accuracy of around 90 % for the classification of five different weed types and soil with the Color Co-occurrence Method. Yet their experimental setup involves complicated artificial lighting conditions and the manual extraction of class samples.

Spectral characteristics

Spectral characteristics are widely utilized for the discrimination between soil and plant. As plants absorb contrary to soil and stones a high amount of light in the red band and have higher reflectance values in the other bands, multispectral camera images can be used for good plant segmentation results. Some possibilities are to compare normalized red- and green-values [5] or to work with more expensive camera hardware which is able to capture infrared images additionally and create a differential image between the infrared and red camera channel [45].

Furthermore, spectral reflectance is also used in order to distinguish between different plant types. Slaughter et al. [7] note, that this technique is more robust to partial occlusion and less computational expensive than shape-based methods. However, in his review on spectral properties of plants and their potential use for crop/weed discrimination in row-crops Zwiggelaar concludes, that "spectral information on its own is not sufficient for robust crop/weed discrimination, although in some specific cases it might give sufficient information. [...] The spectral information is optimal in the red and near-infrared regions of the spectrum" [46].

3. Shared Autonomy Approach

The weed detection process is a highly complex task. An autonomous system is able to handle parts of this tasks, however in order to ensure high detection accuracies, a robust detection process and to be able to react to failures in one of the process steps, we involve the user in the pipeline. Therefore, we introduce the shared autonomy approach in this chapter. At first, we suggest different concepts for integrating the user into the weed detection process. One promising strategy is the interaction between Remote Farmer and classifier. We examine different interaction scenarios. Finally, the realization of our pipeline-user interaction is presented and we describe the user interface.

We sum up the most important aspects considering the pipeline-user interaction and define the following items which have to be considered for the evaluation of our shared autonomy concepts:

Human insertion point The human insertion point is the processing step where the user is integrated into the system. For example, the user can be involved in the algorithm which detects the weed stems based on the contours of weed plants. It is necessary to identify promising insertion points. These are processing steps which are hard to solve autonomously and suitable for human-robot interaction. The performance of an autonomous algorithm applied on a task and the expected improvement by user integration have to be considered for the choice of the insertion point.

Query type Once an insertion point is found, different means of communication between robot and user must be evaluated. The main question is the design of the request for user input, the interface between robot and user. It is a complex challenge, as the requirements on both sides are very different.

Query behavior The query behavior defines, at which point and how often the user is asked for help. In general, we consider the user input as a highly valuable and cost-intensive source. Often, the reaction time of a system is increased drastically when requesting user feedback, as this input requires in general more time than the autonomous computation of a task. Therefore, it is important to develop concepts that allow an intelligent detection of required user input. The query behavior defines the level of autonomy of a robot.

Response usage In general, there are two different ways how a query response can be used:

- Utilize the user feedback directly for the instance of data which triggered the query. For example, the user might be presented a classification result for a detected plant, correct its label and send it back to the robot. The robot can now integrate this response into the dataset describing the type of this plant.
- Employ the user response in such a way, so that it affects the future behavior of the autonomous system. This approach is known as *learning* or also *active learning*, when combined with intelligent query behavior. To continue the example used above, the robot might add the features associated with the plant and the user-assigned class label to a training dataset for its plant classifier.

It is desirable to combine those two characteristics in order to maximize the outcome of user input.

User expertise It is important to consider the user suitability for a defined task. Does the user require any special skills or training or is the task so easy, that it could possibly even be outsourced in the fashion of a crowdsourcing application? This assessment also helps to estimate the expected quality of the user response. While a response with high precision and accuracy might be used as a ground truth overwriting autonomously computed results, a more sophisticated data fusion is required for lower response quality.

Impact on the overall system design In general, the design of a shared autonomy system is an iterative approach. It is not sufficient to design the autonomous system first without consideration of the user integration, identify then critical parts of the system and apply then a shared autonomy strategy for the solution of these parts. The problem of this procedure is the high aberration between algorithm and user requirements. While an algorithm is able to work on a very high abstraction layer with a sharply defined amount of information, the performance of a human user depends on the quality of the visualization of the data and the user might require context information for good response quality. Therefore, it is important to consider the user involvement from the very beginning of the system design. Yet the design focus should always be on the autonomous capabilities of the system in order to avoid an excessive employment of user feedback.

Furthermore, the different capabilities of humans and machines have to be considered. In general, it is not promising to employ the user as a simple replacement or additional information source for distinct autonomous processing tasks. For example, one of the most challenging tasks in the pipeline for the detection of weed stem positions is the proper segmentation of plants. The user could be asked to segment plant regions which cannot be handled by the segmentation algorithm. However, this is a challenging and time-consuming user task and the following processing steps such as classification and stem position extraction introduce new uncertainties. A better user integration is to ask the user to select the stem positions of weed plants for this regions directly which requires one click per weed plant. This decreases not only the processing time significantly but also eliminates uncertainties in the detection process as some processing steps can be skipped.

Side effects The integration of the user facilitates processing steps, but also yields new challenges which have to be considered:

- What is the reaction time of an user and is there a time limit marking the maximum allowed reaction time?
- What happens, if this maximum limit is exceeded?
- How is a quick and fail safe data transmission between user interface and robot ensured?

3.1. User Integration

Now that we have developed the main points of consideration for the design of a shared autonomy system, we must integrate the user into our approach for the weed detection pipeline. Our major goal is to identify and evaluate *human insertion points* and possible *query types*. In the end, we choose the most promising concept which will be realized in the scope of this thesis.

3.1.1. Human Insertion Points

Figure 1.6 gives an overview of the steps in the weed detection process. This design is already created with the shared autonomy approach in mind as the interfaces between the processing steps are clearly defined and therefore allow the insertion of additional dataflow. Furthermore, the employed concept of plant contours as abstract elements describing plant characteristics is a good basis for the visualization of processing results and for user interaction.

We identify three promising human insertion points. The human input has the potential to increase the quality of the processing steps *Segmentation*, *Classification* and *Stem Localization*. All of these insertion points have in common that a high user expertise is required, especially the classification step requires the Remote Farmer to be able to distinguish between different plant types. We compare the performance of inexperienced users with an expert user in Section 7.2.

Segmentation

The goal of the segmentation is the detection of the shapes describing the plant contours. It is the most important step in the processing pipeline as nearly all following steps such as feature value extraction, classification and stem localization depend on its results. We already showed in 1.1, that 65 % of weed plants are overlapping with other plants. As no depth information is available this occlusion will cause undersegmentation. On the other hand, fine plant structures such as stems may lead to oversegmentation.

The user input can be employed to resolve such segmentation errors by modifying the autonomous segmentation results or by providing ground truth information about image background and foreground data so that user-interactive segmentation algorithms such as *grabcut* [47] can be applied. In terms of response usage, the main purpose of this input is to correct distinct failures in the weed detection process directly. However, there are also approaches to improve the segmentation algorithm based on user input with Machine Learning techniques [48].

It is hard to establish a selective query behavior as at least standard implementations of segmentation algorithms do not include a way to detect the quality of a segmentation result.

The main problem of a shared autonomy approach for segmentation is the high user interaction time and restricted learning possibilities. Moreover, experienced user skills are required to work with segmentation tools. As already mentioned above, a direct placement of stem markers in critical segmentation regions is a much more effective way of interaction for an approach like this, where the main focus is on the direct modification of processing decisions and not on learning.

Classification

It is often difficult distinguishing between different plant types. This underlines the complexity of the classification task as the classifier can only be provided with a restricted set

of values describing features of a plant. An evaluation of different classifier types applied on our plant dataset as described in Section 5.1 underlines that an autonomous classification algorithm is not able to handle such a complex task under field conditions with high accuracy.

The user task in this insertion point is to check or assign class labels to contours. A high number of classifiers can be extended with a probabilistic model that assigns certainties to each class label corresponding to a queried instance. This enables us to implement an intelligent query behavior. The user response can easily be employed as direct input in the classification process but it can also be stored for future classifier training. Depending on the choice of the classifier, online-learning is possible. This means the classification model is altered during runtime.

Additionally, a supervised classifier is depending on training data generated by a supervisor. The shared autonomy approach facilitates the generation of such a training dataset.

Stem Localization

The localization of the plant stem is a challenging task as its position varies within the plant contour (compare Section 1.1). While it is hard implementing a sophisticated algorithm for this processing step, a trained user is in most cases able to solve this task without problems. The stem position can be marked with one single click per weed plant, so that the query response is the image coordinate of the plant stem.

As for the segmentation step, this input can be easily used to correct autonomous results for particular instances, however it is difficult to employ it for a learning process, as a high number of additional feature values such as skeleton features [49] would be required in order to establish a supervised learning concept for stem localization. Such a system would also be required in order to realize an intelligent query behavior.

3.1.2. Realization

Each user insertion point has a variety of pros and cons that must be considered when creating a shared autonomy system. The segmentation is a critical processing step, but user involvement is time-intensive, it requires high user skills and the learning possibilities of the autonomous system are limited. The latter problem is also one downside of the user involvement in the stem localization step. Furthermore this is a special operation which cannot be transferred to other use cases.

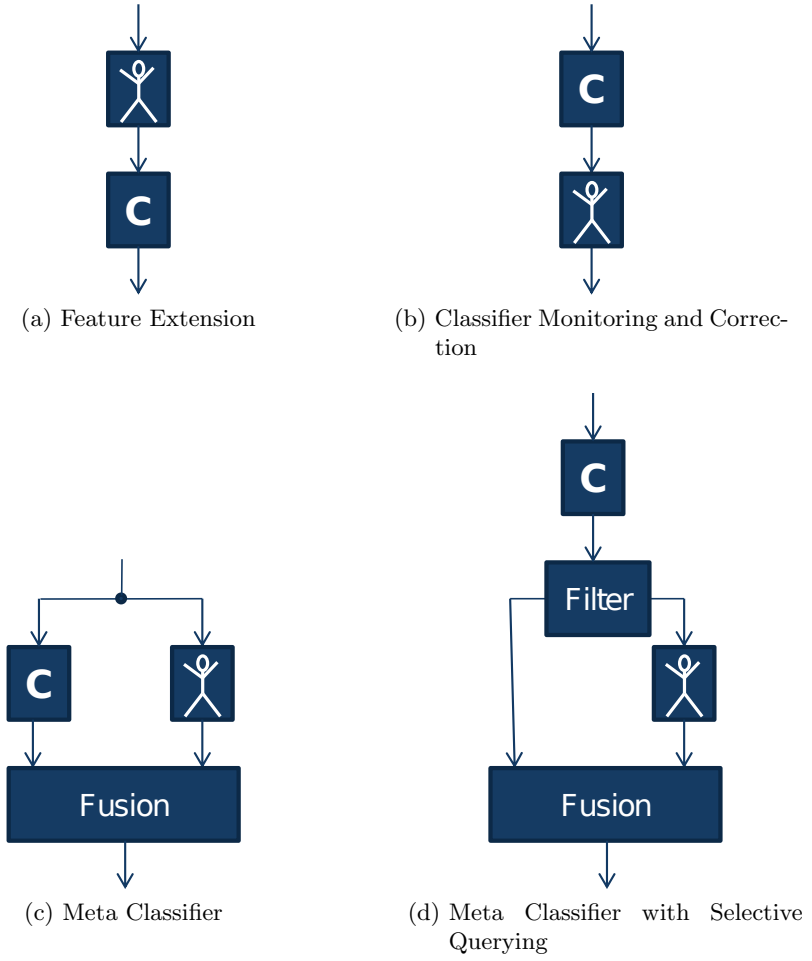
The most promising concept is the user integration into the classification process. Not only enables us this approach to selectively filter the samples sent to the user and employ the user feedback in a learning process, but also is the classification one of the core components in the weed detection pipeline. High performance improvements with a minimum amount of user input seem to be achievable for a shared autonomy classification scenario. Furthermore, this processing step is more generic than the other two ones suggested. Therefore it can be applied in the scope of a lot of other scenarios and is not restricted to the plant classification domain.

3.2. User-classifier Interaction

There are different approaches for the integration of the user in the classification step. In this section, we introduce and analyze four different concepts.

3.2.1. Interaction Scenarios

Figure 3.1 provides an overview of four potential ways of user-classifier interaction. All of them have in common, that the user feedback can be employed for the creation of additional training data for the classifier. This feedback path is not displayed in the figures for the reason of simplicity. Each concept is based on the usage of one user and one classifier and has an acyclic dataflow except of the classifier feedback. The concepts can be extended to more complicated arrangements by introducing several users/classifiers or a cyclical dataflow.



(a) Feature Extension

(b) Classifier Monitoring and Correction

(c) Meta Classifier

(d) Meta Classifier with Selective Querying

Figure 3.1.: Different concepts for user-classifier interaction for object classification

Feature extension

As can be seen in Figure 3.1a, the user processes the data before the classifier. He extends the existing feature vector with additional features. These components can be defined differently, for example one feature can be a simple numeric value defining the plant class. Also more abstract values are possible such as simple visual attributes of the plant, the amount of leaves or the estimated stem height for example.

The advantage of this approach is that the user does not necessarily require the expertise to assign plant labels to a contour. He might already support the classifier by extending the feature vector with information, which is hard to obtain autonomously but can be easily given by a human. The idea to utilize simple information from an user in a human-computer cooperation in order to solve a problem which cannot be solved by the user directly is described by Branson et al. [50].

The downside of this concept is the missing selective sampling in order to reduce the load on the user and the complete lack of a learning process as long as the user does not assign plant labels.

Classifier monitoring and correction

Figure 3.1b displays the second concept. The classifier assigns labels to the data and the user verifies and corrects them if necessary.

It is obvious that, in this approach, the user has a much higher priority compared to 3.1a as he is the final decision-making instance in the classification process. This implies that he must be proficient in his task. Once again, the user has to process all data and cannot be requested distinct instances. Compared to 3.1a the user has access to additional classifier information, which might facilitate its task or also influence him in a counterproductive way.

Meta classifier

The user and the classifier are setup in a parallel layout (Figure 3.1c). Both user and classifier process all the data and after this their output is fused.

In this approach, the user has to classify all objects and cannot rely on any additional classifier information for his decision. The data fusion step allows a flexible weighting of user and classifier influence that can be varied based on the classifier's certainty for an assigned class label or the expected user expertise.

Meta classifier with selective querying

In Figure 3.1d the concept of the meta classifier is extended by a filter which enables us to realize a selective querying scenario. All autonomously classified instances are filtered and send to the user, if a defined criteria is met. The user feedback is integrated into the classification result.

This approach combines most advantages of the other other three concepts: The user can utilize the autonomous classification results, the load on him is decreased and the balance between user- and classifier input can be adjusted. As the results of the classification algorithm are available before the filtering step, this information can be utilized for an intelligent filtering of instances. In case the user response directly alters the classification and the filtering model, this scenario becomes an Active Learning approach.

3.2.2. Realization

As we consider user input highly cost-intensive and want the ability to experiment with different filter and fusion approaches, we design our weed detection pipeline in the style of Figure 3.1d. The flexible configuration of filtering parameters allows also an usage as described in Scenario 3.1b.

Figure 3.2 provides an overview of the weed detection process with user integration. The first four steps Image Preprocessing, Segmentation, Extraction of Feature Values and Classification are executed autonomously. After that, a filter decides for which of the objects additional information in the form of user input has to be requested. This feedback is received from a graphical user interface interacting with the Remote Farmer and afterward fused with the pipeline data. Finally, the weed stems are localized and sent back to the robot with an image for Visual Servoing.

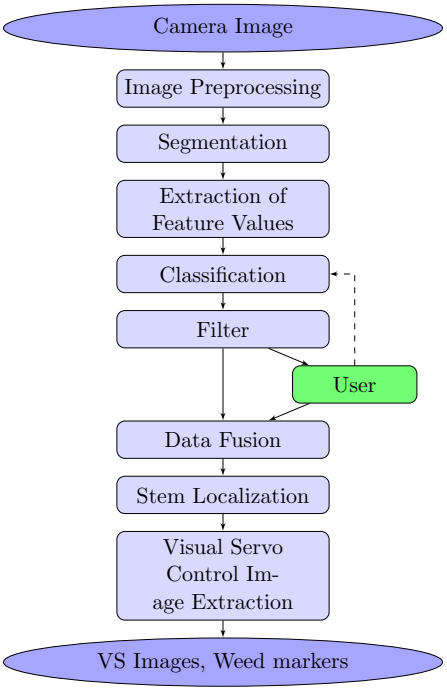


Figure 3.2.: Weed detection process with shared autonomy classification

Query strategy for the filter

We have to formulate a query strategy for the active learning approach . There are several possibilities how to select the samples to be labeled by the supervisor. One is to query data randomly, but results as described by Costa et al. [51] show, that this method is ineffective. A promising approach is *Uncertainty Sampling* [52]. The idea of this strategy is to get those instances labeled by a supervisor which lie in regions where the classification model has a high uncertainty. For example, these instances are points which are close to the hyperplane separating two classes in a Support Vector Machine classifier. It is not only desirable from the active learning point of view to request user feedback for uncertain classification results, but also, because these instances should be checked by the supervisor before they are accepted and further processed in the pipeline.

Data fusion

In case the user assigned a plant label to queried instances, his feedback has to be fused with the autonomous classification results. The data fusion should depend on the uncertainties assigned to the label-providing sources, so that the source with the higher certainty is considered with an heavier weight.

We consider the user input as correct and to not model errors made by the user. This is left as an area for future work. Therefore, an user-defined label corresponding to a plant contour overwrites the label assigned by the classifier. When the user ignores a queried uncertain instance it is considered as non-weed object, as a high precision in the weed detection process is preferred over a high recall.

Optional user input

The results of the weed detection pipeline highly depend on the plant segmentation quality, as there is no way provided to correct segmentation failures. As discussed, a direct user involvement with the aim to correct segmentation results is inefficient. Therefore, we

implement an optional tool which enables the user to overcome segmentation mistakes: Markers for stem positions of weed plants can be set in the images presented to the user. In this way, under- or unsegmented weed plants can still be treated in the end.

3.3. User Interface

The task of the graphical user interface is to visualize computational results of the weed detection process and enable the user to interact with the detection pipeline by providing feedback.

3.3.1. Interface Concepts

We designed different mockups for user interfaces in the scope of our thesis. As the interface will mainly be used by workers not familiar with computer sciences, we put a high focus on the implementation of an intuitive interface with a game-like design that catches the permanent attention of the user and motivates him to carry out his task with a constantly high performance. All mockups are created in such a way so that they can either display still images or a vertically scrolling image stream. This enables plant labeling on overlapping as well as non-overlapping images. Furthermore, we employ the plant contours for visualization purposes and integrate the possibility to incorporate labeling results of an autonomous classifier in the visualized data. We present four different GUI mockups in Figure 3.3.

Concept 1 User interaction requests are presented in the form of speech bubbles. Each bubble refers to one plant, yet depending on the query strategy some bubbles might not even be displayed at all. The user assigns class labels by clicking on a button. Classes which are considered by an autonomous classifier as highly probable could be highlighted to support the user in his decision. The advantage of this interface is an animating design. The speech bubbles give the user the feeling of a very direct communication with the autonomous system which is supposed to motivate him in his task. The downside of this mockup is that the bubbles can occlude some important information in the image and this concept cannot be applied for high plant densities.

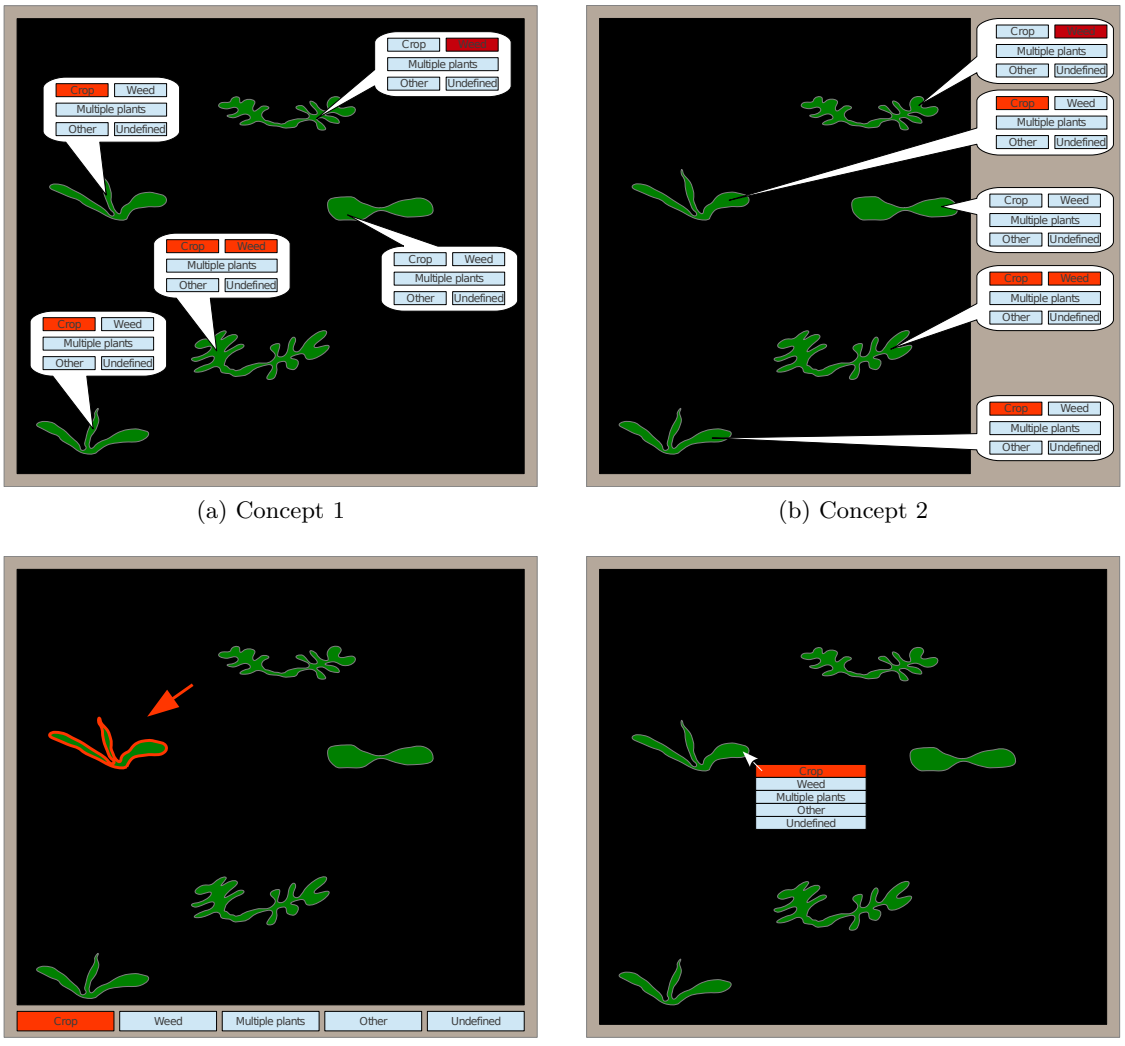
Concept 2 Concept 2 enhances Concept 1. The bubbles scroll now synchronously with the image stream in a bar on the right of the GUI and therefore cause less occlusion. This concept can only be realized in combination with a very restricted plant density.

Concept 3 In Concept 3, single plants are highlighted iteratively. The user has to assign labels referring to the currently highlighted plant by clicking one of the class buttons on the bottom of the interface. Additionally, keyboard shortcuts can be provided so that this interface does not depend on mouse input. The next contour is highlighted as soon as the user clicks one of the buttons. This approach provides an easy way to assign class labels to a completely unlabeled dataset. However, iterating over the contours can be cumbersome when the user just wants to check classifier-assigned labels quickly.

Concept 4 In this concept, a right-click of the user on a plant contour opens a context menu with the labeling choices. The advantage is high flexibility and low occlusion. The disadvantage is a high clickload, two clicks are required to assign one class label, whereas all other concepts require only one click.

3.3.2. Realization

In the final realization of the user interface, we incorporated the ideas of *Concept 3* and *Concept 4* as they complement one another: An iterative high frequency class label assignment by keyboard is possible as well as a selective choice of a fraction of contours which require class label correction. Additionally, we add some features:



(a) Concept 1

(b) Concept 2

(c) Concept 3

(d) Concept 4

Figure 3.3.: Design mockups for user interfaces

- The plant contours are slightly dilated and overlay the tile or full image for a maximum amount of scene context.
- Class labels assigned by an autonomous classifier are displayed below each contour. This facilitates the label correction task of the user. The query strategy decides, whether a contour is marked as *unlabeled*.
- In case of a scrolling image stream, the currently highlighted contour iterates to the next one when it reaches the image border and a label is already assigned—either by the classifier or by the user. If an unlabeled, highlighted contour reaches the image border, the tile stream stops scrolling and waits for user input. This ensures that the user does not forget to assign all labels.
- When the stem marker mode is enabled the user can place additional stem markers by a left-click on points in the image.

Figure 3.4 is a screenshot of the GUI implemented in the scope of our project.

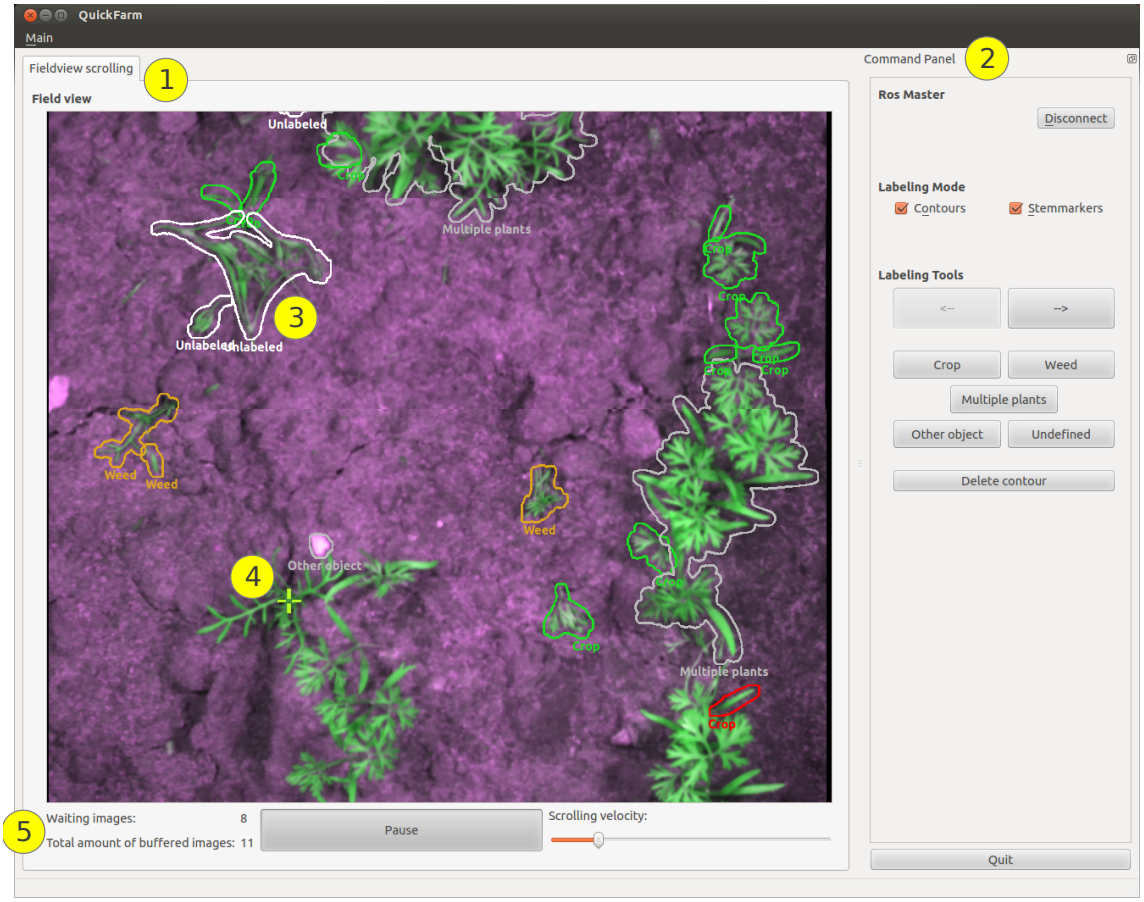


Figure 3.4.: Interface for user feedback in the weed detection process

The field view tab (1) displays a field scene scrolling from top to the bottom and overlaid with stem markers placed by the user, detected plant contours and their assigned classifier labels. In the current configuration, classifier labels with a low certainty are visualized as *unlabeled* (3) so that a label has to be assigned by the user. All other labels are displayed unchanged and can, but do not have to be modified or confirmed by the user. As already described, labels can be changed either by a context menu which is displayed when the user right clicks on the contour in question or by iterating over the contours with the buttons provided in the Command Panel (2) under Labeling Tools. A click on a class button assigns the corresponding label to the contour currently highlighted in red and

moves the iterator to the next contour. While the context menu provides a convenient method to modify only selected contour labels, is the button approach highly valuable for iterating quickly over a completely unlabeled dataset.

Several plants are unsegmented at the bottom of the field view, because they were filtered out as there exists no full image containing the whole contour (see Section 5.2) which indicates that multiple plants are segmented within one contour. Such segmentation failures can be compensated by the user placing a stem marker at the position of weed stems (4).

The bar at the bottom of the field view (5) gives an overview of the amount of images waiting in the GUI queue and enables to user to pause the scrolling process or to change the scrolling velocity. Beneath the Labling Tools contains the Command Panel (2) on the right a button which establishes the connection with the image processing pipeline and two checkboxes for the choice of the labeling mode. This enables us to evaluate different user interaction scenarios as only contours, only stem markers or a combination of both can be displayed in the field view and therefore also used for the generation of user feedback.

4. Concept of Tile Images

In this chapter the tile image approach is introduced. We motivate the need for this concept, introduce *tile contours* and describe the dataflow in our shared autonomy system. The last section of the chapter explains and discusses the tile creation algorithm.

The images sent from the robot contain overlapping areas and are created while the robot moves linearly over the field. Although the same weed plant can appear in several images, the robot only needs to manipulate it once. As our goal is to detect weed plants based on this image data, it is necessary to gain transformation information in order to merge image-associated data such as plant contours.

The only external data which can be employed to create transformations between the images is the robot's odometry values which have a much higher uncertainty than required for the image data association. Therefore, we use image information in order to create a mosaic out of the original images received from the robot, the so-called *tile stream*. The basic idea of this approach is illustrated in Figure 4.1. A *tile image* is the partial image data of an image as received from the robot, in the following called *full image*. Each tile is built such that redundant information contained in the overlapping full images is non-redundant in the tile images—the tile images are non-overlapping. One tile contains data of one full image and exactly one tile is created per full image. The tile stream is an image mosaic built by assembling neighboring tiles at their upper and lower image borders. In the following, we assume that the full images are oriented in such a way that a tile created out of the latest image can be assembled to the bottom border of the image stream (see also Figure 4.1).

The tile stream ensures that transformations between arbitrary full and tile images are available as long as the images captured from the robot are overlapping in such a way that a registration between each two neighbor images can be found.

4.1. Advantages

The main motivation of the tile stream is the association of information from several full images. More advantages are:

Input layer for the shared autonomy approach In the shared autonomy approach, the performance of the autonomous detection pipeline is improved by user feedback. As we are required to reduce the expensive user input to a minimum, the tiles provide

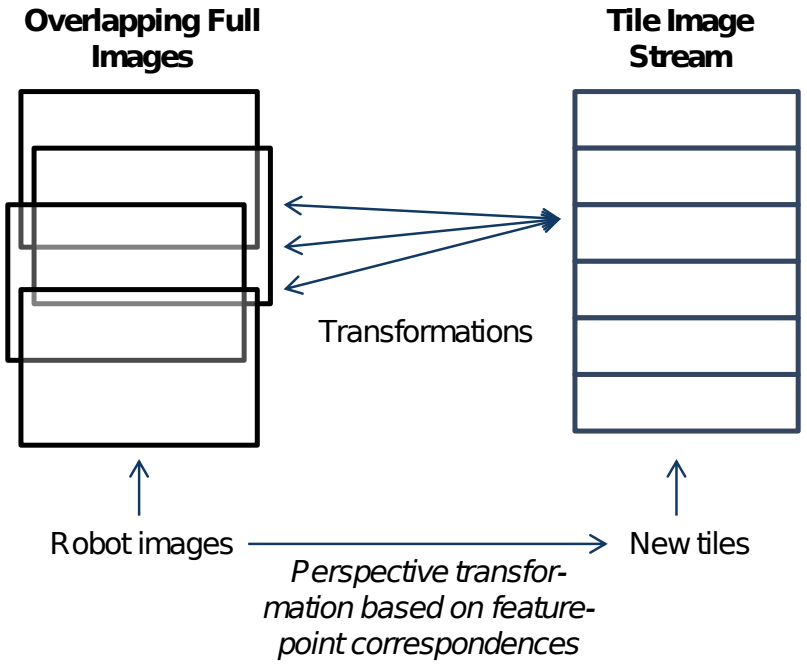


Figure 4.1.: The tile approach. One tile image is created out of every image received from the robot. The assembly of tile images results in the tile stream.

a good layer for the pipeline-user interaction, as it is ensured in this way that the user does not process redundant image data.

Extended scene context for the user The tile stream allows us display a scrolling image to the user that results in a better understanding of the scene context. The user has the feeling of receiving a 'live'-camera image from the robot and it is sufficient to observe newly scrolled-in parts of the stream instead that a whole image has to be scanned every time a new one is displayed to the user.

Reduction of transmission data The user has to be able to interact with the system from a remote destination, as this improves the user convenience of the system and can even enable future developments such as a central "call center" operating weed control robots simultaneously. Since the bandwidth available for data transmission may be limited, it is desired to keep the amount of image data sent to the user as low as possible. The tiles are a more efficient way of sending the complete field scene captured by the robot to the users without transmitting redundant image regions. We evaluate the reduction of transmission data in Subsection 7.1.2.

4.2. Image and Tile Contours

The segmentation step yields contours describing plant shapes in the full images. It is important to understand the difference between the two existing types of contours and their relationship. Each full image contour (*full contour*) is associated with a *tile contour*. This is the transformation of a full contour to the tile image stream. Therefore, the newly detected image contour is transformed into the tile space first. There are two possibilities now: either it is detected that the transformed contour overlaps with another tile contour. In this case, both contours are merged to one common tile contour. The merge process selects one of both contours for tile display and stores a list of image contours associated with those tile contours (compare Figure 4.2). Or the transformed contour does not overlap with another tile contour, probably because it is detected in a field region that has not

been covered by any previously processed image yet. Then a new tile contour is created. The algorithm for the detection of overlapping contours is described in Subsection 5.2.3.

The relationship between the different image and contour types is illustrated in Figure 4.2. A full image contains an arbitrary amount of contours. Each is associated with one tile contour. However, a tile contour can contain the information of several full contours. One tile contour belongs always to exactly one tile image. Per definition, this is the latest image received from the robot which contains a part of the contour.

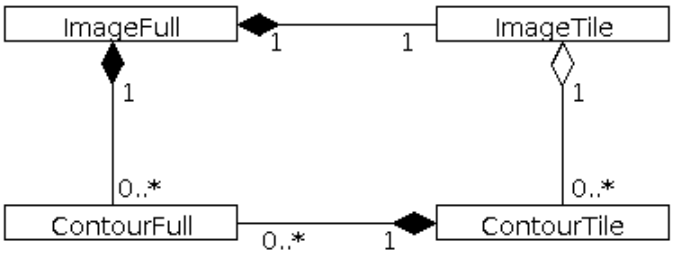


Figure 4.2.: Relationships between the different image and contour types

4.3. Dataflow

One important question the pipeline design depends on is how the data between tile and full images are associated and which processing step is based on what kind of image. Figure 4.3 gives on overview of the dataflow between the different layers. A general rule of thumb is, that all autonomous image processing is operated on the full images in order to avoid additional errors due to tile creation uncertainties and because the image processing library employed in this project (see Section 6.2) expects single images and no image stream as input data. On the other hand, the tile image stream is utilized as base layer for user interaction and for contour data merging.

At first, the images received from the robot are converted to the data format used in the processing pipeline and stored in a buffer as one thread receives images and another one is responsible for the image processing. As soon as the latter thread is done with the previous processing tasks, it takes the last image out of the buffer, preprocesses it and creates a tile image.This process is described in Section 4.4. Plant contours and their corresponding feature values are extracted out of the preprocessed full image and either associated with existing tile contours or used for the creation of new ones as described above.

Next, the full image contours are assigned class labels and certainties by the classifier. The label of a tile contour is determined based on the labels of its associated image contours. Therefore, this label cannot be assigned immediately, because it has to be waited until all associations of this tile contour are established. Instead, the image data is temporarily buffered until it is ensured that no new incoming images will alter the tile information of this image. This is the case as soon as the upper border of the latest image received from the robot and transformed to fit into the tile image stream is below the lower border of the tile in question. As soon as this criteria is met, the tile contour labels and certainties can be determined (compare Subsection 5.4.4) and tile data requiring feedback is sent to the Remote Farmer.

All images are buffered until the corresponding user feedback is received. Buffer 3 has the additional function to control the access to shared states between the thread that processes the images and another one which receives the user feedback and postprocesses the data. It is ensured that the user handles all requests sequentially in the order of reception. The user response is received back in the pipeline, added to the tile data and if possible employed

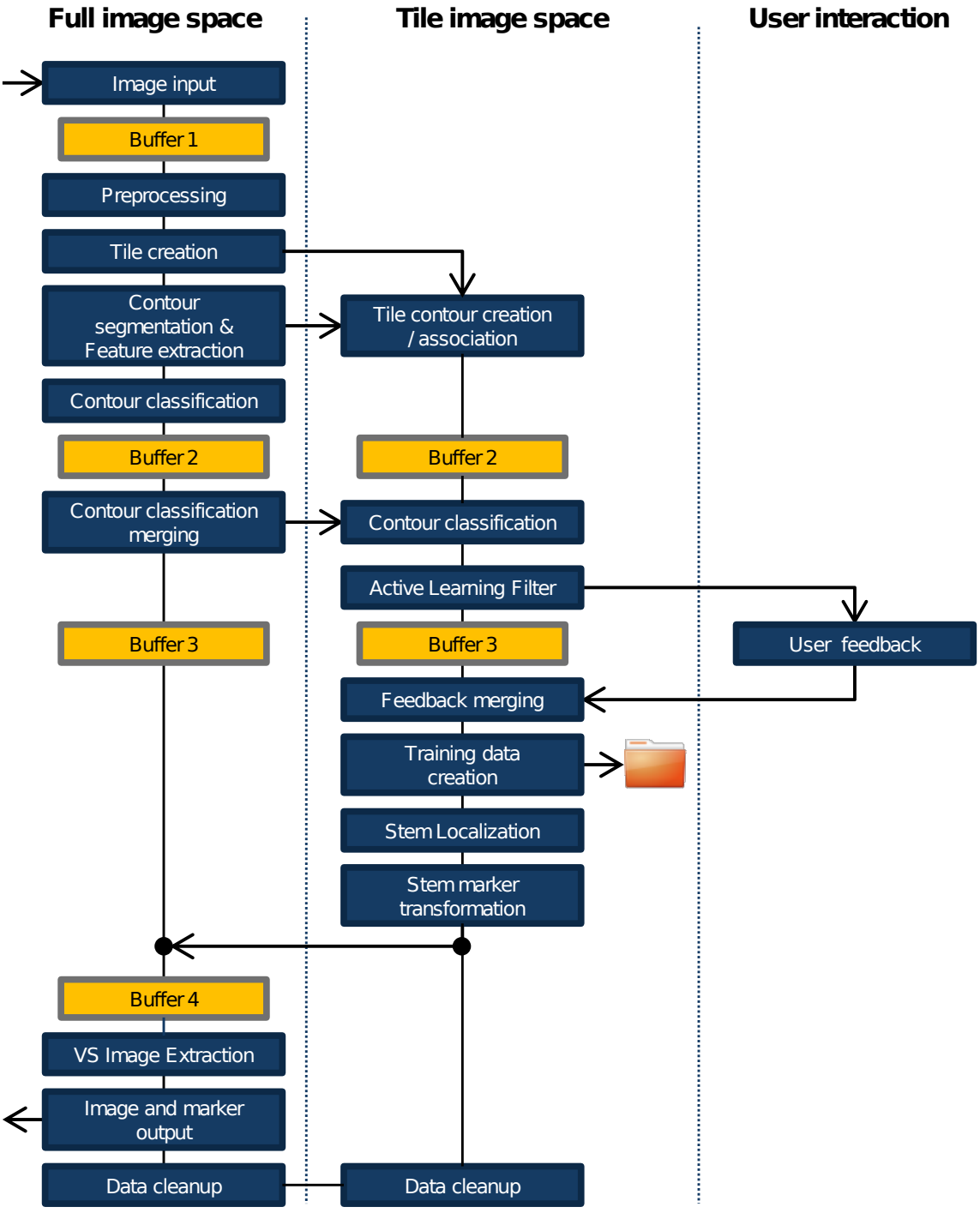


Figure 4.3.: Dataflow between the full image layer, the tile image layer and the user

for the creation of training data. This dataset can be used for extended classifier training on the next pipeline startup.

All plant contours that are marked as *weed* are consulted for the localization of the stem positions of weed plants (compare Section 5.5). These stem markers are transformed back into the original full images. A buffer stores the full image data until it is ensured that no additional markers from tile images will be mapped to this full image. This is the case as soon as the lower border of the full image transformed into the tile stream is above the border of the last tile received back from the GUI. Then the visual servoing image is created out of the original image data received at the very beginning from the robot in order to ensure the highest recognition value. In the current pipeline implementation, the whole full image and all associated markers are published as this approach facilitates debugging and pipeline demonstrations. For the system integration, one visual servoing image should be extracted per marker and sequentially marker-image-pairs published back to the robot. Finally, the data that was sent is cleaned up.

4.4. Tile Creation

The goal of the tile creation process is to create a tile image out of each original image received from the robot. This tile is created directly after a new image was received and preprocessed. A new tile extends the existing tile image stream when assembled to the lower border. We assume again, that the full images are overlapping, captured during a linear robot motion and oriented in such a way that a tile created out of the latest image can be assembled to the bottom border of the image stream. Our system is also able to process non-overlapping image data. However, adjustments should be made if the image input is generally not overlapping in order to exclude the mosaicing step from the processing pipeline.

There is a high number of algorithms available, that can be used to stitch panoramic images. Szeliski [53] provides a good introduction in the steps required for image stitching. A state-of-the-art solution is the algorithm for multi-raw stitching described by Brown and Lowe [54]. It is implemented in the software *Autostitch* and also the OpenCV image stitching pipeline is based on this algorithm. We cannot directly employ this approach as it does not assume linear camera movement but the rotation of the camera around its optical centre instead. Furthermore, there are several requirements beyond the scope of a typical image stitching application:

Number of images Typical panoramic image stitching applications are based on the assumption that only a very limited number of input images is used. However, we create a planar mosaicing algorithm which has to be able to handle a very high number of images as typical dam-lengths treated by the weeding robot can be several hundreds of meters. It has to be ensured that the transformation of images does not create singularities (see Subsection 4.4.3).

Mosaicing Contrary to a typical image stitching software it is not required to blend neighboring images, as a clear border between tile images enables us to recognize matching problems and does not blur the information on which part of the tile an user is working currently. We use the term *mosaicing* instead of *stitching* to underline that there is no blending.

Transformation One of the main purposes of the image mosaicing process is to obtain transformation information between neighboring images. Therefore all image mosaicing steps have to be chosen in such a fashion that forward- and backward transformations between two images are stored and can easily be calculated.

4.4.1. Transformation

Under the assumption that plant objects and soil lie in one plane, the camera is not mounted completely vertical to this plane and the camera image is undistorted, each overlapping image part is a perspective transform of a cut-out of its previous image. Two neighbor images can be matched to a mosaic by finding this transform. Therefore we use a homography in order to describe the transformation between two images, as also applied in [54]:

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j \tag{4.1}$$

whereas $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{u}}_j$ are homogenious image coordinates $\tilde{\mathbf{u}}_i = s\begin{pmatrix}\mathbf{u}_i \\ 1\end{pmatrix}$, $\mathbf{u}_i$ being the image coordinate.

4.4.2. Mosaicing Algorithm

A feature matching approach (compare [53, p. 30ff.]) is used in combination with a RANSAC filter [55] to determine the perspective transformation between two images. Afterwards, the tile image is created as a cut-out of the transformed image.

The mosaicing algorithm is based on [54] and an OpenCV tutorial for feature point matching[1]. The following steps are executed in order to create a new tile image which can be assembled to the top border of the latest tile:

1. Extract feature points and their descriptors out the new image. We use *SURF* features [56], as they are invariant towards translation, rotation, scaling and also robust towards changes of the viewpoint and the illumination within limits.
2. Match the detected features with the feature points of the last tile image. Therefore the *FLANN* library [57] for approximate nearest neighbor search is employed.
3. The best feature matches are found based on their descriptor distance d_i. For this, we calculate a distance threshold y_{thresh} and use all matches with $y_i < y_{\text{thresh}}$:

$$y_{\text{thresh}} = \min_{y_i \in \mathcal{M}_{\text{dist}}}(y_i) + k\Big(\max_{y_i \in \mathcal{M}_{\text{dist}}}(y_i) - \min_{y_i \in \mathcal{M}_{\text{dist}}}(y_i)\Big) \tag{4.2}$$

   $\mathcal{M}_{\text{dist}}$ is a set containing all matching descriptors' distances. $k \in [0;1]$ defines the range of descriptor distances which are considered to be good matches. Based on visual obervations, a value of $k = 0.3$ yielded in the best mosaicing results for our case. Figure 4.4 shows the feature matches of two images after the initial filtering process for $k = 0.15$.
4. Calculate the perspective transformation $\mathbf{H}_{ij}^{(1)}$ between the two images. A RANSAC filter [55] ensures robustness by ignoring outliers. We use a reprojection threshold of 20 px. This means a match of feature points is considered to be an outlier as soon as the point distance is larger than the given threshold after the feature points of the new image were transformed with the calculated perspective transformation. Lower values for the reprojection threshold yielded in worse mosaicing results.
5. Transform the new image with $\mathbf{H}_{ij}$, so that a mosaic image can be created out of both images. Figure 4.5 shows the last tile and the transformed new image which will be used for tile creation.

[1] `http://goo.gl/TQ7BL`

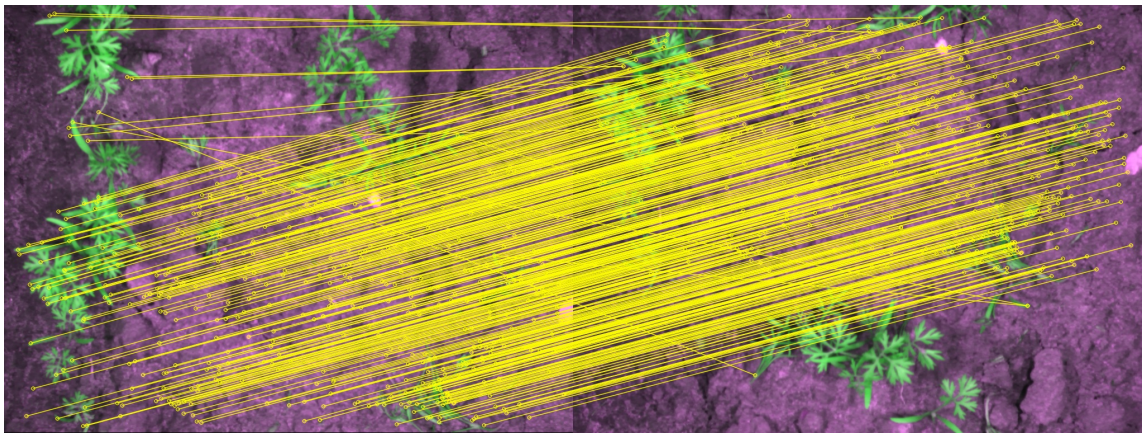


Figure 4.4.: Filtered feature point matches for two images

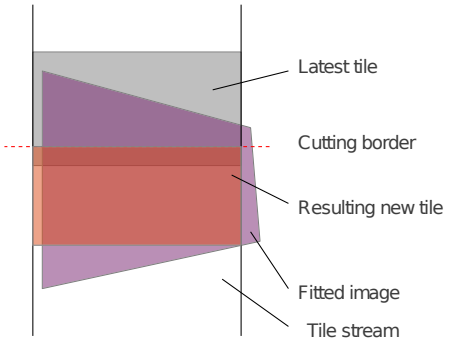


Figure 4.5.: The transformed image and its resulting tile in the tile image stream

6. The new tile image is created as a cut-out of the transformed image. Therefore a horizontal cutting border is determined first. Its vertical position is the mean value of the position of the upper border of the latest tile and the upper corner of the lower border belonging to the fitted image. The latest tile is cropped at this cutting border. The rectangle for the new tile is defined by the cutting border, the left and right border of the tile stream and the lower intersection point of the upper border of the transformed image with the tile stream's borders. In case the intersection point does not exist because both upper corners of the fitted image lie within the tilestream, the new tile's upper border is restricted by the lower upper corner of the transformed image.

7. Finally, we transform the position of the feature points detected at the beginning of the algorithm into the tile stream as well, so that they can be used for the matching with the next incoming full image.

4.4.3. Tile Deskewing

Figure 4.6 shows the tile stream resulting from the mosaicing algorithm applied on 21 full images. A problem becomes clear that is already indicated in Figure 4.5: For a high number of images the tile stream becomes increasingly distorted. In our case a small tilt of the camera results in progressively deformed tile images. In order to handle these deformations, we introduce an additional perspective transformation.

At first, the fitting transformation $\mathbf{H}_{ij}^{(1)}$ is obtained as described in Subsection 4.4.2. Next, this perspective transformation is not applied to the full image but only to four points describing the border of this image. This yields a quadrangle representing the position of

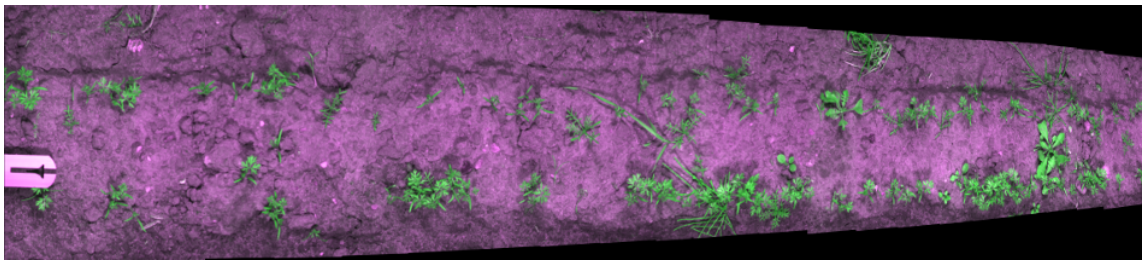
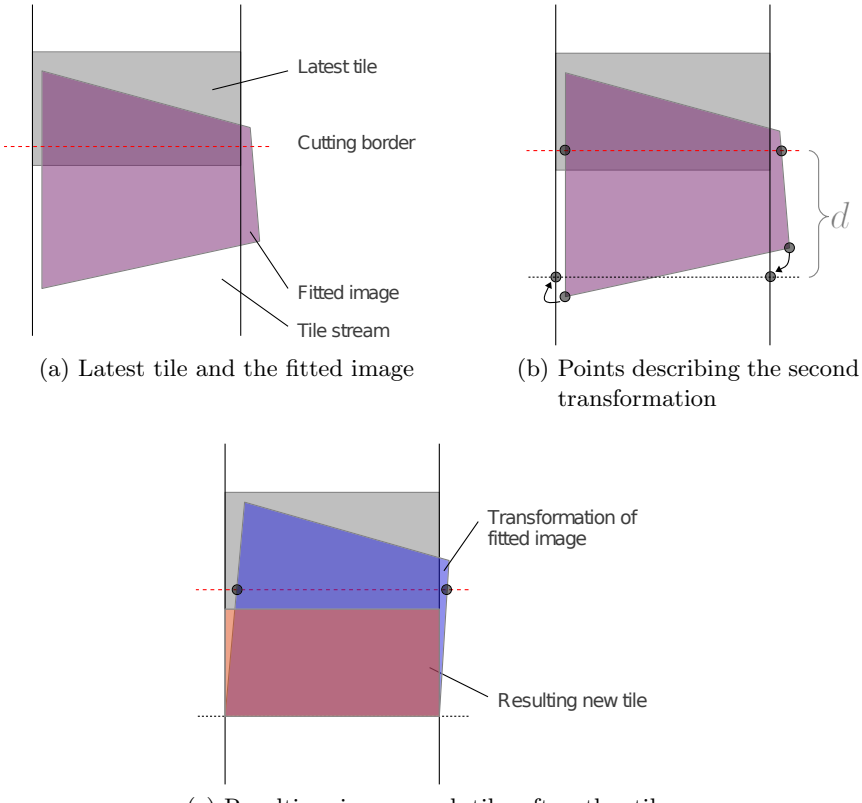


Figure 4.6.: Assembly of 21 tile images without tile deskewing. The bottom of the tile stream is on the right.

the fitted image in the tile stream (Figure 4.7a). Now, a second perspective transformation can be applied with the goal to fit this image into the tile stream.

Therefore, we detect the points of intersection between the fitted image's borders and the cutting border (Figure 4.7b). These points will be fixed in the second transformation in order to ensure the newly created tile still fits to its predecessor. Next, the lower corners of the fitted image are dragged onto the lines which define the borders of the tile stream. The mean value of the distance between the upper and lower points of each side transformed back into the original full image define the vertical distance d to the cutting border. This ensures that a vertical stretching of the image is corrected.



(a) Latest tile and the fitted image

(b) Points describing the second transformation

(c) Resulting image and tile after the tile deskewing

Figure 4.7.: Steps of the tile deskewing process

To sum it up, we have four pairs of points: Each couple defining one origin and one destination point, whereas two pairs contain identical source and destination points. Based on these point correspondences, the homography for tile deskewing transformation $\mathbf{H}_{hi}^{(2)}$ is calculated. The final image transformation is the combination of image fitting and tile

deskewing (see Figure 4.7c):

$$\mathbf{H}_{hj}^{(3)} = \mathbf{H}_{hi}^{(2)}\mathbf{H}_{ij}^{(1)} \tag{4.3}$$

The tile creation process is continued with the cut-out of the tile as outlined in 4.4.2 with the cutting border already determined.

The result of the additional tile deskewing step can be seen in Figure 4.8. It becomes clear that this approach overcomes the effect of increasingly distorted tiles shown in Figure 4.6.

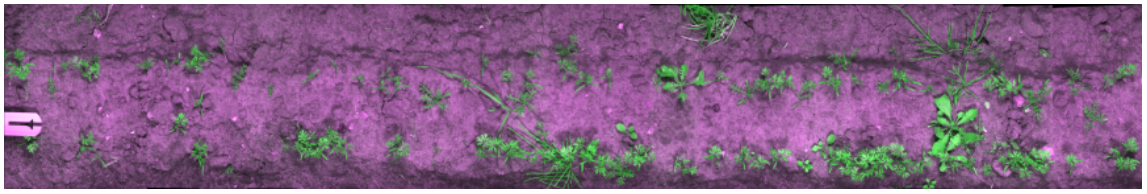


Figure 4.8.: Assembly of 21 tile images with tile deskewing. The bottom of the tile stream is on the right.

4.4.4. Error Handling

It is important to detect incorrect image matches in the mosaicing process. We apply some heuristics in order to detect incorrect matches. As soon as one of the following criteria is met, the mosaicing process is considered to have failed for the image in question:

- The lower corner of the upper edge of the fitted image is above the upper edge of the latest tile image.
- The fitted image is 'twisted' or flipped horizontally or vertically, for example one or both left image corners become right image corners after the fitting transformation.
- The angles between the edges of the warped image exceed given tolerance regions.

In the case of a fitting failure, the whole full image is stored as new tile image and it is remarked that no association between the two full images in question could be found. The same procedure is also used for the first image sent to the pipeline.

4.4.5. Discussion of the Tile Creation Process

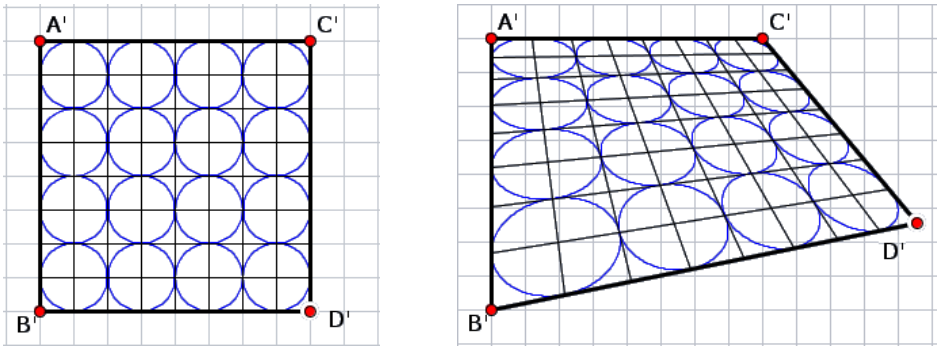
The tile creation process is an effective way to create quickly the tile stream. However, it is based on some assumptions which do not always hold and therefore small failures are introduced in the mosaicing process.

The main problem is that the mosaicing by perspective transformation of the original images assumes that plants and soil lie in one plane. As a high percentage of feature points for image fitting is extracted out of the soil which can be considered to be a plane, mainly plant objects are not fitting correctly between tile borders. This problem is illustrated in Figure 4.9, where the mosaicing algorithm is applied on three example images. While the background of the three original images employed is fitted well, an object which does not lie in the background plane causes mosaicing errors.

A negative side effect of the additional tile undistortion step is that the transformation can alter the pixel positions on the cutting border between the latest tile and the newly created one. The only exception are the two corners of the fitted image which are used as fixpoints. The effect is shown in Figure 4.10. It is visible that the position of transformed

Figure 4.9.: Limitations of the mosaicing algorithm for a non-planar environment

points on the line $\overline{A'C'}$ varies between the two images except for the points A' and C' (see the intersections of the thin black lines with $\overline{A'C'}$). This motivates the need to mount the camera properly orthogonal to the ground on the robot and make sure that the camera images are undistorted before they are sent to the processing pipeline, so that no larger adjustments in the tile undistortion process are necessary any more.



(a) (b)

Figure 4.10.: Illustration of the effects of a perspective transformation with two fixed points A' and C'[2]

In the current approach, one tile image per input image is created. This means, the size of the tile images is determined by the overlapping factor of input images. For a high factor, an unnecessary amount of tile images is created which does not only result in high computation times but also in an increased probability of mosaicing errors. Therefore it is important to ensure in the system integration step that the overlapping factor is around 0.3. In case of a higher overlapping factor, this can be easily ensured by subsampling created camera images, so that just a fraction of them is used for image processing.

[2]Created with `http://www-m10.ma.tum.de/bin/view/MatheVital/GeoCal/GeoCal2x1`

5. Object Classification and Stem Localization

We introduced our approach for the user integration in the weed detection process in Chapter 3. This chapter describes the autonomous part of the process applied on each image sent to the detection pipeline with the final goal to detect the stems of weed plants. It is important that the autonomous system is created with respect to a shared autonomy approach. Whenever possible out-of-the-box solutions are employed as our main focus is the setup and evaluation of a complete shared autonomy framework rather than the optimization of single processing steps. In the following sections, all steps concerning processing of images are explained in the order they are applied on the dataflow.

The pipeline starts with the image preprocessing and the creation of tile images. The sections afterward explain, how plant contours are segmented out of the image data and feature value corresponding to these shapes extracted. After this, we motivate the choice of our classification setup and finally describe, how stem positions can be determined based on plant contours.

5.1. Image Preprocessing

The raw image data is captured by a multi-spectral camera[1] with one Near Infrared (NIR) and RGB channels. Unlike soil and stones, chlorophyll absorbs a high amount of light in the red band but has high reflectance values in the infrared band [58]. This characteristic can be used to subtract the image background [45]. The process is illustrated in Figure 5.1.

5.1.1. Optimization for Channel Subtraction

We add an additional optimization step to achieve better results. The brightness and contrast of the red channel is adjusted iteratively, so that the mean squared error between the value of background pixels in the infrared and the red channel is minimized.

1. Obtain an initial guess for background pixels $\mathcal{B}_0$. We calculate the intensity range in the NIR channel, cut it at 25 % of the value range and use all pixels with intensity values in the lower part. The value of 25 % is empirically derived with the goal to retrieve a subset of background pixels with high precision.

[1] `http://www.jai.com/en/products/ad-080ge`

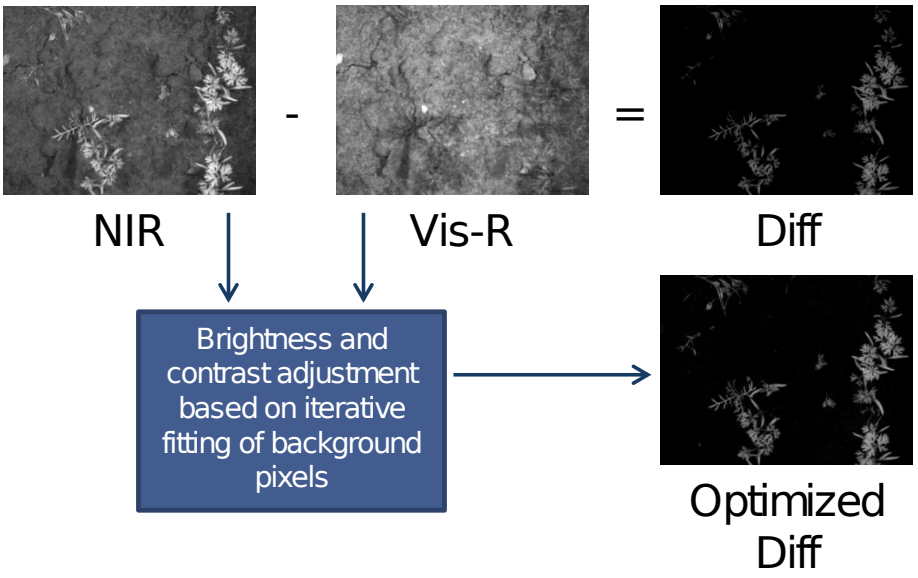


Figure 5.1.: Image subtraction for plant-soil discrimination

2. Iterate over the following steps for a given number of iterations $i = 0..(N-1)$ or until another stop criterion such as a small change of the calculated adjustment parameters is met:

   a) Find the parameters $\hat{\alpha}_{0,i}, \hat{\alpha}_{1,i} \in \mathbb{R}$ which minimize the sum of squared differences between the background pixels' intensity in both channels:

$$E_i(\alpha_{0,i}, \alpha_{1,i}) = \sum_{\mathbf{u}_i \in \mathcal{B}_i} \left[\mathrm{src}_{\mathrm{NIR}}(\mathbf{u}_i) - \left(\alpha_{1,i}\, \mathrm{src}_{\mathrm{VisR}}(\mathbf{u}_i) + \alpha_{0,i} \right) \right]^2 \tag{5.1}$$

   b) Calculate the current optimized differential image

$$\mathrm{src}_{\mathrm{Diff,i}}(\mathbf{x}) = \mathrm{src}_{\mathrm{NIR}}(\mathbf{u}_i) - \left(\hat{\alpha}_{1,i}\, \mathrm{src}_{\mathrm{VisR}}(\mathbf{u}_i) + \hat{\alpha}_{0,i} \right) \tag{5.2}$$

   c) If this is not the last iteration, define a new set of background pixels $\mathcal{B}_{i+1}$. This set includes all pixels of $\mathrm{src}_{\mathrm{Diff,i}}$, which value is below an intensity threshold t_i . We determined experimentally good visual results for a value of $t_i = 20$ for eight bit images.

3. The last differential image $\mathrm{src}_{\mathrm{Diff,N-1}}$ is the result of the optimization process.

This algorithm requires that the background pixels have a wide intensity range. This is the case for the field images in our application. It is important to ensure in the implementation that the pixels' intensity range of a manipulated image does not exceed the values allowed by the data format.

The algorithm can be sped up significantly by decreasing the image size used to estimate the adjustment parameters and by limiting the number of iterations. In our case, a reduction of the image size from 10^5 px to 10^3 px decreases the processing time from 32 s to 10 ms without noticeable changes in the results for $N = 2$. Two iterations are sufficient to find good adjustment parameters, as the α-values converge quickly.

5.2. Segmentation

In this section, we describe the choice and evaluation of segmentation algorithms.

5.2.1. Segmentation Algorithm

The goal of this step is to segment the contours of all plants. The contours are highly suitable for our approach, as on one hand they can be employed for autonomous plant classification and stem localization, but on the other hand they do also fit in the user-centered system design concept, as they provide a good visualization of single plants in the GUI.

Four different segmentation algorithms for contour retrieval are compared on the test dataset. All of them are applied on the differential gray level image obtained in the preprocessing step.

Simple thresholding The algorithm has a low processing time. As a threshold parameter has to be set manually the algorithm cannot adjust to varying lighting conditions.

Thresholding with Otsu's Method[59] A fixed threshold is calculated by the algorithm with the aim to divide the values of an image optimally into two different classes. As darker regions of plants are assigned to the background class, plant details such as stipes are lost.

Adaptive thresholding The basic idea of the algorithm is to facilitate a threshold that varies depending on the image position. The threshold for one pixel depends on an average value calculated out of the pixel's adjacent region. This enables the algorithm especially to handle varying light conditions in an image. As the illumination in the test dataset is convenient, Adaptive Thresholding does not improve the results and produces unwanted noise in regions with a very high or low plant density.

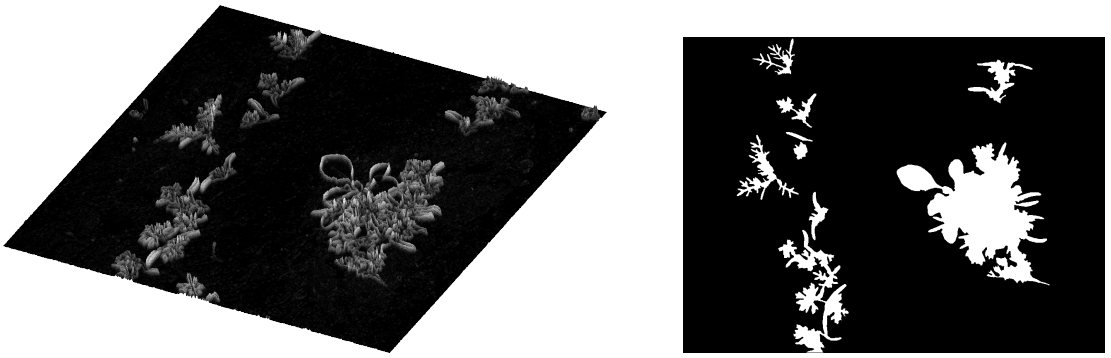
Watershed A non-parametric, marker-based Watershed implementation [60] leads to the best results and is our final choice. The basic idea of a watershed algorithm is to create a relief out of the gradient image. The relief height is the magnitude of the gradient (see Figure 5.2). As object contours result in high gradients, each object is represented by a basin. Positive and negative ground truth markers define 'water sources'. Figuratively, the relief is filled with water, whereas the height of the water level is gradually increased. As soon as the waterfronts of two basins with different types of ground truth meet, an image contour is created.

The advantage of this watershed algorithm is its flexiblity towards changing lighting conditions, low noise and good segmentation results. As Otsu's Method tends to oversegment the dataset its segmentation results are used as positive ground truth. The negative ground truth is defined by simple thresholding with the threshold value defined as a fraction of the Otsu-calculated threshold and a slight erosion. We use $t_{\text{gt-}} = 0.25 \cdot t_{\text{Otsu}}$

The outer object contours are extracted out of the detected non-background watershed regions. A low-cut filter eliminates contours with small area. Contours displayed only in the very left or very right image region are filtered, as only the center strip of the soil dams has to be treated. Additionally, all contours which intersect with a bottom or top image border are filtered out. As a tile stream is created in vertical directions and these contours do not label complete plants, they are not allowed to be displayed in the tile stream. Edge contours of neighbor images could be merged to one contour, but this effort is not necessary, as it is highly possible that, due to the overlapping images, the plant in question will appear completely in another image.

5.2.2. Evaluation of Image Subtraction and Segmentation

As described in Section 5.1 an additional step is introduced in order to optimize the subtraction of the NIR- and R-image channels. We evaluate this step based on a visual



(a) Relief of the gradient image (b) Resulting watershed segmentation

Figure 5.2.: Segmentation with the watershed algorithm

assessment of the segmentation results. Figure 5.3 shows one example output of the image preprocessing module without and with optimized image subtraction and the resulting watershed segmentations.

Figure 5.3.: Top-left: Unoptimized differential image. Top-right: Optimized differential image. Below: Their resulting watershed-segmentations with unfiltered contours. Each image shows two weed (top, bottom) and two carrot plants (middle). In both cases, the upper carrot is split up in two contours. The lower one is segmented correctly in the right image.

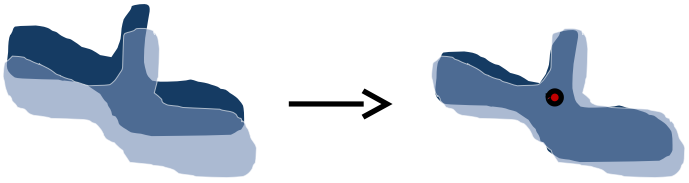
Fine plant contours such as the stipe of the upper weed plant are more distinct in the optimized differential image. This results in less oversegmentation (lower weed plant), while no significant contour details are lost due to the optimization step. The optimization step improves the segmentation results without any strong negative side effects.

5.2.3. Contour Overlapping Detection

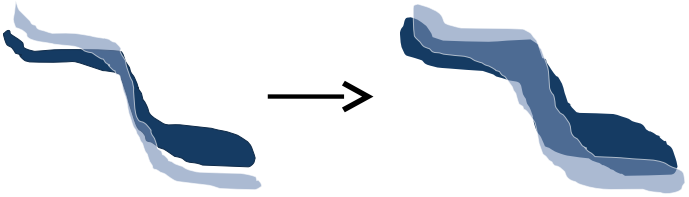
The contours of the same plant which appears in several images are clustered in one tile contour as described in Section 4.2. For this process it is necessary to implement an algorithm which detects overlapping contours in the tile image space. We define two contours as overlapping as soon as the ratio of the overlapping area of the two contours to the area of one of the contours is higher than a threshold n_{thresh}:

$$\frac{A_{\mathrm{C}_1 \cap \mathrm{C}_2}}{A_{\mathrm{C}_1}} < n_{\text{thresh}} \quad \text{or} \quad \frac{A_{\mathrm{C}_1 \cap \mathrm{C}_2}}{A_{\mathrm{C}_2}} < n_{\text{thresh}} \tag{5.3}$$

We employed the empirically determined value $n_{\text{thresh}} = 0.5$. Additionally, we added two optimization steps for the contour overlapping detection. In order to compensate errors from the image mosaicing or a change of the camera perspective, we correct small displacements of two overlapping contours. Therefore we make the centroids of two contours identical, if their distance is lower than a threshold of 20 px before the overlapping ratios are calculated (compare Figure 5.4a). In order to detect the overlapping of fine plant contours more robustly, we dilate all contours with a circle tool which has a diameter of 7 px as shown in Figure 5.4b. These two measures can the source of false-positive overlap detection, but our observation is they resulted mainly in an improved overlapping detection behavior.



(a) Displacement correction



(b) Optimization for the overlapping detection of thin contours

Figure 5.4.: Optimizations in the contour matching algorithm

5.3. Extraction of Feature Values

This section examines the extraction and evaluation of feature values employed in our shared autonomy pipeline.

5.3.1. Feature Extraction

20 shape- and 2 texture-based features are extracted out of the outer plant contours and the texture within. They are completely listed in the Appendix under Section A. All of these features except the contour area are invariant to translation, rotation and scaling[2]. Although they are rarely used for plant identification (compare to Subsection 2.3.3), we use texture features, as our plant contours do not contain holes and these features are able to indicate soil regions within a plant contour.

The shape-based features include amongst others Hu-Moments , Compactness, area ratios of bounding forms to the contour area and statistical values derived from convexity defects. The latter seem to be highly interesting for the description of plant contours, but have only been used in other fields such as human activity recognition [61] yet to the author's best knowledge.

5.3.2. Evaluation of Feature Values

As our feature space with 22 features can be considered as relatively high-dimensional and some state-of-the-art weed detection research such as that of Weis et al. [6] suggests the

[2]Not considering a restricted image resolution.

employment of a lower number of features, we apply a feature selection approach in order to determine important dimensions of our feature vector and to evaluate a reduction of our set of features.

Specifically, we apply the WEKA implementation of RELIEF (Kira and Rendell [62]), a correlation-based feature selection approach. The basic idea of this method is that only statistically relevant features are chosen. The algorithm is inspired by lazy classifiers: For a randomly picked instance the *near-hit instance*, the closest instance with the same class, and the *near-miss instance*, the closest instance which belongs to another class, are determined. For each dimension of our feature vector i the distance of the near-miss instance to our selected instance $d_{\text{near-miss},i}$ and of the near-hit instance to the selected instance $d_{\text{near-hit},i}$ is calculated. The higher $d_{\text{near-miss},i} - d_{\text{near-hit},i}$, the higher is the relevance of this feature. This process is repeated and in the end, a ranking of the features can be done. The advantages of this feature selection methods are according to the authors noise tolerance and "not being fooled by feature interaction".

We select the ten best feature values (compare to Appendix, Section A):

- Perimeter over the square root of the contour area $\sqrt{A_{\text{C}}}$ (F8)
- Eccentricity of a bounding rectangle (F12)
- Number of convexity defects (F13)
- Median of the depth of convexity defects divided by $\sqrt{A_{\text{C}}}$ (F14)
- Mean value of the depth of convexity defects divided by $\sqrt{A_{\text{C}}}$ (F15)
- Standard deviation of the depth of convexity defects divided by $\sqrt{A_{\text{C}}}$ (F16)
- Median of the length of convexity defects divided by $\sqrt{A_{\text{C}}}$ (F17)
- Standard deviation of the length of convexity defects divided by $\sqrt{A_{\text{C}}}$ (F19)
- The normalized mean of intensity values in the differential image (F20)
- The normalized standard deviation of intensity values in the differential image (F21)

It is interesting to note that none of the Hu-moments are considered relevant, whereas our features derived from convexity defects were selected as well as the texture features. Two other feature evaluation approaches, one based on Information Gain and another on correlation-based subset selection [63] yielded similar results. They additionally incorporated the contour area.

An evaluation of different classifier types with the selected feature subsets shows that the classification accuracy varies within a few percents points compared to a classification based on all features. It seems to be possible to reduce the dimensionality of the feature space in order to optimize the processing time. However, we evaluate our system with all feature values as we want to avoid additional uncertainties introduced by feature subset selection.

5.4. Object Classification

The function of the classifier is to assign class labels to each segmented contour. This information can be used either to detect weed plants autonomously or to support the Remote Farmer. In general, classifiers can be divided into two different types:

Supervised The classifier is provided with a priori knowledge in the form of a training dataset containing supervisor-labeled instances. The classifier's model is either created generalizing its training data before a query (eager learning) or the classifier uses

the provided training data directly for the prediction of labels for unseen instances (lazy learning) during the query.

Unsupervised An a priori training dataset does contain instances and their associated features, but no class labels. Based on this information, the classifier is able to group instances (clustering) but cannot predict class labels. This method is applied when one or all classes a population can be divided into are unknown.

5.4.1. Choice of Classes

As all classes for contours are known in our use case, we rely on a supervised classifier. We define the following classes:

Crop Contour around one crop plant or a part of it.

Weed Contour around one weed plant or a part of it.

Multiple Plants Resulting contours from under-segmentation that contain several plants or the parts of several ones.

Other Object A non-plant object, e.g. stones or pieces of wood.

Undefined This label can only be assigned by the supervisor and is applied for segmentation errors or plants which cannot be classified by the supervisor. When the training data for the classifier is created, all instances that belong to the class *Undefined* are filtered out.

A simple binary classifier for the detection of weed plants is not utilized because it limits extension possibilities. For example, it might be necessary to filter detected weed plants based on their distance to crops. If a weed plant is very close to a crop it is not allowed to be manipulated in order to ensure that the crop is not damaged. Furthermore, image regions recognized as multiple plants could be processed with additional user input in order to detect possibly included weed plants.

5.4.2. Classifier Requirements

In order to find a suitable classifier, the following points were considered:

Multi-class classification As multiple classes are defined, the classifier has to support multi-class problems. This is in general possible for all classifiers, as multi-class problems can be split up in binary problems by decomposition [64]. Famous approaches are “one-against-one“, where each class is compared against each other classes and “one-against-all‘, where multiple binary classifiers are trained to compare one class against all other remaining ones summed up.

Performance in the use case A basic classifier requirement is that it predicts queried data as correctly as possible. Furthermore, the classifier has to be able to generalize given a priori training information so that it behaves robustly. This means that the classifier does not only perform well on a known dataset, but is also able to transfer the knowledge gained from this set to accurately label an unknown dataset. Several classifier types were evaluated in the scope of this thesis. See Subsection 5.4.3 for more information.

Incremental Learning Incremental Learning is the ability of a classifier to maintain a dynamic model that can be refined with additional training data perceived during runtime. This approach is highly interesting for the project, as user feedback can be employed to improve the classifier’s behavior instantaneously. There are three

different ways how incremental learning can be realized [65]. One option is to retrain the classifier during runtime from scratch as soon as some additional training data was gathered. Therefore, quick classifier retraining has to be possible. Other options are classifiers allowing on-line learning or the use of instance-based classifiers such as *k-nearest-neighbor*.

Certainty metric A common query strategy for Active Learning is Uncertainty Sampling. The results of Lewis and Gale [52] illustrate, that this sampling approach can reduce the amount of training data requested from an user without a decrease in the classifier's effectiveness. In order to apply such a technique, a classifier is required to assign certainties to its output labels e.g. as shown in Figure 5.5.

Availability We use existing implementations of typical classifiers, as our work does not focus on the development of a classifier which is optimized for the shared autonomy approach but rather focuses on a basic evaluation of classifier types and the interaction process between user and classifier. We want to test all classifiers with the existing framework, so a `C++` implementation should be available. In this way, the use of interfaces which can decrease the framework performance, can be avoided.

5.4.3. Evaluation of Classifier Types

Besides the classifier requirements, we experiment with different types of classifiers in order to evaluate the classification performance of different types on our dataset. Therefore an image dataset containing 64 non-overlapping images with carrot and weed plants was created. The automatically extracted contours were hand-labeled as *crop*, *weed*, *multiple plants* or *other*. Afterwards the WEKA Data Mining Software [66] was utilized to run a 10-fold stratified cross-validation with varying classifiers configured with WEKA-standard-parameters.

The results of this evaluation are displayed in table 5.1.

Classifier name	Accuracy [%]
lazy.lB1 [67]	68.6 (±3.7)
lazy.lBk ($k = 5$) [67]	72.6 (±5.6)
functions.SimpleLogistic [68]	76.1 (±4.4)
functions.Logistic [69]	75.8 (±4.5)
functions.MultilayerPerceptron	75.6 (±6.0)
functions.SMO (speed up for SVM [70])	73.0 (±3.7)
bayes.NaiveBayes [71]	66.7 (±2.6)
bayes.BayesNet	68.5 (±4.2)
trees.J48 [72]	72.7 (±5.8)
meta.logitBoost (DecisionStump) [73]	76.5 (±3.9)

Table 5.1.: Classification accuracy with standard deviation for different classifiers

It becomes clear, that the choice of the classifier type does not influence the performance of the classification step statistically significantly—at least for out-of-the-box solutions. The standard deviation varies between different classifier types. An important remark is, that the final choice the classifier not only depends on its accuracy but also on its suitability for the implementation in our shared autonomy approach (compare Subsection 5.4.2).

5.4.4. Realization

We decide to use the C-Support Vector Classification of the LIBSVM library [74] in the framework. This is a multi-class implementation of a Support Vector Machine available in `C++`, which is able to assign probability estimates [75] for the possible labels of each queried instance. A Radial Basis Function is employed for the classifier's kernel. This classifier fulfills most of our requirements described in Subsection 5.4.2. Furthermore, it is shown in the evaluation of different classifier types, that this type of Support Vector Machine has a good accuracy on the plant dataset compared to other techniques.

The classification process is illustrated in Figure 5.5. It is required to detect uncertain instances for the active learning scenario. These instances are selected based on two strategies. Either a certainty threshold is applied to the class with the highest certainty or another threshold is applied on the certainty differences between the class with the highest certainty and the second highest one.

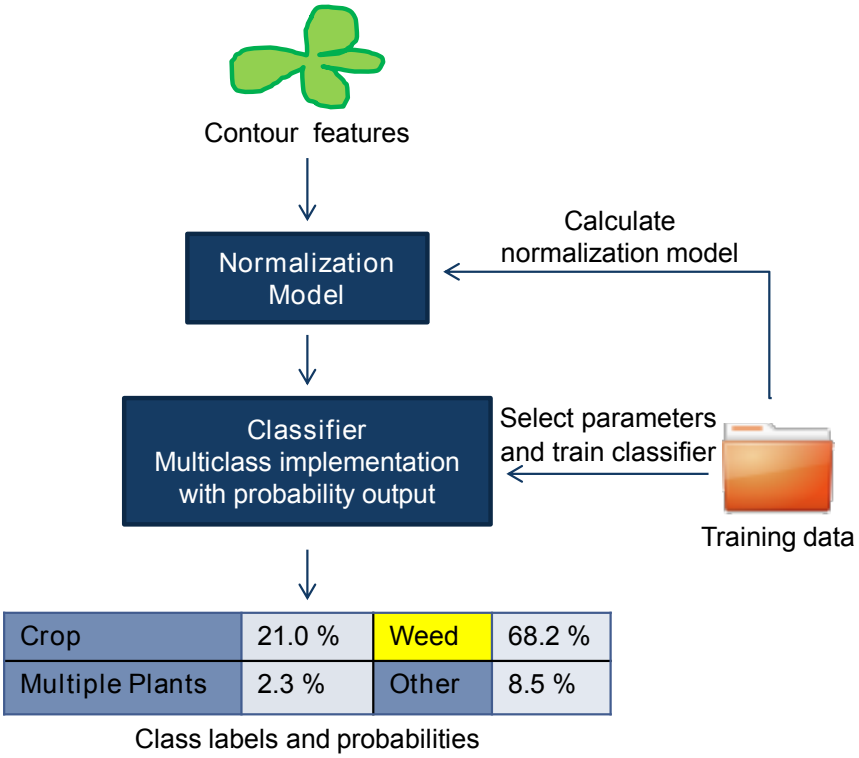


Figure 5.5.: The contour classification process

Normalization

First, a normalization model must be created, as it is important that the values for each feature are in the same numerical range to achieve the same weighting of all features. Based on a set of training instances with the feature vectors $\mathbf{u}_i \in \mathbb{R}^m, i = 1 \dots n$ the factors for linear scaling are created so that each feature value $x_{i,j}$ is in the range $I = [r_{\min}, r_{\max}]$.

$$\bar{x}_{i,j} = m_j x_{i,j} + c_j \tag{5.4}$$

$$\text{with} \quad m_j = \frac{r_{\max} - r_{\min}}{\max_{i=1\dots n}\{x_{i,j}\} - \min_{i=1\dots n}\{x_{i,j}\}} \tag{5.5}$$

$$\text{and} \quad c_j = r_{\min} - m_j \cdot \min_{i=1\dots n}\{x_{i,j}\} \tag{5.6}$$

The normalization model is applied both to the training dataset and to each queried feature vector.

Parameter selection and training

The training data is not only utilized for classifier training, but also to determine two important parameters which influence the classifier's accuracy significantly:

The soft margin parameter C defines the tolerance of a support vector machine towards classification mistakes. For large values, the SVM is more tolerant towards mistakes ("Soft-Margin SVM") and the hyperplane seperating two classes is defined by an equal or higher number of support vectors.

The factor γ scales the Radial Basis Function which is used as kernel:

$$K(\mathbf{u}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{u}_i - \mathbf{x}_j\|^2} \tag{5.7}$$

Both parameters are determined in a grid search. At first, points representing different value pairs for C and γ are defined in a grid. In the next step, a 5-fold cross-validation based on the training data determines the classifier accuracy for each grid point. Finally the results for the grid search are visualized in a contour map as can be seen in Figure 5.6 and either the parameter pair resulting in the highest accuracy is interpolated or a refined second grid search is conducted based on the results of the first one.

As soon as the parameters were determined, the whole normalized training dataset is used in order to create the classifier model.

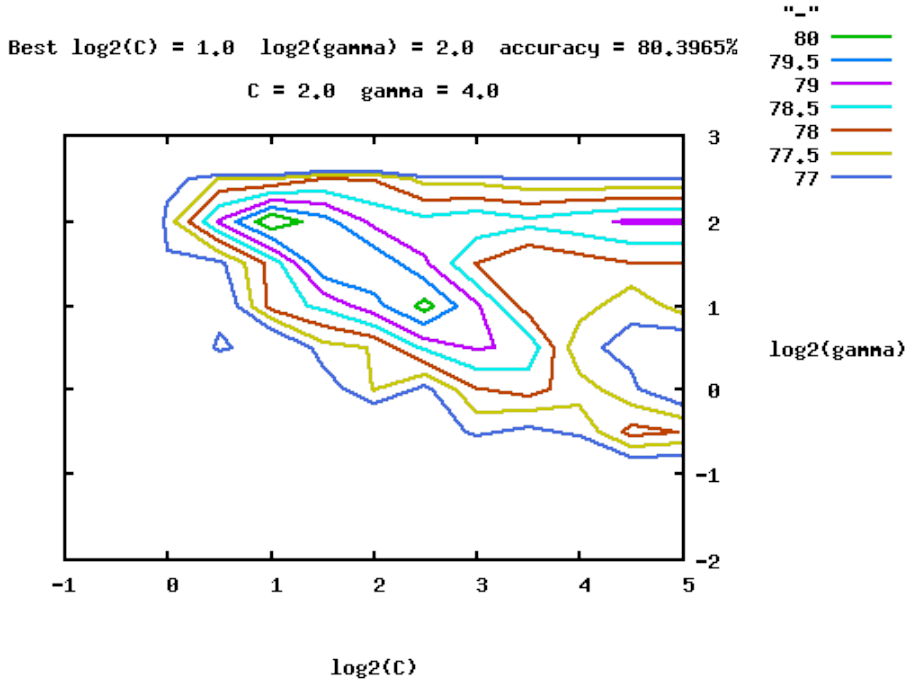


Figure 5.6.: Second iteration of a grid search for SVM parameter selection. The resulting parameters are $C = 2$ and $\gamma = 4$.

Classification

Before the feature vector $\mathbf{u}_i$ derived from a plant contour can be classified, it has to be normalized based on the existing model. Then it is sent to the classifier which outputs the class label with the highest probablitiy as well as all other class labels and their probability (see also Section 5.5).

Classification merging

The autonomous classifiers assigns all labels and class probabilities to the contours of full images. These contours are associated with unclassified tile contours (compare Section 4.2). It is necessary to determine the class label of a tile contour based on the classification results of its associated full image contours. This is done in two steps. At first, the class probabilities of the tile contour are computed as mean value of the probabilities of its associated full contours. For k classes and N associated full contours it is:

$$P_{\text{tile}}(y = i) = \frac{\sum_{n=1}^{N} P_{\text{full},i}(y = i)}{N}, \quad i = 1, \ldots, k \tag{5.8}$$

In the second step, the class with the highest probability is assigned to the tile contour.

5.5. Stem Localization

The goal of the stem localization step is to determine for every weed contour the position where the plant stem intersects with the soil surface. This location is the point of application for the weed manipulation tool of the robot. Figure 5.7 gives an overview of detected weed plants and their ground truth stem location assigned by an experienced user. It becomes clear, that the stem position within the contour has a high variation and not only depends on the plant type, but also on the growing direction of the weed.

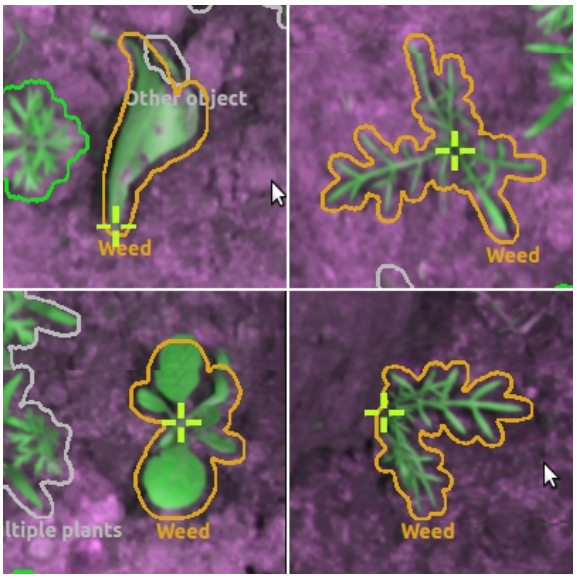


Figure 5.7.: Weed plants, their contours and user-assigned stem locations

The implementation of a sophisticated stem localization algorithm would require substantial efforts. The contours' feature values introduced in Section 5.3 are too general and do not provide enough detailed information about the plant structure. Additional values such as skeleton features [49] would be required for a machine learning approach localizing the stem position within a contour.

Therefore, we introduce a heuristic which places the marker in the centroid of each contour. This approach is straightforward and promising as most of the weed plants grow upright and their leaves are therefore distributed equally around the orthogonal stem. The stem position of a weed contour in the tile image space is determined by mapping all stem positions of associated full image contours into the tile space and calculating the average position out of these coordinates.

6. Implementation

In this chapter we explain general implementation decisions and introduce the tools and software libraries we leveraged as part of our implementation. Our goal was to use tools and existing infrastructure to quickly create a complete basic system in order to evaluate the potential of a shared autonomy concept and generate design suggestions for future systems. Furthermore, we aim to keep the variety of employed libraries as small as possible to avoid a high number of dependencies and compatibility problems.

6.1. Middleware

We use the *ROS (Robot Operating System)* [76] framework for communication and as build system. ROS provides a structured communication layer that enables to user to create encapsulated processing units, the so-called *nodes*, which are connected based on a peer-to-peer topology. There are several ways of communication between the nodes, we employ ROS *topics*, an asynchronous communication method based on standard- or user-defined message types. The ROS-version utilized for this project is *fuerte* in combination with the operating system *Ubuntu 12.04LTS*.

6.1.1. Code Structure

Different function units can be divided into *packages*. Several packages can be summed up in one *stack*. Every stack and package has the possibility to define dependencies on other stacks or packages. The complete source code for the detection framework is bundled in one ROS-stack. Table 6.1 provides an overview of the packages included. All core-components are written ROS-agnostic in `C++` and wrapped in a ROS package, so that any other communication framework could replace the current solution.

6.1.2. Advantages

The main advantages of the use of the ROS framework for this project are:

Build system ROS ships with an integrated build system. This facilitates the installation of the weed detection framework on a new machine.

Code structure The concept of ROS nodes enforces the separation of different function units. This results in a good code maintainability including the possibility to quickly exchange complete nodes with minimum effort.

Package name	Description
rf_detection_pipeline	The basic weed detection pipeline. It receives images from the robot, processes them and sends them to the GUI for user feedback. This additional input is merged with the autonomous detection data and used to determine the stem positions of weed plants which are published back to the robot.
rf_detection_gui	The GUI receives images with labeled or unlabeled plant contours and/or stem markers from the pipeline, displays them to the user and offers modification tools. After this, the data is published back to the pipeline. Two basic interfaces are available. One is for scrolling and one for still image display.

Table 6.1.: Overview of the main ROS-packages contained in the project stack

Communication It is important to consider that the robot and the Remote Farmer are locally separated and it might even be interesting to outsource the processing extensive image mosaicing to an additional external computer. Therefore, the communication framework has to provide the possibility for inter-platform communication. The concept of ROS topics enables us to use existing ethernet connections in order to run several nodes on different machines. Figure 6.1 is an overview of the network topology of the weed detection pipeline during runtime.

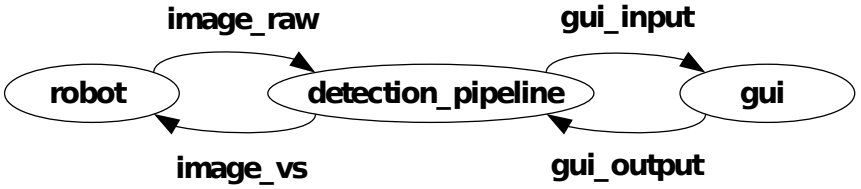


Figure 6.1.: Network topology of the weed detection pipeline. The ellipses are ROS nodes which can be run on different machines. They communicate by ROS topics.

Dependencies The ROS framework supports a high number of libraries typically used for robotic applications. Existing stacks and packages facilitate the integration of system dependencies such as *OpenCV* or *Qt* and ensure high code reusability.

Debug utilities ROS provides a number of debug utilities such as tools to detect network communication problems or to record and replay messages sent during runtime.

Project compatibility As ROS is not only used for the image processing pipeline, but also for functions of the remote farming robot such as navigation or image transportation, the interfaces to the weed detection pipeline can be easily accessed.

6.2. Image Processing

Most of the image processing is done with the *OpenCV* [77] library. "OpenCV is an open-source, computer-vision library for extracting and processing meaningful data from images" (Ibid.). It not only offers a good choice of state-of-the-art image processing algorithms, a `C++` interface and extensive library documentation but is also able to handle large amounts of image data effectively, for example by zero copy passing of data between different objects

or by offering low level C-style access for pixel-wise operations. OpenCV is employed for all image processing steps except of contour classification (see Subsection 5.4.4 for classifier implementation).

There is no central storage for full- and tile image objects. Instead, those objects are connected in the style of a linked list as it is important to maintain the sequential relationship of the data. Figure 6.2 shows, how different image objects are associated. There is no direct connection between neighboring tile images, as this association is indirectly established over their corresponding full images. In case it is not possible to match two neighboring full images, the connection between the full image objects is created anyway, but a flag is set which indicates the missing visual association. The tile image data of the full image which could not be matched with its predecessor equals the full image data itself as it could not be fitted to the existing tilestream.

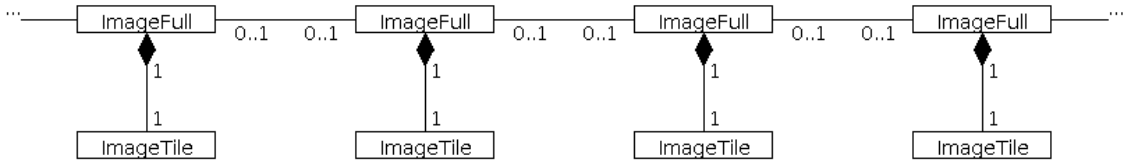


Figure 6.2.: Associations between full- and tile images

6.3. Graphical User Interface

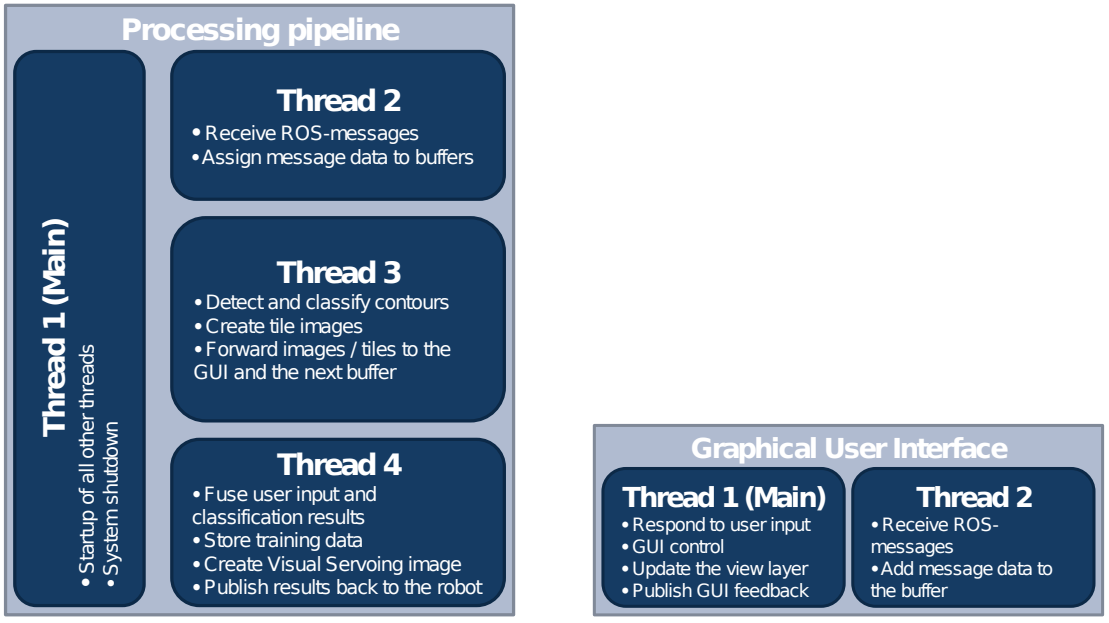
The main challenge regarding the graphical user interface is displaying the scrolling image stream composed out of several tile images with an abstract overlay containing high-level information such as detected plant contours and interaction requests. As the final GUI is wrapped in a ROS-node, an easy integration into the ROS build- and communication system has to be possible.

The cross-plattform framework Qt is widely used in ROS applications such as `rviz` and `rqt`. Furthermore, the `QGraphicsScene` provides tools to overlay the animated user output with additional layers that can respond to user input events. The platform independency of Qt would theoretically even allow to use the GUI in combination with MS Windows, an operating system that is probably more familiar to a typical Remote Farmer than the Linux environment. Yet there is currently no stable ROS implementation for Windows, so the communication system would have to be changed.

6.4. Thread Handling

In the image processing pipeline some computationally expensive operations such as the matching of feature points for image mosaicing (compare Subsection 4.4.2) are executed. Furthermore, some events such as the reception of new image data from the robot or the reception of user feedback are triggered externally in a spontaneous fashion. In order to make as well the pipeline and the graphical user interface responsive to such events and to avoid processing intensive operations from blocking the whole data flow, we implement pipeline and GUI in a multithreaded fashion.

Figure 6.3 provides an overview of the different threads running in the system. The threading architecture is implemented employing the *Boost.Threads* library [78] in the processing pipeline and the Qt multithreading functionality for the graphical user interface.



(a) Processing pipeline (b) Graphical user interface

Figure 6.3.: Overview of the threads and their tasks running in the processing pipeline and the GUI

7. Evaluation and Analysis

In the first section of this chapter the influence of the tile approach on the pipeline performance is evaluated. After this, we introduce an user study with the goal to evaluate five different concepts for user involvement in the weed detection process. Finally, we evaluate the impact of the shared autonomy approach for our overall system.

7.1. Tile Approach

In this section, we evaluate the effects of the tile approach on our system.

7.1.1. Accuracy of Marker Placement

It is important to examine the decrease in accuracy due to the uncertainties introduced by the image stitching approach. In a basic shared autonomy scenario, the user marks the stem position of weed plants in a floating image stream which was created by the image processing pipeline. These tile marker positions are then mapped back to the original full images which are returned to the robot for visual servoing based manipulation of the detected weed plants.

The goal of this evaluation is to test the tile-to-full-image mapping functionality and to determine the accuracy-loss during the mapping of marker positions. In a first step, five different datasets each containing 15 overlapping images are selected. The ground truth is created by an expert user who places weed stem markers in the full images. Next, the same expert user marks the weed stems in the five tile streams created out of the datasets. For every tile image, all markers are mapped back to every full image in which they appear. The position of these markers are compared to the ground truth stem markers, whereupon a mapped marker was considered as 'hit' when its distance to the next ground truth marker that has not been hit so far is below a threshold. Several thresholds are applied and for each threshold, the precision, recall and the $F_{0.5}$-score measured. For our evaluations, we prefer the $F_{0.5}$-score (see Chinchor [79]) over the F_1-score, as we want to emphasize the importance of precision. In the context of our project, the a false-positive weed plant detection is worse than an undetected weed (false-negative), as a possibly destroyed crop weights more than an undestroyed weed. An example for the distance tolerance can be found in Figure 7.1.

The results as shown in Figure 7.2 underline that the mapping seems to work reliably, if a position tolerance radius of 25 pixels can be accepted. As no camera calibration

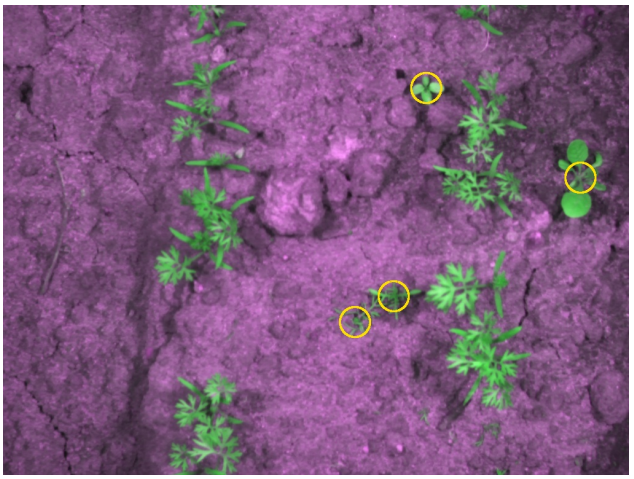


Figure 7.1.: Example for a distance threshold of $r = 25\,\text{px}$

information and no description of the camera position relative to the field in the test setup is available, the real-world distances can only be roughly estimated based on camera images with a measuring tape lying on the soil. The 25 px tolerance corresponds to a value of 4.5 mm.

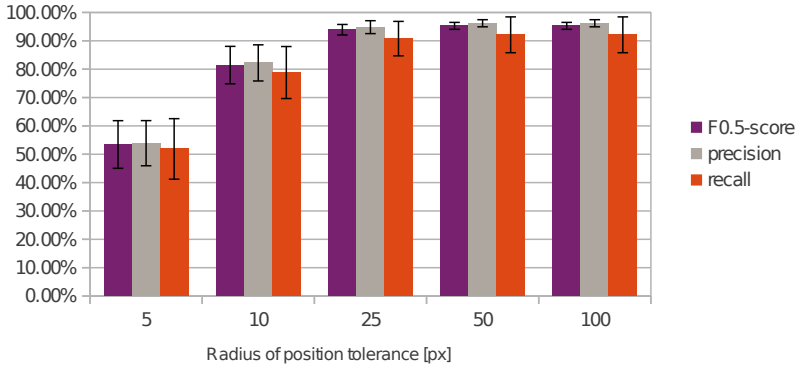


Figure 7.2.: $F_{0.5}$-score, precision and recall of the stem marker mapping for different distance thresholds

An error of around 8 % in the recall and 4 % in the precision remains even for extremely high position radius tolerances. The main source of this difference cannot be uncertainties in the image mosaicing process itself—these should also result in an equal precision decrease. Rather this phenomenon can be ascribed to the expert user who is proficient in marking detected weed plants correctly (high precision) but tends to miss some of them. This effect is amplified for missed weed plants in the tile image stream, as one tile marker usually results in several full image markers due to overlapping. If we assume the user missed the same percentage of weed plants in the full images and the tile image stream, we can conclude that the decrease in the recall must be higher than the decrease in the precision.

It is unclear, to which parts the missing 4 % of precision refer to a mapping error and to which parts they refer to missed weed plants in the ground truth data. In general, an inaccurate placement of markers by the user seems to be neglectable for higher precision tolerances, as the same user marked both the ground truth and the image stream dataset and it seems implausible that an experienced user varies a marker position for more than 5 pixels.

To sum it up, the image mosaicing introduces mapping uncertainties that can be accepted, if a position tolerance of is allowed. This value depends on the final choice of the weed

manipulation method. It is important to keep in mind that the evaluated phenomenon appears in every scenario, where stem positions in the tile image stream are set by the user or created by an autonomous algorithm and adds to the total error estimation.

There are several approaches to reduce the mapping uncertainty:

- Improve the image mosaicing algorithm e.g. by obtaining depth information in the images.
- Do not work with stem positions in the tile image stream.
- Introduce a second region-restricted image association for every stem position in the tile image stream. This is an algorithm that fits the image regions around tile stem positions with the expected region in the full image and uses the information gained for an improved position mapping.

7.1.2. Processing Efficiency

We want to show three main advantages of the tile approach: The reduction of the clickload on the user and the decrease of processing time and transmission data. Therefore we need an estimation of the length of a tile image stream first.

Length of the tile image stream

We want to derive a formula which enables us to approximate the total length of a tile image stream based on a given number of full overlapping images. Therefore, some assumptions have to be introduced:

- All full images have the same size $h \cdot w$
- The centroids of the fitted images are positioned on a vertical line.
- The only transformation required for fitting two neighbor images is a translation in the image plane.
- The overlap factor between every pair of neighbor images is constant.

Based on the measures defined in Figure 7.3 we define the overlap factor as followed:

$$k = \frac{l_0}{h} \tag{7.1}$$

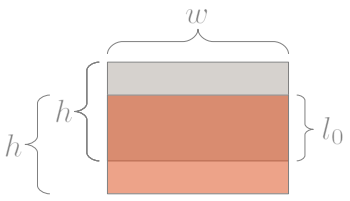


Figure 7.3.: Two overlapping images

The resulting function for the length of the tile stream for n full images is:

$$l(n) = \big(n - (n-1)k\big)h \qquad \forall n \in \mathbb{N}_{\geq 1} \tag{7.2}$$

Proof. We can show this with mathematical induction:

Base case: Clearly $l(1) = h$.

Inductive step: We want to show that if $l(n)$ is true for a $n \in \mathbb{N}_{\geq 1}$, then so is $l(n+1)$:

$$\begin{aligned} l(n+1) &= l(n) + h - l_0 = l(n) + (1-k)h = \big(n - (n-1)k\big)h + (1-k)h \\ &= \Big((n+1) - \big((n+1)-1\big)k\Big)h = l(n+1) \end{aligned} \tag{7.3}$$

□

Clickload reduction

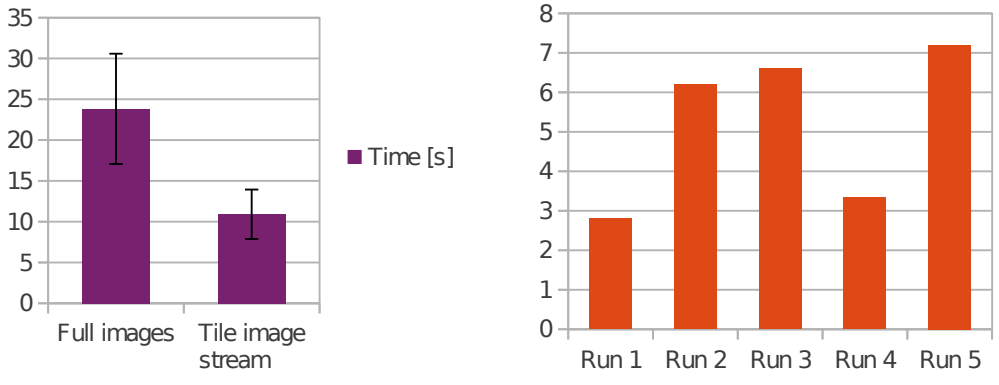
We expect that the tile approach reduces the user interaction time significantly. The overlapping of full images results in some of the tile markers being mapped back to several images. This reduces the clickload on the user, as one click per weed plant is required and it is avoided that the same plant appears in several images and therefore has to be marked multiple times.

Based on the formula derived in 7.1.2 we estimate the expected decline: In a first step, the k-factor for image overlap has to be found. From the total length of the tile streams of the five datasets for the mapping test is the average value and its standard deviation calculated: $k = 0.32(\pm 0.0098)$. The weed density in the full image stream and in the tile image stream can be assumed to be identical. With this information the ratio between the amount of tile markers and full image markers is calculated. It equals the ratio of the length of the tile image stream to the sum of the length of all full images:

$$x = \frac{l_{\text{tile}}}{l_{\text{full}}} = \frac{\big(n - (n-1)k\big)h}{nh} = 1 - (1 - \frac{1}{n})k \approx 0.70 \tag{7.4}$$

Reduction of processing time

The time the user took to process the five datasets was measured in the study described above. The results can be found in Figure 7.4a. The most interesting observation is that the processing time for the tile image streams is not, as expected, reduced by 30 % but rather decreases by 54 % — the user takes less time to mark a weed plant when working on the tile data.



(a) Processing time per image

(b) Weed ground truth markers per image

Figure 7.4.: Evaluation results of the tile image approach

We ascribe this effect to the better scene context of the tile image stream: While the user has to search every full image completely for weed plants, he only has to scan the upper region of the currently scrolling in tile. This effect shows, that the tile approach is not only more efficient because it condenses the feedback required from the user but also, because

the data is presented in a way to the user that enables quicker interaction. This effect is expected to be independent on the user interaction scenario and we expect the same results could be achieved for the user labeling plant contours instead of placing markers.

As seen in Figure 7.4b is the number of weed plants highly dependent on the chosen image datasets and varies between the test runs. This results in the high variance of the processing times.

Reduction of transmission data

It is crucial to ensure a good communication between robot and Remote Farmer especially for an online shared autonomy scenario, where the robot requests information from the user during his field activity. As the bandwidth of this connection may be limited, the amount of data transmitted from the robot to the user and back has to be kept to a minimum.

One of the most data-intensive tasks is sending the field images from the robot to the user. Therefore it is desired to condense all full images in a way such that the user still has all necessary information, but the transmission of redundant data is avoided. This is exactly what is pursued in the tile image approach. If we assume the image transmission data is proportional to the image area, Equation 7.4 enables us to estimate the ratio between tile image data and full image data for a given amount of images and a given overlapping factor.

Future implementations of the pipeline might go other ways in order to reduce transmission data. For example, it is possible to send only cropped images to the user in case that his feedback is required for an instance. However, we want the user to be able to observe all classifier decisions and therefore this is not an option.

7.2. Evaluation of Shared Autonomy Approaches

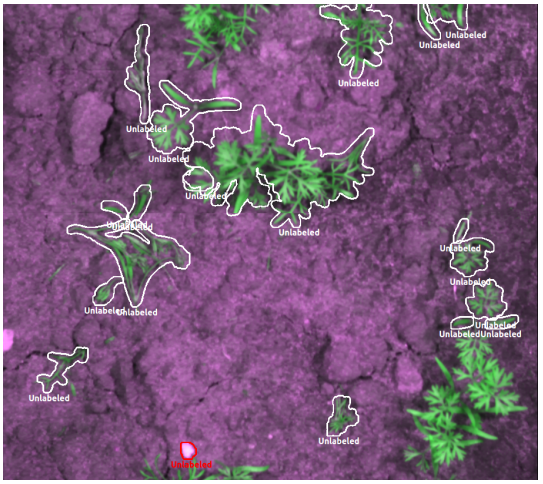
We evaluate different approaches for the integration of the user in the weed detection pipeline and the interaction between user and classifier. The goal of this study is to determine the best scenario for the human-machine interaction and to identify problems arising from the shared autonomy concept.

7.2.1. User Study Setup

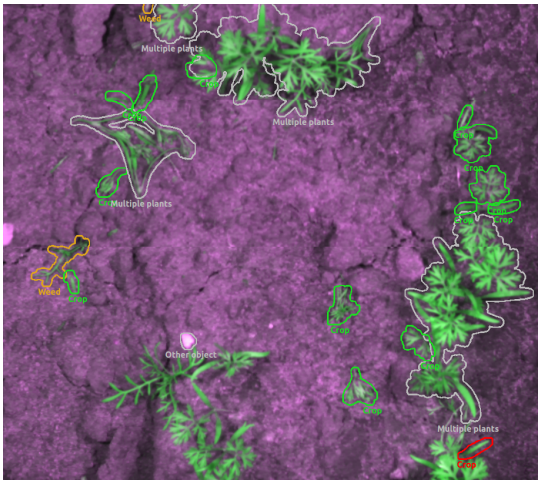
We develop five different scenarios for pipeline-user interaction. They are introduced in Figure 7.5. All of the five scenarios depend on the GUI with scrolling tile images created out of 15 full original images. In the simplest approach, the autonomous pipeline does only create the tile image stream and the user places stem markers in this stream which are mapped back to the original images. Other scenarios with a higher level of autonomy aim at detecting plant contours and classifying those. The labeled contours are checked by the user and finally one stem marker per weed contour is created.

Scenario 1 The segmentation results are presented to the user in the form of unlabeled plant contours. The user has to assign a class label to each contour and cannot place any stem markers. This scenario does not involve any classifier and can also be seen as the training data generation step for an untrained classifier.

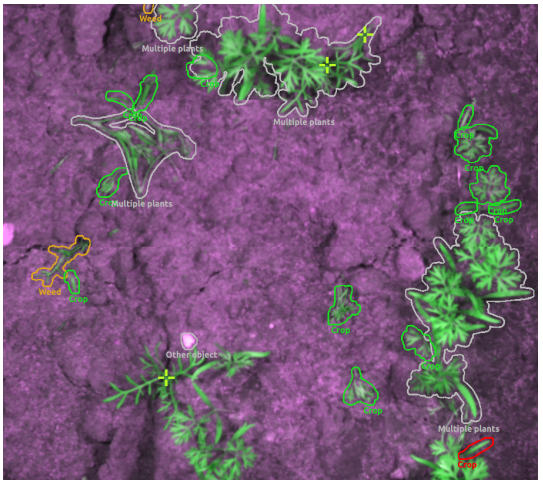
Scenario 2 Every contour displayed in the GUI has a plant label assigned by the classifier. Uncertainties in the classification process are not considered at all. The user has to check the assigned labels and correct them whenever he detects mistakes. The comparison with *Scenario 1* enables us to evaluate how strong the user is influenced by the classifier and how the introduction of a classifier affects the system behavior.



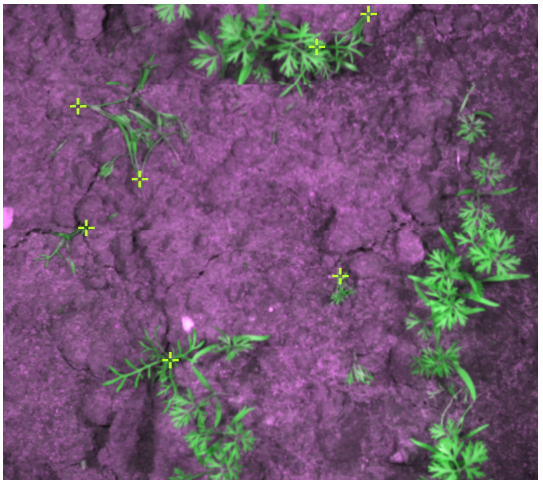
(a) **Scenario 1:** All detected plant contours are marked as *unlabeled* (white contours) and presented to the user, who is asked to assign a label (compare Subsection 5.4.1) to each one.



(b) **Scenario 2:** The detected plant contours are additionally pre-labeled by a classifier trained with 821 instances of plant contours created out of a different dataset. The user has to check the labels assigned by the classifier and correct wrong ones.

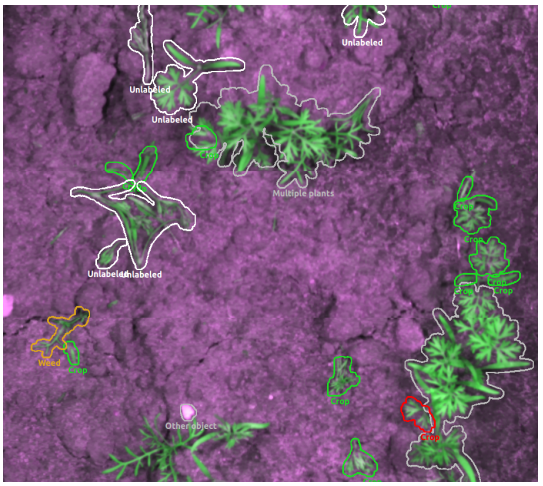


(c) **Scenario 3:** Additionally to the functionality of scenario 2, the user has the possibility to place markers on the stem of weed plants which were either not segmented by the classifier or are part of a segmented *multiple plants*-object.



(d) **Scenario 4:** A plain image stream without any high-level information such as contours or labels is displayed. The user has the task to place all stem markers manually by mouse-clicks on the stem positions of weed plants.

Figure 7.5.: The first four different types of interfaces evaluated in the user study



(e) **Scenario 5:** A filter is applied to segmented and pre-labeled plant contours. This filter calculates for each contour the difference between the class with the highest and second highest probability as assigned by the classifier. If the difference is below a threshold of 20 % the associated contour is marked as *unlabeled*. The user is asked to label only all of those filtered contours, but leave the other labels unchanged.

Figure 7.5.: The fifth interface evaluated in the user study

Scenario 3 Additional to the setup of *Scenario 2* the user is provided with the opportunity to place weed stem markers in the scrolling field view. He is asked to place a stem marker for every weed plant which is not segmented correctly for example because of undersegmentation.

Scenario 4 In this scenario, the whole abstraction layer created out of segmented contours and their class labels is abolished. Instead, the user is asked to place a stem marker for every weed plant in the image. This scenario allows us comparing the high-level interaction approaches against human-machine interaction on a lower level.

Scenario 5 This scenario is a combination of *Scenario 1* and *Scenario 2*. A selective querying strategy is introduced as described in Subsection 3.2.2. All contour labels are assigned by the classifier and presented to the user except of contours where the probability difference between the class with the highest and second-highest value is below 20 %. These contours are marked as *unlabeled*. The user is asked to assign a class label to all unlabeled contours, but leave all other contours with their classifier-assigned label.

It remains to compare these different approaches against each other. We are not only interested in the weed detection performance, but also in the user experience. Therefore, five inexperienced users, 1 female and 4 male adults aged between 20 and 29 years, were asked to support the weed detection system by interacting with the five different types of interfaces.

In a first step, the users were introduced to the project and taught how to distinguish between different plant types. After this, they tested the different interface functionalities in an introduction run. Next, the user processed the data with each of the different interfaces described above. The interfaces were presented to the users in random order. Not only the input perceived from the users was logged, but also the processing time per interface and the amount of input events (mouse-clicks, key-presses). After every run, the

users were given a quick questionnaire and asked to evaluate the interface from different point of views. The questionnaire can be found in the appendix in Section B. Finally, after all of the five interfaces were processed, they were asked to sum up their experiences in an evaluation sheet.

The results of this user study should be considered preliminary since the subjects were mainly young people educated in computer technologies and may not be representative of the general population. However, the results show general trends which are valuable for future design decisions.

7.2.2. Weed Detection Performance

Processing time

Figure 7.6 displays the average processing times for the different interfaces. It becomes clear, that the "intelligent" approaches required more user time than the simple markers-only interface. The active learning filter interface required the least user time since users only had to lane a limited number of contours marked as unlabeled due to classifier uncertainty. This shows, that an intelligent interface can improve the performance in terms of processing time. However, the accuracy of this approach still has to be evaluated. Pre-labeled contours help the user to fulfill his task quicker compared to the interface which presents the user unlabeled contours only.

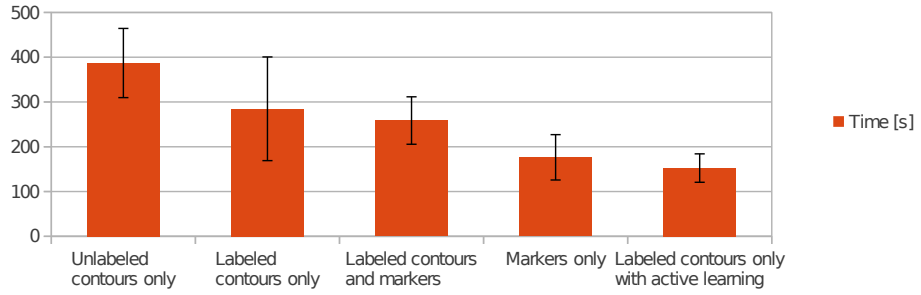


Figure 7.6.: Average processing times and standard deviation of the different interfaces

In general, it takes an inexperienced user around 150 seconds to process the 15 images for the quickest interface. The total length of the image stream of around 7900 px corresponds to an estimated real-world distance of 1381 mm. Based on this data we can calculate the length of a field section the user can process per second: 9.2 mm. As the target driving speed of the weed manipulation robot is 50 $\frac{\text{mm}}{\text{s}}$ [3], an online-processing of the data seems not to be feasible even under these best-case assumptions. There are several approaches how this challenge can be tackled:

Stop-and-go The robot stops while the user processes the data and continues as soon as the user feedback on the current field region was received and all weed manipulation tasks were done or it can be ensured that they can be finished while the robot is driving on.

Parallel processing The data collected from the robot is distributed to several users. This scenario is challenging in terms that the single user tasks have to be assigned sophisticatedly under the consideration of transmission and user reaction times.

Others There is a high number of alternative solutions. They include reduction of the robot speed, improved user training so that he can process the data quicker or a realization with a higher degree of autonomy that requests less data from the user or is not directly dependent on the user's feedback but rather only utilizes it in a long-term learning process.

The fully autonomous weed detection takes around 19 s for the same amount of images (compare Section 7.3). This results in possible driving speeds up to 73 $\frac{\text{mm}}{\text{s}}$, which is acceptable. We will examine the impact of the shared autonomy approach in detail later.

Input events

Not only the processing time but also the amount of user input events are recorded during each testrun. An user required one right-click and one left-click to assign a new label to a contour and one left-click in order to place a marker. The key-presses refer to pause- and unpause-events as the user had the possibility to stop the scrolling image stream by pressing the space-key.

A look at the amount of user input events (Figure 7.7) shows a strong correlation to the processing times. This indicates, that it is essential for a quick processing of the data to restrict the required user input to a minimum.

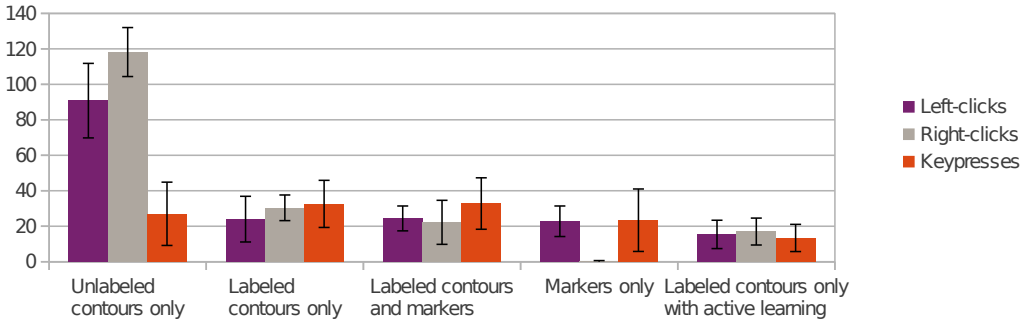


Figure 7.7.: Average user input events and standard deviation

Stem detection performance

The most important metric for comparison of the different user interaction approaches are the final detection rates for weed stem positions in the full images. This is the output of the image processing pipeline that is sent back to the robot. The stem positions detected in every run of the user study are compared against a ground truth. Figure 7.8 shows the resulting detection rates. As all interfaces are based on scrolling tiles, the results of the marker mapping evaluation from 7.1.1 can be considered to be the best values achievable. We use a marker position tolerance of $r = 25\,\text{px}$, as we could show that this value yields to a good balance between avoidance of mapping errors and precisely manipulated weed plants.

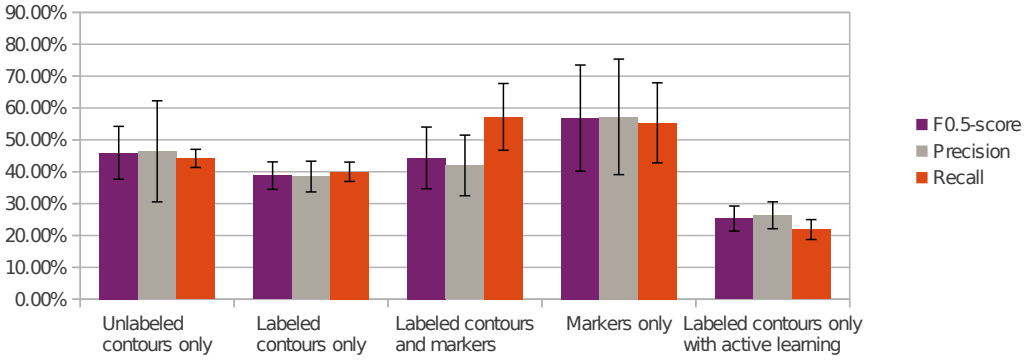


Figure 7.8.: $F_{0.5}$-score, precision and recall of the weed stem detection with different interfaces for a distance threshold of 25 px

The interface where the user only has to place markers directly yields the best results, because many error sources can be excluded from the weed detection process. However, an average $F_{0.5}$-score not higher than 56% can be achieved. This is extremely low compared to the performance of an expert user of around 94 % (Figure 7.2) and underlines, that the task of the Remote Farmer cannot be considered as a typical crowdsourcing application. It requires a reasonable training to be able to detect all weed plants correctly. The user introduction at the beginning of the study seems to be insufficient for a good performance. The noticeably high standard deviation for all three different metrics shows, that the variation between the different users was extremely high. In general, the standard deviation can be seen as a measure of user uncertainty.

This uncertainty can be reduced by introducing the abstraction layer. When the users are given unlabeled contours only the recall values for all five participants are between 40 % and 50 %. However, the standard deviation for the precision is roughly the same value as before—some user's seem to label a high amount of non-weed plants as weed. In general, the detection rates of this interface are around 10 % lower compared to the markers-only approach. This seems to be a result of the plant contours, which restrict the user, as a proper plant segmentation and marker extraction is assumed for good performance. As shown in the user experience evaluation (Figure 7.12), this effect is also noticed by the user.

The introduction of the classifier which produces prelabeled contours that only have to be verified by the user can compensate user uncertainties, but also results in a slightly worse detection performance. A fully autonomous classifier performs significantly worse (compare Section 7.3). The user tends to trust the classifiers choice and therefore does not correct all classification mistakes. The performance decrease is in contrast to a much better average processing time (compare to Figure 7.6).

The interface with combined contour labeling and and markers increases the recall remarkably to the level of the markers-only interface. This is due to the gained user flexiblity—finally the user has a tool to mark weed plants that are not correctly segmented by the classifier, e.g. because they belong to a *multiple-plants* object. However, the precision remains on a low level. Probably two factors yield to this bad performance: The extraction of a stem position for weed contours has a low precision and the user tended to label non-weed plants as weed.

The active learning approach performs poorly. The main problem here is a high amount of classifier mistakes that cannot be corrected by the user, because the active learning filter considers these wrong labels as 'certain'. Either the certainty filtering threshold was set too low or the probabilistic model in the SVM classifier does not perform in the expected way.

In total, we could show that a good integration of the user into the autonomous weed detection process decreases the user's processing time and increases the system overall performance. It is hard for high-level interfaces to compete against less sophisticated user integration concepts in the terms of processing time and overall weed stem position detection rate. A selective user input querying strategy can result in significantly decreased processing times, however it also decreased the detection performance in our approach noticeably.

7.2.3. User Experience

The user experience is evaluated based on a questionnaire each user has to file after he is finished working with one interface and on a final evaluation sheet in which the user is asked to sum up his experiences with the different human-robot interaction scenarios.

All users confirm, that every interface is easy to understand (Figure 7.9) and the additional data such as contours or contour labels provided in some of the scenarios is helpful (Figure 7.10). The test group is split in the question, whether their task was easy or hard to fulfill. This underlines the observation during the different testruns, that some users had problems to distinguish between the different plant types even after the introduction.

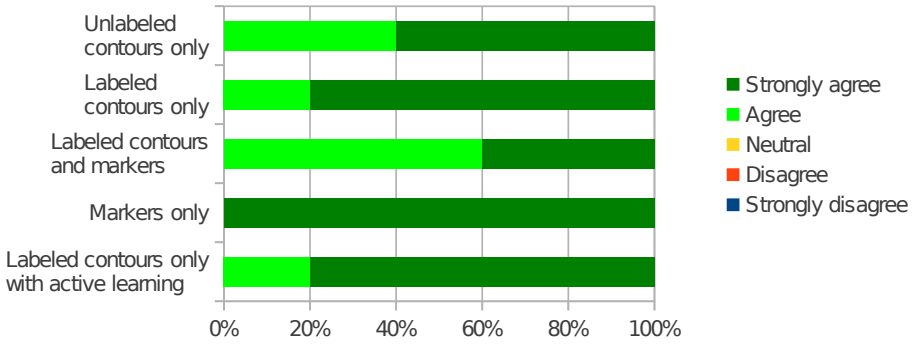


Figure 7.9.: All interfaces are easy to understand

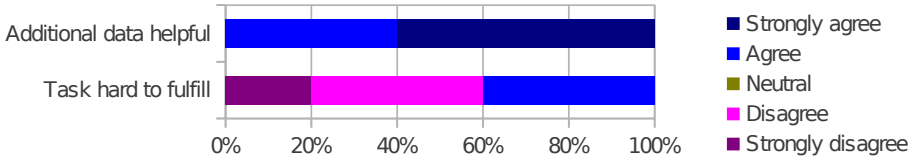


Figure 7.10.: General questions

It can be seen in Figure 7.11 that the users consider the first and the third interface partially as stressful to handle. There seem to be different reasons for this. In case of the first interface, a high amount of clickwork has to be done (Figure 7.7). Interface 4 is perceived as stressful, because multiple tasks have to fulfilled: The user has to check contour labels and place markers additionally. On the other hand, Figure 7.12 illustrates, that the combination of contour and marker interface enabled the user to provide feedback to the robot in a satisfying way. All other interfaces were rated worse in terms of flexibility. Some users seem to desire an interface that provides them with other improved methods for providing feedback. Especially interface 5 restricted the users noticeably, as they could see wrongly labeled contours, but were not allowed to correct them because they were supposed only to assign labels to unlabeled contours.

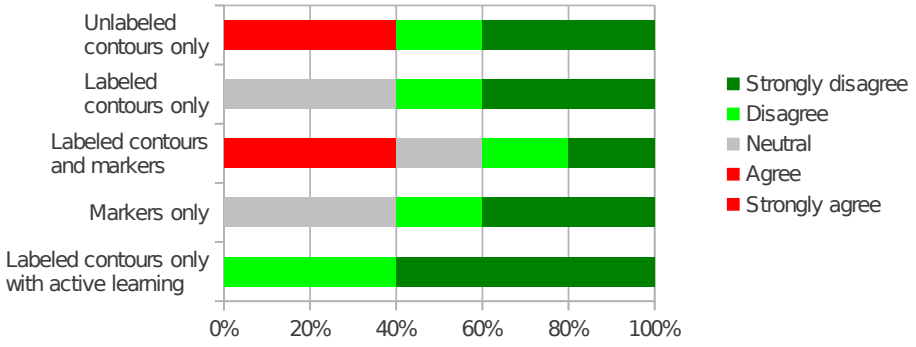


Figure 7.11.: Whether user consider the interface to be stressful to use

7.3. Impact of the Shared Autonomy Approach

In our user study, we evaluated different human-machine interaction concepts with varying levels of autonomy. Now, we want to determine the influence of the user on the performance

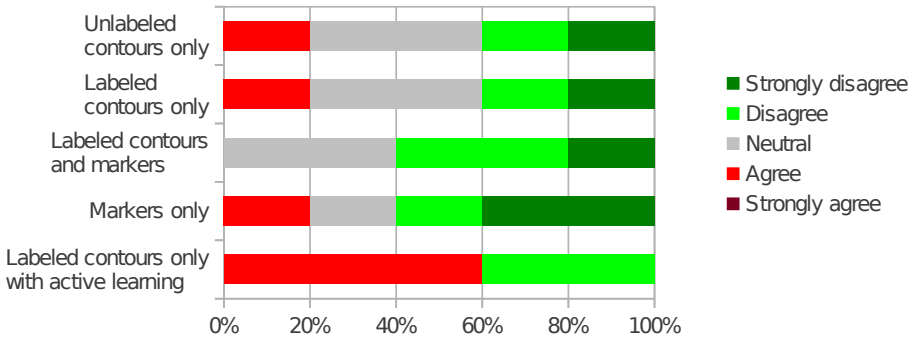


Figure 7.12.: Users' answers to the question, whether they felt restricted in providing feedback to the robot

of the overall system. Therefore, we incorporate the results of *Scenario 2* in the user study, where all plant contours are prelabeled by the classifier and the user can change contour labels, but is not allowed to place additional stem markers. We compare the performance of the five inexperienced users involved in the study against a fully autonomous approach. In this scenario the labels of the classifier are directly used for the stem localization step without user interaction. The results for this configuration are maintained as the average values out of five runs as well. Additionally, we add a third scenario for comparison: An expert user takes over the label checking and correction task. As only one expert user was available, we were not able to repeat this test for several times and therefore provide his results for one run and without standard deviation.

In Figure 7.13 the $F_{0.5}$-score, precision and accuracy are displayed for the three different configurations. Again, we employ a distance threshold of 25 px to determine, whether a resulting stem marker is positioned correctly. It becomes clear that even the integration of an inexperienced human user is able to double the detection rates. The classifier seems to have problems distinguishing between weed and non-weed plants. This is an interesting observation as we could show in a preliminary classifier evaluation presented in Subsection 5.4.3 that our choice of feature values and classifier types results in classification accuracies of around 73 % without any parameter optimization. However, the accuracy metric is good to estimate the overall classifier performance on a multiclass problem, but not for this use case, where we are particularly interested in a binary classification problem and have a biased class distribution. Out of the confusion matrix for our preliminary results, we can calculate the SVM performance for weed detection: A precision of 82 % and a recall value of 38 %. It becomes clear that the SVM classifier has problems when it comes to identifying all existing weed plants in a dataset correctly. This explains to some extent the problems in our classifier-only testruns. The most important additional influencing factors are:

- The employed implementation of the SVM classifier and the plant dataset differs between the two evaluations.
- In the user study, the classifier is applied on full image contours. Later the labels and the class probabilities of one or more full image contours are merged for the classification of tile image contours, which are the base for the stem localization (compare Subsection 5.4.4). This merging introduces additional uncertainties.

The recall of the inexperienced and the expert user are on the same level. If we consider the results of the expert user as ground truth, we can conclude that also an inexperienced user is able to determine all contours of weed plants. However, the limitation of stem markers only derived from centroids of contours in this scenario prevents the users to

achieve higher recall values than 40 %. The increased precision of the expert user indicates that inexperienced users produced some false positives—non-weed plants labeled as weed. It is interesting to note that the standard deviation of the inexperienced users and the classifier runs are in the same range. As identical training data was used for all classifier-only runs, the classifier's performance seems to vary because of probabilistic deterministic behavior which is introduced at several points in the autonomous weed detection pipeline such as the image matching with support of a RANSAC filter or the classifier training.

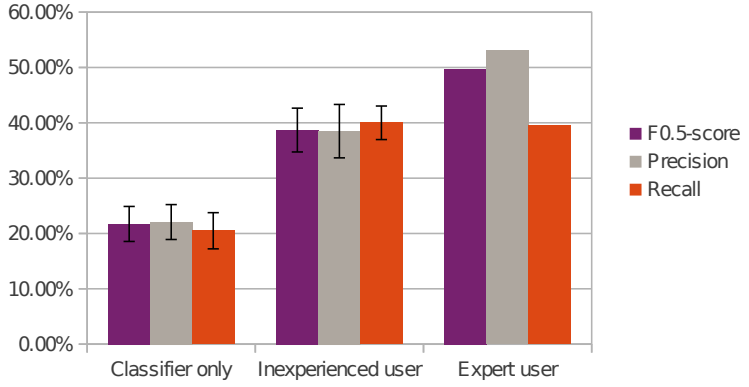


Figure 7.13.: $F_{0.5}$-score, precision and recall of the weed stem detection without any users and with two different user types for a distance threshold of 25 px

Finally, we compare the processing times for the configurations without any users and with inexperienced users in Figure 7.14. It is obvious that the user integration increases the processing time for our 15 overlapping images significantly by a factor of 15. This can be accepted for an offline scenario, where the user works on data previously recorded by the robot, but is critical for online weed control.

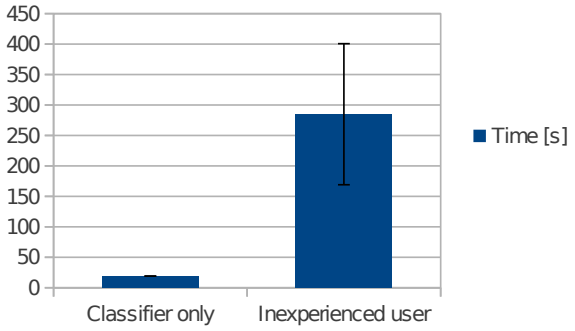


Figure 7.14.: Processing times for 15 images of the fully autonomous approach and with an user integrated into the weed detection process

In this chapter, we evaluated the performance of our overall system. We demonstrated that the tile approach introduces small uncertainties, if a distinct position tolerance can be accepted. Furthermore, it reduces the load on the user, his processing time and the transmission data significantly depending on the overlapping factor of the full images. We examined our system in an user study with users that worked on five different interfaces with varying levels of autonomy. It turned out that it is difficult for a system with a high level of autonomy to compete against less sophisticated integration solutions of the human user due to the complexity of the weed detection process. More complicated interfaces were considered as stressful by the users, but they realized that such approaches provide additional flexibility for robot feedback. Finally we compared our shared autonomy approach to a completely autonomous system which revealed a significant performance improvement by the integration of the user, but also an noticeable increase of the total processing time.

8. Summary and Future Work

8.1. Summary

In this thesis, we describe the development, implementation and evaluation of a shared autonomy weed detection framework which is designed for the integration into an agricultural robot. The goal of our system is to receive overlapping images that are created while a robot is driving over a field, process these images and detect the point of application for weed plants, the so-called *stem markers*. These points are employed by a manipulator for active weed control.

We inserted a human user into our processing pipeline to increase the robustness, reliability and performance of the overall system. Therefore, we developed a framework which is designed in an user-centered manner. For example, we employed plant contours as an abstract unit that can be easily handled by object recognition algorithms as well as provide good visualization and interaction possibilities when displayed in an user interface. As our focus was the evaluation of different interaction scenarios between user and system, we used out-of-the-box solutions for the autonomous data processing, analyzed different human integration concepts and finally evaluated a choice of five interfaces in an user study.

First, we examined different points of application for our shared autonomy concept based on the functionality of a fully autonomous weed detection system. We determined three possible different human insertion points, the *segmentation*, the *classification* and the *stem localization* step. Our implementation involves the user in the plant classification step, as user feedback for autonomous classification results can be provided easily and quickly and our approach for shared autonomy classification can be transferred to other use cases. Next, we introduced different user-classifier interaction scenarios. We finally integrated the user in such a way that he can check and correct a filtered amount of instances of data already processed by an autonomous classification algorithm. Different concepts for graphical user interfaces were suggested and we finally implemented one flexible solution. In this GUI we added the optional possibility for the user to mark stem positions directly in order to overcome segmentation failures. The ROS framework was employed as build system and for the communication between the user and the autonomous processing pipeline.

We introduced the concept of an image tiling technique. Created tiles can be assembled to an image stream with two main goals: To create associations between the overlapping images received at the beginning of the pipeline and to provide an intuitive visualization of

the stream of overlapping images collected by the robot and therefore improve the system-user collaboration. The scrolling stream of tile images is displayed in the GUI and serves as base for all user interaction requests. Out of the overlapping full images we created an image mosaic with homographic transformations by matching received images with a feature point based algorithm. As our image stream can be, contrary to conventional panorama pictures, indefinitely long in one dimension, we introduced a deskewing step which avoids singularities in the image transformation.

In Chapter 5 is the autonomous portion of the image processing described. We developed a custom algorithm for the subtraction of the NIR and red image channel in order to improve the contrast between plants and background. A non parametric, marker-based watershed algorithm is used to detect plant contours. Out of these contours, 20 shape- and 2 texture-based features are extracted. Based on the normalized features and a set of training data, the implementation of a support vector machine classifier assigns a class label and class probabilities to all plant contours. This information is the initial point for the user interaction. Either the user checks and corrects all labels assigned by the classifier or only selected instances are queried from the user, if the certainty of the assigned label is low. The user feedback is employed to correct classifier decisions and for the creation of new training data. The last autonomous processing step extracts markers for the stem position out of weed contours and maps them from the tile image space back to the original images.

We evaluated the overall performance of our system based on qualitative and quantitative measures. At first, we examined the implementation of the tile approach. We could show, that the mapping of stem markers from the tile space to the full image space introduces only small uncertainties, if a position tolerance of 25 px can be accepted. Additionally, the tile approach reduces the processing time, the clickload on the user and the transmission data between processing pipeline and user interface significantly.

Furthermore, we examined five different shared autonomy scenarios with various levels of autonomy in an user study. We showed that it is difficult for high level approaches where the user interacts with the classification step of the weed detection pipeline to compete against approaches with a low level of autonomy. In our low level approach, the user places stem markers directly per click and is only supported by the autonomous part of the system which creates the tile image stream. The click interface had the second shortest processing time but also produced the highest stem detection performance. Only in one case, the total image processing time was quicker. In that scenario, the user was only requested to check and correct labels of instances which were determined by an uncertainty sampling strategy. In this way, the load on the user was decreased. However, this scenario yielded in the worst overall detection performance. We observed that the user is influenced when presented with the results of the autonomous classifier compared to a scenario where the user is not presented these results. The human user tends to accept incorrect classifications from the autonomous part of the system and be more certain in his actions.

The user experience was examined qualitatively with a questionnaire. In general, the users did not have any problems to understand the interfaces and considered the additional data displayed in the form of plant contours and preassigned classifier labels to be helpful. The human users preferred the interfaces where they had to do less clickwork and could concentrate on one task. On the other hand, they recognized that a more complex interface offers a higher flexibility in terms of providing feedback to the weed detection system.

In a final evaluation, we determined the impact of the shared autonomy approach on the overall system. It turned out, that an inexperienced user is able to improve the detection rate of stem markers by a factor of 2 only by checking and correcting the labels of the autonomous classifier. An expert user could increase the precision even more, but had the

same recall. The downside of the user integration is a processing time that is fifteen times higher than the one of an autonomous system.

8.2. Conclusions

The autonomous image based detection of the stems of weed plants is an extremely complex process with success rates depending highly on external influences such as the plant density and the growth stage of plants. Due to the high occlusion and the restricted image quality, we can consider the preliminary dataset which served as base for our thesis as difficult. We could show that an autonomous system created with out-of-the-box solutions has difficulties detecting the positions of weed stems. Deeper investigation into custom-made autonomous techniques is required, however this is beyond the scope of this thesis. The introduction of a shared autonomy approach is necessary to overcome some of these difficulties and increased the detection rate by the factor of 2 for an user interaction which is restricted to the correction of autonomous classification results. However, to achieve better detection rates it was necessary to overcome restrictions which arose from the design of the autonomous framework and offer the user the possibility to interact with the system on a lower level of control by placing stem markers directly. This additional information is important immediate feedback, but hard to incorporate in learning processes.

Our user study revealed that human users are influenced by the decisions of an autonomous classifier and become more certain in their choices. There were significantly higher variations in the detection rates between the single users when they placed the markers directly (see Figure 7.8). This indicates that low level control depends highly on the experience and skills of an user whereas our abstraction layer could equalize the user performance to some extent. In general, all of our interfaces required some user introduction so that the task of the human user cannot be considered for crowdsourcing to inexperienced users.

The human accepts to work on a more abstract level of control and corporate with some machine intelligence, especially when he notices that this abstraction layer results in a reduced amount of necessary user input. However, the more we increase the level of autonomy, the higher are the demands regarding the performance of the autonomous processing system in order to maintain a stable overall system performance. Additionally, a high level of autonomy increases the risk of an user relying too much on the system's decisions and being not challenged enough to contribute his skills.

There are potential sources for errors in all portions of the autonomous framework, one of the main challenges is the segmentation of plants. Occluded areas yield undersegmentation, fine plant structures lead to oversegmentation. The cotyledons of carrot crops can be easily confused with leaves of weed plants, this seems to be the main source of classification mistakes.

8.3. Design Recommendations and Future Work

Our weed detection framework is built with the purpose to evaluate the interaction between user and autonomous system. Therefore, we employed standard solutions for autonomous processing tasks and did not focus on the tuning of this part. However, we hope that our framework will be valuable as a base system for further development and we are able to derive recommendations for future implementations based on our experiences and evaluation results.

Segmentation and stem localization

Even an expert user is not able to detect more than 40 % of the weed stem positions (see Figure 7.13) given he can only influence the classification results for detected contours.

If we assume the expert user is capable of classifying all contours correctly, this is the maximal performance an autonomous system is currently able to achieve.

This motivates the need to improve the contour segmentation step perhaps by increasing the camera resolution or employing depth information. Further improvements are possible by clustering oversegmented plant parts and allowing holes within contours. However, this would affect the choice of feature values. Another option is the incorporation of user input for the segmentation step. Stem markers set by the human user can be employed for the correction of segmentation results based on interactive segmentation algorithms such as *grabcut* [47].

Furthermore, it must be ensured that the stem position within a correctly segmented weed contour can be determined with a higher certainty. Especially here the application of depth data seems to be very helpful, as the intersection point between plant stem and soil plane can easily be determined. Another approach is the use of a camera with the optical axis parallel to the soil level that captures additional images from the side.

Classification

Based on our experiences, the final choice of the classifier does not affect the detection accuracy significantly (see Subsection 5.4.3) and rather the classifier requirements and its availability have to be considered. One important step will be the introduction of a classifier which is able to learn incrementally or can be retrained quickly. This will allow the system to maintain a dynamic classifier model during runtime which adjusts according to user input. Therefore, the system will be more robust to environmental changes during runtime. Furthermore, our experiences with the probability model for the class labels assigned by the SVM classifier is that these class probability values are not reliable. If the employment of a certainty based query strategy is continued, probabilistic classification methods such as a Bayesian Network should be considered. In general, it is a challenge finding a classifier that performs well, fulfills our system requirements and is able to estimate its own performance reliably.

The choice of feature values depends highly on the image acquisition method and the data structure used for the representation of plant objects. Additional interesting contour features for plant classification are skeleton-based features [6], curvature scale space representations [41] and fourier descriptors [80].

Tile approach

The creation of tiles is not a critical step in the image processing pipeline. However, the feature point based image fitting is time consuming and odometry data of the robot could be employed to obtain an initial estimation for the feature point matching process. Currently the tile matching is restricted to a camera movement in one dimension, more flexibility can be obtained by extending this matching algorithms to a second dimension. In general, the concept of tile images improved the situation awareness and performance of the user significantly and decreased transmission data.

Shared autonomy

The shared autonomy approach seems indispensable for such a complex system which operates in highly unstructured environments. Future challenges will be to improve the user performance by better training, maybe to develop even a game-like training application for Remote Farmers. Furthermore, the online weed detection requires a decreased user processing time, which could for example be introduced by the incorporation of several users assigned to one robot. This results in new challenges regarding synchronization and task

sharing. Another strategy to improve the user certainty is to make several users process redundant data and than incorporate their results, maybe even including certainty values assigned to the single users.

System integration

One of the most important tasks is the integration of the weed detection pipeline in the robot. In general, the ROS architecture allows an easy exchange of nodes communicating with each other, therefore only the corresponding camera image publisher and the receiver for the visual servoing images have to be created. The more challenging system integration part will be to establish the communication with the Remote Farmer, handle failure cases, make sure that the raw images are overlapping with a given ratio and synchronize image creation, robot movement, weed detection process including user input and manipulator control.

Appendix

A. Feature Values

In this Section, all 22 employed feature values are listed. The explanation for symbols and operators can be found in the Nomenclature.

F0–6 Hu-moments [40]: Position, rotation and scale-invariant features[1] derived from shape moments up to the 3rd order.

F7 Contour area A_{C}

F8

$$\frac{p}{\sqrt{A_{\mathrm{C}}}}$$

p is the contour's perimeter.

F9

$$\frac{\text{Area of the convex hull [81]}}{A_{\mathrm{C}}}$$

F10

$$\frac{\text{Area of the minimum circle enclosing the contour}}{A_{\mathrm{C}}}$$

F11

$$\frac{A_{\mathrm{bRect}}}{A_{\mathrm{C}}}$$

A_{bRect} is the area of the contour's minimum-area bounding rectangle.

F12

$$\frac{l_{\mathrm{bRect}}}{h_{\mathrm{bRect}}} \quad \text{with } l_{\mathrm{bRect}} \geq h_{\mathrm{bRect}}$$

F13 Number of convexity defects (see Figure A.1).

F14

$$\frac{\mathrm{median}_{d_i \in \mathcal{D}_{\mathrm{depth}}}(d_i)}{\sqrt{A_{\mathrm{C}}}}$$

$\mathcal{D}_{\mathrm{depth}}$ is the set with all depths of convexity defects.

F15

$$\frac{\mathrm{mean}_{d_i \in \mathcal{D}_{\mathrm{depth}}}(d_i)}{\sqrt{A_{\mathrm{C}}}}$$

F16

$$\frac{\mathrm{std}_{d_i \in \mathcal{D}_{\mathrm{depth}}}(d_i)}{\sqrt{A_{\mathrm{C}}}}$$

[1] Not considering a restricted image resolution.

F17

$$\frac{\text{median}_{l_i \in \mathcal{D}_{\text{length}}}(l_i)}{p}$$

$\mathcal{D}_{\text{length}}$ is the set with all lengths of convexity defects.

F18

$$\frac{\text{mean}_{l_i \in \mathcal{D}_{\text{length}}}(l_i)}{p}$$

F19

$$\frac{\text{std}_{l_i \in \mathcal{D}_{\text{length}}}(l_i)}{p}$$

F20 Normalized mean of intensity values in the differential image:

$$\frac{\text{mean}_{\mathbf{u}_i \in \mathcal{C}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right) - \min_{\mathbf{u}_i \in \mathcal{I}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right)}{\max_{\mathbf{u}_i \in \mathcal{I}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right) - \min_{\mathbf{u}_i \in \mathcal{I}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right)}$$

F21 Normalized standard deviation of intensity values in the differential image:

$$\frac{\text{std}_{\mathbf{u}_i \in \mathcal{C}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right)}{\max_{\mathbf{u}_i \in \mathcal{I}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right) - \min_{\mathbf{u}_i \in \mathcal{I}} \left(\text{src}_{\text{Diff}}(\mathbf{u}_i) \right)}$$

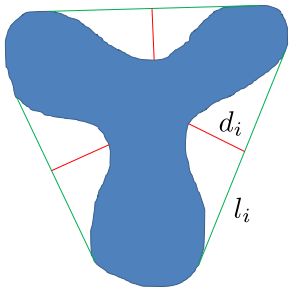


Figure A.1.: Convexity defects of a shape, d_i is the defect depth and l_i the defect length

B. Questionnaire for the User Study

Evaluation questionnaire – Interface 1

Question:	strongly disagree	disagree	neutral	agree	strongly agree
This interface was easy to understand.					
This interface was fun to use.					
It was stressful to fulfill the given task with this interface.					
I had the feeling that I could help the robot to do its task.					
The interface restricted me in the means of providing feedback to the robot.					
The plant contours provided in the interface were useful for me.					

General comments:

Evaluation questionnaire – Interface 2

Question:	strongly disagree	disagree	neutral	agree	strongly agree
This interface was easy to understand.					
This interface was fun to use.					
It was stressful to fulfill the given task with this interface.					
I had the feeling that I could help the robot to do its task.					
The interface restricted me in the means of providing feedback to the robot.					
The labeled contours provided in the interface were useful for me.					
The labels previously assigned by the classifier could not help me in fulfilling my task.					

General comments:

Evaluation questionnaire – Interface 3

Question:	strongly disagree	disagree	neutral	agree	strongly agree
This interface was easy to understand.					
This interface was fun to use.					
It was stressful to fulfill the given task with this interface.					
I had the feeling that I could help the robot to do its task.					
The interface restricted me in the means of providing feedback to the robot.					

General comments:

Evaluation questionnaire – Interface 4

Question:	strongly disagree	disagree	neutral	agree	strongly agree
This interface was easy to understand.					
This interface was fun to use.					
It was stressful to fulfill the given task with this interface.					
I had the feeling that I could help the robot to do its task.					
The interface restricted me in the means of providing feedback to the robot.					

General comments:

Evaluation questionnaire – Interface 5

Question:	strongly disagree	disagree	neutral	agree	strongly agree
This interface was easy to understand.					
This interface was fun to use.					
It was stressful to fulfill the given task with this interface.					
I had the feeling that I could help the robot to do its task.					
The interface restricted me in the means of providing feedback to the robot.					

General comments:

Final Questions

My favorite interface was:

Interface 1 Interface 2 Interface 3 Interface 4 Interface 5

Question:	strongly disagree	disagree	neutral	agree	strongly agree
The additional data provided in some of the interfaces helped me in fulfilling my task.					
It was in general hard for me to fulfill the task.					

With which gave you the feeling of having the most control?

Interface 1 Interface 2 Interface 3 Interface 4 Interface 5

Which interface did support you the best?

Interface 1 Interface 2 Interface 3 Interface 4 Interface 5

My sex:

Female Male

My age:

15-19 20-24 25-29 30-34 35-39 40-44 45-49 50-54 55-59 60-64 65-60

Bibliography

[1] Bund "Okologischer Lebensmittelwirtschaft, "Zahlen, Daten, Fakten: Die Bio-Branche 2013," tech. rep., 2013.

[2] F. Rahe, K. Heitmeyer, P. Biber, U. Weiss, A. Ruckelshausen, H. Gremmes, R. Klose, M. Thiel, and D. Trautz, "First field experiments with the autonomous field scout BoniRob," in *Proceedings 68th International Conference Agricultural Engineering 2010*, pp. 419–424, 2010.

[3] A. Michaels, A. Albert, M. Baumann, U. Weiss, P. Biber, A. Kielhorn, and D. Trautz, "Approach towards robotic mechanical weed regulation in organic farming," in *Autonomous Mobile Systems 2012*, Informatik aktuell, pp. 173–181, Springer Berlin Heidelberg, 2012.

[4] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, October 1996.

[5] B. Åstrand and A.-J. Baerveldt, "Plant recognition and localization using context information," in *Proceedings of the IEEE Conference Mechatronics and Robotics 2004–special session Autonomous Machines in Agriculture*, pp. 1191–1196, 2004.

[6] M. Weis and R. Gerhards, "Feature extraction for the identification of weed species in digital images for the purpose of site-specific weed control," *Precision Agriculture '07*, pp. 537–543, 2007.

[7] D. Slaughter, D. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63–78, 2008.

[8] A. Paap, S. Askraba, K. Alameh, and J. Rowe, "Photonic-based spectral reflectance sensor for ground-based plant detection and weed discrimination," *Opt. Express*, vol. 16, pp. 1051–1055, Jan 2008.

[9] S. Hiremath, V. A. Tolpekin, G. van der Heijden, and A. Stein, "Segmentation of Rumex obtusifolius using Gaussian Markov random fields," *Machine Vision and Applications*, vol. 24, pp. 845–854, May 2013.

[10] J. Bohren, R. Rusu, E. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5568–5575, 2011.

[11] T. Fong, C. Thorpe, and C. Baur, "Robot, asker of questions," *Robotics and Autonomous systems*, vol. 42, no. 3, pp. 235–243, 2003.

[12] B. Pitzer, M. Styer, C. Bersch, C. DuHadway, and J. Becker, "Towards perceptual shared autonomy for robotic mobile manipulation," in *ICRA'11*, pp. 6245–6251, 2011.

[13] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[14] J. W. Crandall, M. A. Goodrich, D. R. Olsen Jr, and C. W. Nielsen, "Validating human-robot interaction schemes in multitasking environments," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 4, pp. 438–449, 2005.

[15] A. E. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, "Strategies for human-in-the-loop robotic grasping," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 1–8, ACM, 2012.

[16] M. Marge, A. Powers, J. Brookshire, T. Jay, O. Jenkins, and C. Geyer, "Comparing heads-up, hands-free operation of ground robots to teleoperation," in *Proceedings of Robotics: Science and Systems*, (Los Angeles, CA, USA), June 2011.

[17] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," *Proc. AAAI*, 2011.

[18] S. R. Dixon, C. D. Wickens, and D. Chang, "Unmanned aerial vehicle flight control: False alarms versus misses," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 48, pp. 152–156, SAGE Publications, 2004.

[19] N. K. Pholchai Chotiprayanakul, Dalong Wang and D. Liu, "A haptic base human robot interaction approach for robotic grit blasting," in *The 25th International Symposium on Automation and Robotics in Construction. ISARC-2008*, pp. 148–154, Vilnius Gediminas Technical University Publishing House "Technika", 2008.

[20] A. Ubeda, E. Ianez, J. Azorin, J. Sabater, N. Garcia, and C. Perez, "Improving human-robot interaction by a multimodal interface," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 3580–3585, 2010.

[21] I. Goodfellow, N. Koenig, M. Muja, C. Pantofaru, A. Sorokin, and L. Takayama, "Help me help you: Interfaces for personal robots," in *Proc. of Human Robot Interaction (HRI)*, (Osaka, Japan), ACM Press, ACM Press, 2010.

[22] R. Parasuraman, T. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 30, no. 3, pp. 286–297, 2000.

[23] D. Damos, "Aviation automation: The search for a human-centered approach," *ERGONOMICS*, vol. 41, p. 560, APR 1998.

[24] T. Kaupp and A. Makarenko, "Measuring human-robot team effectiveness to determine an appropriate autonomy level," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 2146–2151, IEEE, 2008.

[25] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: Captchas–understanding captcha-solving services in an economic context," in *USENIX Security Symposium*, vol. 10, 2010.

[26] B. Sankaran, B. Pitzer, and S. Osentoski, "Failure recovery with shared autonomy," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 349–355, Oct. 2012.

[27] J. Chen, E. Haas, and M. Barnes, "Human performance issues and user interface design for teleoperated robots," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 6, pp. 1231–1245, 2007.

[28] H. Keskinpala, J. Adams, and K. Kawamura, "PDA-based human-robotic interface," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 4, pp. 3931–3936, 2003.

[29] M. Baker, R. Casey, B. Keyes, and H. Yanco, "Improved interfaces for human-robot interaction in urban search and rescue," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 3, pp. 2960–2965, 2004.

[30] T. Witzig, J. M. Zöllner, D. Pangercic, S. Osentoski, R. Jäkel, and R. Dillmann, "Context aware shared autonomy for robotic manipulation tasks.".

[31] T. Fong, C. Thorpe, and C. Baur, "Multi-robot remote driving with collaborative control," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 4, pp. 699–704, 2003.

[32] R. Murphy, "Human-robot interaction in rescue robotics," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 2, pp. 138–153, 2004.

[33] F. J. Pierce and P. Nowak, "Aspects of precision agriculture," vol. 67 of *Advances in Agronomy*, pp. 1–85, Academic Press, 1999.

[34] N. Zhang, M. Wang, and N. Wang, "Precision agriculture—a worldwide overview," *Computer*, vol. 36, pp. 1130–132, Nov. 2002. Engineering and Technological Sciences, International Conference on.

[35] B. Åstrand and A.-J. Baerveldt, "An agricultural mobile robot with vision-based perception for mechanical weed control," *Autonomous Robots*, vol. 13, pp. 21–35, 2002.

[36] T. Bak and H. Jakobsen, "Agricultural robotic platform with four wheel steering for weed detection," *Biosystems Engineering*, vol. 87, no. 2, pp. 125–136, 2004.

[37] T. Bakker, K. A. van, J. Bontsema, J. Müller, and G. S. van, "Systematic design of an autonomous platform for robotic weeding," *Journal of Terramechanics*, vol. 47, no. 2, pp. 63 – 73, 2010.

[38] H.-W. Griepentrog, M. Nørremark, and J. Nielsen, "Autonomous intra-row rotor weeding based on GPS," in *Proceedings CIGR World Congress-Agricultural Engineering for a Better World*, vol. 9, 2006.

[39] M. Yang, K. Kpalma, J. Ronsin, *et al.*, "A survey of shape feature extraction techniques," *Pattern recognition*, pp. 43–90, 2008.

[40] Hu, "Visual-pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[41] F. Mokhtarian and A. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 789 –805, aug 1992.

[42] D. Andújar, A. Escolà, J. Dorado, and C. Fernández-Quintanilla, "Weed discrimination using ultrasonic sensors," *Weed Research*, vol. 51, no. 6, pp. 543–547, 2011.

[43] U. Weiss, P. Biber, S. Laible, K. Bohlmann, and A. Zell, "Plant species classification using a 3D LIDAR sensor and machine learning," in *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pp. 339–345, Dec. 2010.

[44] T. Burks, S. Shearer, and F. Payne, "Classification of weed species using color texture features and discriminant analysis," *Transactions of the ASAE*, vol. 43, pp. 441–448, Mar.-Apr. 2000.

[45] Weis and Gerhards, "Qualitative und quantitative Messung der Verunkrautung in Kulturpflanzenbestaenden mittels Bildanalyse," *Bornimer Agrartechnische Berichte*, vol. Heft 60, 2007.

[46] R. Zwiggelaar, "A review of spectral properties of plants and their potential use for crop/weed discrimination in row-crops," *Crop Protection*, vol. 17, no. 3, pp. 189–206, 1998.

[47] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, pp. 309–314, Aug. 2004.

[48] S. H. Lee, H. Il Koo, and N. Ik Cho, "Image segmentation algorithms based on the machine learning of features," *Pattern Recogn. Lett.*, vol. 31, pp. 2325–2336, Oct. 2010.

[49] L. Ji and J. Piper, "Fast homotopy-preserving skeletons using mathematical morphology," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 653–664, June 1992.

[50] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie, "Visual recognition with humans in the loop," in *Proceedings of the 11th European conference on Computer Vision: Part IV*, ECCV'10, (Berlin, Heidelberg), pp. 438–451, Springer-Verlag, 2010.

[51] J. Costa, C. Silva, M. Antunes, and B. Ribeiro, "On using crowdsourcing and active learning to improve classification performance," in *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 469–474, November 2011.

[52] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," pp. 3–12, Springer-Verlag, 1994.

[53] R. Szeliski, "Image alignment and stitching: a tutorial," *Foundation and Trends in Computer Graphics and Vision*, vol. 2, pp. 1–104, Jan. 2006.

[54] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, pp. 59–73, Aug. 2007.

[55] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[56] T. T. Herbert Bay, Andreas Ess and L. V. Gool, "Surf: Speeded up robust features," in *Computer Vision and Image Understanding (CVIU)*, vol. 110, pp. 346–359, 2008.

[57] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09*, pp. 331–340, INSTICC Press, 2009.

[58] C. L. Wiegand, A. J. Richardson, D. E. Escobar, and A. H. Gerbermann, "Vegetation indices in crop assessments," *Remote*, vol. 35, pp. 105–119, 1991.

[59] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.

[60] F. Meyer, "Color image segmentation," in *Image Processing and its Applications, 1992., International Conference on*, pp. 303–306, 1992.

[61] M. Youssef, K. Asari, R. Tompkins, and J. Foytik, "Hull convexity defects features for human activity recognition," in *Applied Imagery Pattern Recognition Workshop (AIPR), 2010 IEEE 39th*, pp. 10–7, October 2010.

[62] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning*, (San Francisco, CA, USA), pp. 249–256, Morgan Kaufmann Publishers Inc., 1992.

[63] M. A. Hall, *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, Department of Computer Science, 1999.

[64] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, 2002.

[65] R. Polikar, L. Upda, S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 31, no. 4, pp. 497–508, 2001.

[66] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, pp. 10–18, Nov. 2009.

[67] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, Jan. 1991.

[68] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, pp. 161–205, May 2005.

[69] S. le Cessie and J. van Houwelingen, "Ridge estimators in logistic regression," *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.

[70] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning* (B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds.), pp. 185–208, Cambridge, MA, USA: MIT Press, 1999.

[71] G. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 338–345, Morgan Kaufmann, 1995.

[72] J. R. Quinlan, *C4.5 : programs for machine learning*. The Morgan Kaufmann series in machine learning, San Mateo, Calif.: Kaufmann, 1993.

[73] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[74] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011.

[75] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *J. Mach. Learn. Res.*, vol. 5, pp. 975–1005, Dec. 2004.

[76] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[77] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[78] B. Kempf, "The boost.threads library," *C/C++ Users Journal*, pp. 6–13, 2002.

[79] N. Chinchor, "Muc-4 evaluation metrics," in *Proceedings of the 4th conference on Message understanding*, MUC4 '92, (Stroudsburg, PA, USA), pp. 22–29, Association for Computational Linguistics, 1992.

[80] B. Jähne, *Digital image processing : 155 exercises and CD-ROM*. Berlin: Springer, 6 ed., 2005.

[81] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79–83, 1982.

List of Figures

1.1. State-of-the-art weed regulation technique for organic carrot fields in Germany 6
1.2. Bonirob2 with delta kinematic for weed manipulation 7
1.3. Evaluation of difficulties in the weed segmentation process 8
1.4. Examples for plants encountered in our dataset 8
1.5. Overview of the system design with the shared autonomy approach 10
1.6. Autonomous weed detection process 10

2.1. Levels of automation and four-stage model of human information processing 15
2.2. Redesign of an user interface for an USAR application [29] 18
2.3. Aeryon scout quadrocopter for image acquisition 20
2.4. Vehicles for autonomous weed control 21
2.5. Commercial products related to autonomous weed detection 22

3.1. Different concepts for user-classifier interaction for object classification . . . 29
3.2. Weed detection process with shared autonomy classification 31
3.3. Design mockups for user interfaces 33
3.4. Interface for user feedback in the weed detection process 34

4.1. The tile approach 38
4.2. Relationships between the different image and contour types 39
4.3. Dataflow between the full image layer, the tile image layer and the user . . 40
4.4. Filtered feature point matches for two images 43
4.5. The transformed image and its resulting tile in the tile image stream 43
4.6. Assembly of 21 tile images without tile deskewing 44
4.7. Steps of the tile deskewing process 44
4.8. Assembly of 21 tile images with tile deskewing 45
4.9. Limitations of the mosaicing algorithm for a non-planar environment 46
4.10. Illustration of the effects of a perspective transformation with two fixed points 46

5.1. Image subtraction for plant-soil discrimination 48
5.2. Segmentation with the watershed algorithm 50
5.3. Comparison of unoptimized and optimized preprocessed camera images . . 50
5.4. Optimizations in the contour matching algorithm 51
5.5. The contour classification process 55
5.6. Second iteration of a grid search for SVM parameter selection 56
5.7. Weed plants, their contours and user-assigned stem locations 57

6.1. Network topology of the weed detection pipeline 60
6.2. Associations between full- and tile images 61
6.3. Overview of the threads and their tasks 62

7.1. Example for a distance threshold of $r = 25\,\mathrm{px}$ 64
7.2. $F_{0.5}$-score, precision and recall of the stem marker mapping 64

7.3. Two overlapping images 65
7.4. Evaluation results of the tile image approach 66
7.5. The first four different types of interfaces evaluated in the user study 68
7.5. The fifth interface evaluated in the user study 69
7.6. Average processing times and standard deviation of the different interfaces 70
7.7. Average user input events and standard deviation 71
7.8. $F_{0.5}$-score, precision and recall of the stem detection with different interfaces 71
7.9. All interfaces are easy to understand 73
7.10. General questions 73
7.11. Whether user consider the interface to be stressful to use 73
7.12. Whether users felt restricted in providing feedback to the robot 74
7.13. $F_{0.5}$-score, precision and recall of the weed stem detection without any users 75
7.14. Processing times for 15 images 75

A.1. Convexity defects of a shape, d_i is the defect depth and l_i the defect length 84

List of Tables

5.1. Classification accuracy with standard deviation for different classifiers . . . 54

6.1. Overview of the main ROS-packages contained in the project stack 60

Nomenclature

A_C	Area of a contour
p	Contour perimeter
d_i	Convexity defect depth
l_i	Convexity defect length
y	Distance between two descriptor vectors of feature points
$\tilde{\mathbf{u}}$	$\tilde{\mathbf{u}} \in \mathbb{R}^3$ is a homogeneous image position with $\tilde{\mathbf{u}}_i = s \begin{pmatrix} \mathbf{u}_i \\ 1 \end{pmatrix}$.
$\mathbf{H}$	Homography matrix.
mean	$\mathrm{mean}_{x_i \in \mathcal{X}}(x_i) = \frac{1}{n} \sum_{x_i \in \mathcal{X}} x_i$ with n being the number of elements in the finite set $\mathcal{X} \subset \mathbb{R}$
median	$\mathrm{median}_{x_i \in \mathcal{X}}(x_i) = \begin{cases} \frac{1}{2}(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}) & \text{for } n \text{ even} \\ x_{(\frac{n+1}{2})} & \text{for } n \text{ odd} \end{cases}$ for a sorted sequence $x_{(1)}, x_{(2)}, \ldots x_{(n)}$ containing all elements of the finite set $\mathcal{X} \subset \mathbb{R}$
$\mathbf{u}$	$\mathbf{u} \in \mathbb{N}_0^2$ is an image coordinate.
$\mathcal{B}$	$\mathcal{B} \subset \mathbb{N}_0^2$ is a finite set containing all background coordinates of an image
$\mathcal{C}$	$\mathcal{C} \subset \mathbb{N}_0^2$ is a finite set containing all pixel coordinates within a contour
$\mathcal{D}_\mathrm{depth}$	$\mathcal{D}_\mathrm{depth} \subset \mathbb{R}$ is a finite set containing all convexity depth values of a contour
$\mathcal{D}_\mathrm{length}$	$\mathcal{D}_\mathrm{length} \subset \mathbb{R}$ is a finite set containing all convexity lenght values of a contour
$\mathcal{M}_\mathrm{dist}$	$\mathcal{M}_\mathrm{dist} \subset \mathbb{R}$ is a finite set containing the distance between the descriptors of feature points.
$\mathcal{I}$	$\mathcal{I} \subset \mathbb{N}_0^2$ is a finite set containing all pixel coordinates of an image
src	$\mathrm{src}(\mathbf{u}_i)$ is the intensity value of an graylevel image at the coordinate $\mathbf{u}_i$.
std	$\mathrm{std}_{x \in \mathcal{X}}(x) = \sqrt{\frac{1}{n-1} \sum_{x_i \in \mathcal{X}} \left(x_i - \mathrm{mean}_{x_i \in \mathcal{X}}(x_i)\right)^2}$ with n being the number of elements in $\mathcal{X} \subset \mathbb{R}$
GPS	Global Positioning System
GUI	Graphical User Interface

HRI Human Robot Interaction

NIR Near Infrared, infrared light close to the range of human-visible wavelengths

PA Precision Agriculture

Qt A cross-plattform framework mainly used for the development of graphical user interfaces.

Remote Farmer The user interacting with the weed detection framework

ROS Robot Operating System

SVM Support Vector Machine. A classification algorithm.

UAV Unmanned Aerial Vehicle

USAR Urban Search and Rescue

VS Visual Servoing