SAPIENZA
UNIVERSITÀ DI ROMA

# Perception and Environment Modeling for Robotics in Agricultural Contexts

**Ciro Potena**

ID number 1689956

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Engineering in Computer Science*

*XXXI Cycle*

Department of Computer, Control,
and Management Engineering
Sapienza University of Rome
Rome, Italy

January 2020

**Thesis Advisor**

Prof. Daniele Nardi
Eng. Alberto Pretto

**Co-Advisor**

Prof. Marilena Vendittelli

**Review Committee**

Prof. Cyrill Stachniss
Prof. Tom Duckett

Thesis defended on February 2020
in front of a Board of Examiners composed by:

Prof. Paolo Merialdo (chairman)
Prof. Matteo Matteucci
Prof. Maria Francesca Costabile

**Perception and Environment Modeling for Robotics in Agricultural Contexts**
Ph.D. thesis. Sapienza University of Rome

Website: https://cirpote.bitbucket.io/

Author's email: potena@diag.uniroma1.it

# Acknowledgments

Among the words that have overwhelmingly entered in my vocabulary it is important to mention: *sugar beet*, *sunflower*, *precision agriculture*, *pushcart* (in italian "*carretto*"), *crop rows*, and more recently *orchar*, *hazelnut*, and *sucker*. Until 5 years ago, I would never thought to use the above mentioned words in my daily life. This is especially thanks to my advisors: Daniele Nardi and Alberto Pretto. Their passion and dedication have been the major source of inspiration for my growth as a researcher. Specifically, I want to thank Daniele Nardi for his continuous support and encouragement, and I would also like to emphasize that my journey began thanks to him. Then, I want to thank Alberto Pretto. I really enjoyed the 5 years working together, full of discussions (till late at night) to improve our research and understanding about this challenging world that is farming robotics. In general, without your dedication, I would not have been who I am now.

I am grateful to Roland Siegwart and Juan Nieto who gave me the opportunity to enrich my journey with a wonderful experience in their lab.

Thanks to all the Lab.Ro.Co.Co. members for having been fantastic colleagues and friends. In particular I want to thank Marco: we started our journey in the Ph.D. program and in the Flourish project together. We shared blood, sweat, sunny fields, pixels, and the weirdest experiences.

Thanks to Angela, for being such an awesome partner since more than 9 years. In you, I found comprehension, inner peace, and eventually the love of my life.

Thanks to my family, my father Roberto, my mother Concetta, and my brother Nicandro. You have been supporting me since my birth, always indicating me the right direction to follow. If I reached my destination it is only thanks to you.

In a competitive universe, such as research, I cannot forget to thank also my academic rivals (that I do not cite individually). In such moments, where the passion was not enough, they silently helped me to find the proper motivation to keep working hard.

Last but not least, a special mention to the *capsella bursa pastoris*, having been the major obstacle along my research journey, and having slowly become an essential building block for the passwords of most of my academic accounts.

## Abstract

The automation of agricultural tasks is the key to sustainable and efficient food production. In this context, Precision Agriculture (PA) and robotics are set to have a great impact and to transform this sector. The former is a farming management concept based on monitoring key indicators of crop health and targeting treatment only to plants or infested areas that need it. The latter is the technological tool that will allow applying the PA concept in the fields. However, to effectively carry out almost any (farming) task in autonomy, a robot must be capable of perceiving its surroundings and, at the same time, modeling the environment in a meaningful manner.

Despite the great progress achieved during the last years in automating farming activities by using robotic platforms, most of the existing systems do not provide a sufficient autonomy level. This is due to a couple of major factors. On the one hand, the farming scenario shows peculiar characteristics and challenges, while, on the other hand, current robotic solutions mainly focus on specific farming tasks and may lack in flexibility. Making farming robots more autonomous brings the benefits of completing faster the assigned tasks and adapting to different purposes and farm fields, which make them more useful while increasing their profitability. However, making farming robots more autonomous involves increasing their perception and awareness of their surrounding environment.

This thesis focuses on perception methods that enable robots to autonomously operate in farming environments, specifically a localization and mapping method and a collaborative mapping between aerial and ground robots. They improve the robot perception capabilities by exploiting the unique context-based characteristics of farm fields and by merging sensorial data gathered from several heterogeneous sensors. Additionally, this thesis addresses the problem of crop/weed mapping by employing end-to-end visual classifiers. This thesis also presents contributions to perception-based control methods. Such approaches allow the robot to navigate the environment while taking into account the perception constraints. The following is a full list of contributions:

- A method to summarize a big dataset by information entropy maximization. The manual annotation of the summarized dataset allows the trained network to obtain a similar classification accuracy while down-scaling the manual annotation effort.

- A model-based dataset generation method for crop and weed detection. The generated data can be used to supplement a real-world training dataset, reducing the manual annotation effor. Some synthetic data are made available as open-source datasets.

- A multi-cue positioning system for ground farming robots that fuses several heterogeneous sensors and incorporates context-based characteristics.

- A novel multimodal environment representation that at the same time enhances the key characteristics of the farm field, while filtering out redundant information.

- A collaborative mapping method that registers agricultural maps acquired by both aerial and ground vehicles.
- Perception-based control methods that steer a (farming) robot to the desired location while satisfying perception constraints.

Moreover, another important outcome of this thesis is a set of open-source software modules and datasets, which I hope the community will benefit from. The work developed in this thesis has been done following the operating scenario proposed by the Flourish project[1], in which Sapienza, University of Rome, participated as a consortium partner.

[1] http://flourish-project.eu/

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

There is a growing demand for food production due to the rapid growth of the world population. Sustainable agriculture is the only viable way of meeting this demand without causing a negative environmental impact. Precision agricultural techniques seek to address this challenge by observing, measuring and responding to inter and intra-field variability in crops [92]. Intervariability may result from a significant number of factors such as weather variables (temperature, precipitations, relative humidity, etc.), soil characteristics (texture, depth, nitrogen levels), cropping practices (till/no-till farming), weeds and diseases, etc.

These practices, however, cannot meet alone themselves the growing demand of food since they are extremely labour intensive and time consuming. An effective solution to this problem can be found in the robotic sector. Indeed, as pointed out in [39, 17], the availability of autonomous system architectures gives the opportunity to develop a new range of flexible agricultural equipment based on small, smart machines that reduce waste, improve economic viability, reduce environmental impact, and increase food sustainability. The exploitation of this technology will enable to automatize almost any agricultural task. An Unmanned Aerial Vehicles (UAVs) will be able to survey a field from the air, to build an aerial map and to detect infested areas. On the other hand, a multi-purpose agricultural Unmanned Ground Vehicles (UGVs) can refine the aerial map, perform targeted intervention on the ground, and provide detailed information for decision support, all with minimal user intervention.

However, state-of-the-art farming robots show automation capabilities that are not mature enough to allow them to safely operate in different real-world scenarios, or work reliably during long periods of time. One of the main reasons derives from the farming context itself. Autonomous operations of robotic systems in an agricultural environment, indeed, is rather a challenging task due to the peculiar characteristics and features of common farming scenarios. The environment is dynamic, non-deterministic and semi-structured with many sources of noises. In addition, a strong perceptual aliasing arises in either visual or geometrical sensorial data.

For instance, a robot autonomously navigating the rows in a farm field should be able to accurately estimate its pose along with other essential navigation parameters such as the distance between the rows while they are constantly changing in their size and appearance. It should also be capable of reliably detect the end of the row when it is in sight and plan a safe trajectory to not damage the surrounding plants. Likewise, the weed control unit, with which a farming robot is generally equipped, should be able to identify, in real-time, weeds from the plants. Differentiating crops from weeds is a challenging problem since they share similar textures, sizes, and shapes. The robot should also be able to estimate the position of the weed to a high degree of precision to carry out an effective focused treatment without damaging the neighboring plants. Finally, the robots, if more than one, should use the newly gathered data to update an environmental map from where obtain useful high level information to help the high level decision making. These problems are further accentuated by the strong perceptual aliasing and by the fact that the sensors have some inherent limitations in their range and resolution, and sensor measurements can be noisy.

This thesis tackles some specific perception problems that arise when designing an autonomous farming robot. It describes scene segmentation methods and algorithms to reduce the manual data annotation effort. These algorithms consist of data summarization approach that automatically selects from a large dataset the most informative subsets that better describe the original one, and a model-based dataset generation approach that exploits a modern graphic engine to build a highly realistic synthetic dataset. Additionally, this thesis describes localization and mapping methods, either individual or collaborative, to enable a farming robot to represent a farm field and to localize within it. This thesis also presents perception-based control methods to enable the robot to autonomously move in the farm field, while explicitly taking into account perception constraints.

This introduction is structured as follows. Sec. 1.1 provides a brief introduction to Precision Agriculture, and a general overview about the advantages and challenges in using robots in agricultural contexts. Sec.1.2 summarizes the state-of-the-art of the main research topics covered by this dissertation. Sec. 1.3 depicts the operating scenario adopted in this thesis, helping in contextualizing the contributions and motivating the research objectives of this dissertation. Sec. 1.4 provides a full list of the contributions presented in this thesis.

## 1.1. Precision Agriculture

Precision Agriculture (PA), also known as 'smart farming', is now a term used throughout agricultural systems worldwide. However, the concept of fully autonomous agricultural vehicles goes back in the past: the first development of robotics in agriculture can be dated as early as the 1920s [165], with research to incorporate automatic vehicle guidance into agriculture beginning to take shape, while examples of early driverless tractor prototypes using leader cable guidance systems date back to the 1950s and 1960s. However, the first time the PA concept has made its appearance was in the late 1980s with the matching of grid-based sampling of soil chemical properties for grain industries. Since then, with the advancement and availability on the market of new technologies (e.g. Global Navigation Satellite System (GNSS) in the 1990s, and so on ...), PA concept spread across different farming industries and crop types. For a more comprehensive review of the PA history, I refer the reader to [150].

Many definitions of PA exist, but the most appropriate comes from the US House of Representatives (US House of Representatives, 1997) which defines PA as an approach to manage the farming production process that allows to "an integrated information and production-based farming system that is designed to increase long term, site-specific and farm production efficiency, productivity and profitability while minimizing unintended impacts on wildlife and the environment".
This definition properly resumes the main PA principles: taking into account the time and space variability of factors that influence the farming production process to improve the input efficiency in the dynamic management of the process. Improving efficiency means to use fewer resources while achieving better results, or, achieving the same results while using the same amount of inputs. At first glance, therefore, it would seem that the objective of PA is a revisitation of good agricultural management, for example as regards crop production. However, the commonly adopted good agricultural practices do not take adequately into account the dynamics of agricultural systems, which lead to large temporal variability in the responses to the production factors. In this regard, PA makes use of the best solutions technology has to offer in terms of monitoring and actuation capabilities, and it is usually applied in fields by smart systems that can automatize the work in the field and effectively target the treatments. For example, intelligent irrigation systems can measure the crop growth status and the water stress with a local granularity and respond by irrigation strategy that aims in minimizing the water waste.
However, in the last years, the development of new sensing technologies and smart machines gave the opportunity to robots to emerge as enabling technology for the future robotic PA systems [39].

### 1.1.1. Robots in Agriculture

Ever since quadrotor, or robots in general, became affordable and reliable about a decade ago, the number of their applications has increased very fast and by now it covers essentially every task that involves surveying, e.g., gathering data in a fast, reliable, and autonomous manner. Among the others, a part of these applications falls in the PA context.
Emerging applications of robots in agriculture include weed control ([32], [5], [4]), cloud seeding [33], planting seeds, harvesting, environmental monitoring, and soil analysis ([10], [101]), while flying drones are successfully deployed for crop monitoring and spraying [34]. Other use-cases concern sheep shearing and autonomous horticultural tasks such as pruning and spraying.
In summary, most of the applications of agricultural robots are for active tasks, *i.e.* moving and interacting with the environment autonomously. Such tasks are hard, repetitive, and expensive to accomplish without robots, but they are also what robots are particularly well suited for. Companies and Start-ups ([8], [2]), all together, form a billion-dollar market, which is expected to grow much more in the coming years. Therefore there is a huge need to further exploit the potential of autonomous robots. To make such robots really useful and more profitable, it is crucial to increase their autonomy, their capability to properly model the surrounding environment, and the speed at which they can operate. The current bottleneck in the implementation of these functionalities is largely in the perception capabilities of agricultural robots

that are the main focus of this dissertation.

### 1.1.2. Advantages

The potential benefits brought by robots, either UAVs or UGVs, in PA applications are several.

From an aerial perspective, flying drones can perform fast aerial inspections of the target field. Their high thrust-to-weight-ratio enables UAVs to carry different kinds of sensors, enabling the user to customize the data acquisition according to the specific needs of the user. The gathered data are well suited to compute RGB and multispectral maps of the farmland, helping the farmer to better monitor diseases, pest, irrigation, fertilizer condition, and the crop growth information as well. Moreover, thanks to the recent developments in embedded computing platforms, they can also carry-out data processing on-line and fly autonomously, allowing the operator to work on other tasks.

On the other hand, ground farming robots are designed to replace human labor. They do not have stringent payload constraints such as aerial vehicles, therefore they can carry more sensors and end effectors as well. In particular, end effectors allows the platform to actively interact with the environment and to perform different tasks such as harvesting, berry-thinning, spraying, and bagging. Actuators can be designed in a modular manner by choosing different sensor settings, status indicators, and ground treatment packages, thus leading the platform to be adaptable to a wide range of farm management activities and different crop types. Moreover, ground vehicles can be also deployed to perform detailed inspections of specific areas of the field. Finally, the gathered information can be used alongside existing precision agriculture machinery, for example, by providing position maps for fertilizer application.

In addition, besides the platform typologies, an essential advantage is the possibility to bring several types of sensors, in particular, visual sensors. They represent one of the most informative sensors on the market, compared to their cost, and offer several benefits for farming tasks. Vision-based is mandatory to perform tasks such as detection, segmentation, and classification of crop versus weeds ([96, 36, 127]). It is also crucial to track people or obstacles to enable safe navigation within the fields, and it also allows for an accurate actuation through visual-servoing techniques.

In summary, robots are well suited for agricultural tasks since they can autonomously perform several farming activities in an autonomous manner.

### 1.1.3. Challenges

Despite there has been great progress on automating agriculture activities using robots, there are still some open challenges that prevent farming robot to safely and effectively operate in all types of farming scenarios.

A common challenge is represented by weather and environmental conditions. Robots should work in real-world conditions such as humidity, fog, and varying temperatures. For example, UAVs should be robust to rain and snow and should guarantee long operational time, while UGVs should be capable of traveling on uneven and noisy terrains and working at night. In addition, sensors and batteries should guarantee a working range with both low and high temperatures.

Another important challenge to address is the flexibility of farming robots. Indeed, most existing systems have been developed to solve only specialized tasks. The specificity of each farming task, either from the perception or from the actuation perspectives, does not allow for an easy application of the same software and hardware solution in different contexts. Different harvesting tasks, for instance, require ad-hoc end effectors to pick the fruits, and computer vision methods can not easily adapt the robot semantic scene understanding according to the crop-type features. Moreover, the robot may need an adaptable shape to deal with different interspaces between crop rows.

The agricultural scenario itself also involves, in general, unique challenges. The scene presents strong perceptual aliasing and it is also dynamic. As a direct consequence, standard localization and mapping algorithms are likely to yield poor results, and loop closures methods, either visual and geometrical, are not reliable enough. On the other hand, relying on a RTK-GPS as the main localization source leads the robot to be vulnerable to outages and usually involves an initial effort for setting up the base stations. This also leads the collaboration in a multi-platform robotic solution to be extremely challenging. The data sharing among robots without a reliable data association also prevents the solution to be effective and usable.

The unique agricultural field characteristics also make field navigation quite challenging. For example, the ground vehicle has to move with sufficient accuracy to avoid damaging the crop, while the flying drone, in case of a per-plant inspection, should constantly frame the target plant with the on-board camera due to the strong visual aliasing.

Agricultural scenarios also offer a big challenge for computer vision and machine learning. The former requires to be robust to illumination changes, weather conditions, and perceptual aliasing while ensuring enough accuracy to effectively carry out weed control and to estimate crucial vision-based parameters for safe navigation. Computer vision applied to PA often depends on machine learning, particularly for those classification and segmentation tasks such as crop and weeds identification. Most of the employed machine learning techniques fall in the Deep Learning area, such as Convolutional Neural Networks (CNNs). While these methods can solve individual classification problems with impressive accuracy, the visual appearance of agricultural scenarios dramatically varies field by field. Also training a classifier for a specific crop presents overwhelming challenges: (i) the many differences between the crop types, (ii) the crop growth under different conditions, and (iii) even the different ground conditions make this task extremely challenging for machine learning. A statement that properly summarizes these concepts may be: "*Deep learning is great at interpolating conditions between what it knows; it is not good at extrapolating to situations it hasn't seen. And in agriculture, you always feel that there is a set of conditions that you haven't yet classified*" [35]. Moreover, the problem of optimally train machine learning-based classifiers is still open. Dealing with the above-introduced variability involves the acquisition and pixel-wise annotation of large datasets through a large human effort. Finally, a working machine learning architecture should also adapt the learner models during long-term operations to deal with the strong varying conditions.

Summarizing, the above-mentioned problems and the lack of flexibility pose a high risk of no return on investment for farmers. This thesis is tackling the following challenges around farming robots:

- Development of methods to minimize the manual annotation effort for training effective crop-weeds visual classifiers;
- Achieving a reliable localization and mapping capability for farming robots by exploiting the characteristics of the agricultural environment;
- Enabling a ground-aerial farming robot collaboration by providing a shared environment representation;
- Allowing robots to explicitly take into account perception targets in the vehicle control pipeline.

## 1.2. State-of-the-art

This section summarizes the state-of-the-art in the key research problems addressed in this thesis: crop/weeds classification, localization and mapping, heterogeneous robot collaboration, and perception-based control. Our aim is to show how the work presented in this thesis relates to and extends the state-of-the-art methods.

### 1.2.1. Crop/Weeds Classification

The problem of plant classification can be considered an instance of the so called *fine-grained visual classification* (FGVC) problem, where the aim is to distinguish between objects belonging to the same macro-class, such as species of animals, or models of cars. FGVC problems are intrinsically difficult, since the visual differences between these similar categories (in our case, plant species) are often minimal, and only in recent works have been obtained noteworthy results ( [27], [66], [96], [97]).
Despite the literature on this topic is extremely wide, it can be roughly split into two major paradigms: hand-crafted features ([146], [23], [67], [27],[86]) and end-to-end learning methods ([66], [96], [97], [127], [138], [95]).
The former methods are based on common, or hand-crafted features, such as Color Co-occurrence Method (CCM) from hue, saturation and intensity color space, Scale Invariant Feature Transform (SIFT), Grid-Based Color Moment (GBCM), and many others. Those methods report high classification performances in the order of 80-90% in terms of accuracy and strongly depend on the type of features chosen.
Conversely, end-to-end learning methods address the problem in a different manner: they train neural networks to allow them to directly learn the proper features for the classification/segmentation task. These approaches achieve classification accuracy around 94%, due to their ability to discriminate also crops that are very similar to weeds but suffer from inflexibility and limitations in their representative power. Indeed, all those data-driven systems share the need for a tedious labeling effort. In the context of precision agriculture, the requirement for large datasets leads to a significant effort: datasets should be acquired across different plant growth stages and weather conditions.

### 1.2.2. Localization and Mapping

An accurate global pose estimation system is an essential component for an effective farming robot. However, self-localizing within farmlands poses several challenges, in particular for the strong aliasing, either visual or geometrical, and for the lack of distinguishable landmarks. In addition, crops grow over time constantly changing their appearance. One common way to overcome these limitations is by using Global Navigation Satellite System (GNSS) to geo-localize the robot in the farm field ([113], [151], [155]). However, the vulnerability to outages makes the above solutions not fully reliable in the farming robotics context.

On the other hand, the benefits in using local sensory information ([163], [29], [76], [105], [65], [84]), and the incorporation of a robust detection of crop rows ([41], [83]) in the navigation pipeline offer the possibility to overcome the RTK-GPS related issues. Although those methods proved sufficient reliability, they might suffer from the challenging crop row extraction in the post-emergency growth stage of the crops or, in some cases, they are tailored for specific use-cases and might not generalize over different crop types.

### 1.2.3. Aerial-Ground Robots Cooperation

Thanks to the complementarity of their characteristics, a combination of aerial and ground robots is commonly deployed in different contexts. PA particularly benefits from this multi-platform solution since UAVs allow for rapid inspections of the farm field, while UGVs can perform targeted interventions.

The multi-robot cooperation is a recurrent problem and many solutions have been proposed by either using multi-robot SLAM algorithms or map merging/map registration strategies in both 2D ([15, 19, 139]) and 3D ([21, 51, 75]) settings. However, the heterogeneity of the involved robots and the lack of distinctive visual and geometrical features in an agricultural environment prevent the employment of standard multi-robot SLAM methods. On the other hand, map registration is a challenging problem, especially when dealing with heterogeneous robots, where data is gathered from different points-of-view and with different noise characteristics. It has been intensively investigated, especially in the context of urban reconstruction with aerial and ground data ([25], [152], [49]).

All the above-mentioned solutions make use of strong context-based assumptions that might not fit the agricultural context. There are a few works that deal with map registration in an agricultural context by using homogeneous robots ([28], [37]). However, the further challenges that appear when registering UAV on UGV lead the former methods to not be enough reliable.

### 1.2.4. Peception-Based Control

Controlling a robot in a PA context means to steer the vehicle from a specific pose configuration to another one to perform certain agronomic interventions and/or collect data. In this regard, the goal pose may depend on the agronomic target location in the field. However, a farming field often shows a strong repetitiveness in its patterns, either geometric or visual, possibly misleading the on-board navigation system.

A possible solution to overcome this challenge is to constantly keep the agronomic target in the center of the field-of-view of a sensor mounted on-board, such as a camera. This problem is commonly known as *visual servoing*, namely controlling a robot through a direct visual feedback, and has been widely investigated in the last decades ( [124], [63], [61], [9], [104]).

The above-cited methods can be split into two parallel branches: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). In PBVS, the 3D goal pose is directly obtained from a 3D view of the target, while IBVS formulates the problem in terms of image feature locations. Both suffer their own weaknesses: the former is very sensitive to initial conditions, camera calibration parameters and image noise corruption. In the latter, it is particularly challenging to model the relationship between vehicle dynamics and the feature projection error, especially for under-actuated systems. The above-introduced issues are even more pronounced in a farming scenario since retrieving a 3D prior knowledge from an agronomic target is extremely challenging.

Recent approaches are based on hybrid techniques, where image features and 3D data are fused together to develop a more stable controller than IBVS or PBVS alone ([62], [147], [43], [42]).

Despite the promising results, the formerly introduced approaches might suffer a high computational cost and, in general, they assume to fly in an obstacle-free environment.

## 1.3. The Flourish project

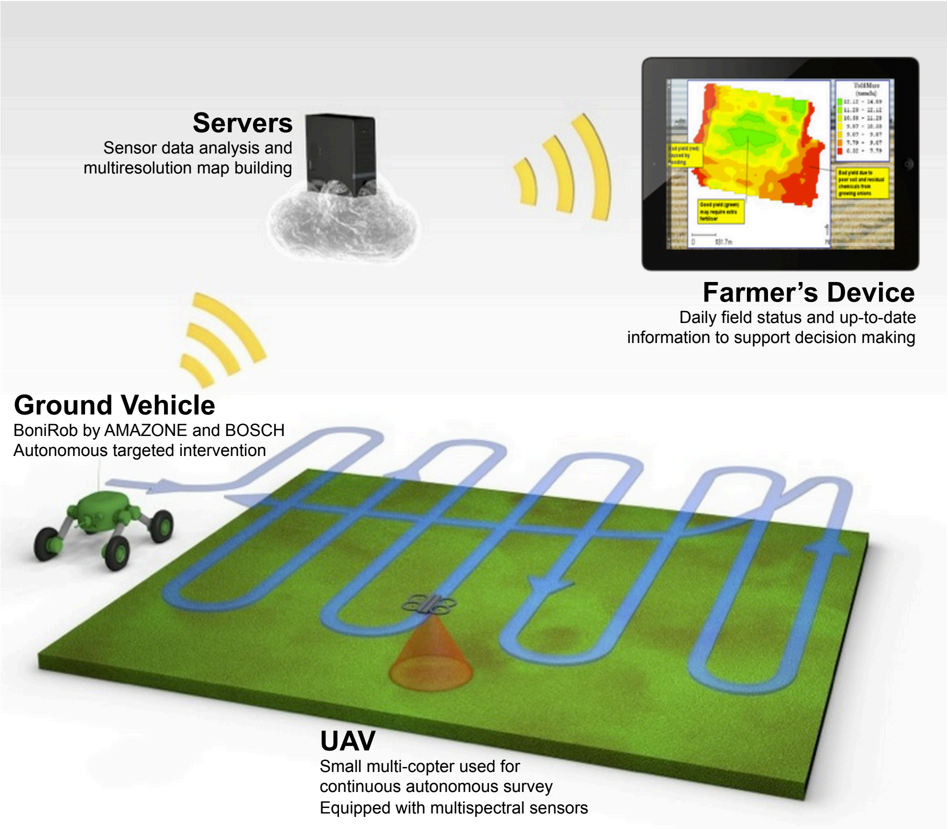

Servers
Sensor data analysis and
multiresolution map building
Farmer's Device
Daily field status and up-to-date
information to support decision making
Ground Vehicle
BoniRob by AMAZONE and BOSCH
Autonomous targeted intervention
UAV
Small multi-copter used for
continuous autonomous survey
Equipped with multispectral sensors

**Figure 1.1.** The working scenario: an aerial robot and a ground robot collaborate to monitor a farm field. The picture has been taken from the EU founded Project "Flourish".

To set the application that motivates the work done in this thesis, this section presents an operating scenario: the Flourish Project, an EU funded project, in which Sapienza University of Rome participated as partner of the consortium, working on the mapping and on the robot collaboration part. The Flourish project proposes to develop an adaptable robotic solution for precision farming by combining the aerial survey capabilities of a small autonomous multi-copter Unmanned Aerial Vehicle (UAV) with a multi-purpose agricultural Unmanned Ground Vehicle (UGV). With such a multi-platform setup, the whole system is able to survey a field from the air, perform a targeted intervention on the ground, and provide detailed information for decision support, all with minimal user intervention. An example of a working scenario is depicted in Fig. 1.1: an aerial robot and a ground robot collaborate to monitor a farm field. The UAV is equipped with multispectral sensors that are used to extract vegetation health indexes and to spot weed patches within the field. Thus, through continuous monitoring of the field, the UAV builds and keeps up to date a multi-spectral map of the field, communicating to the UGV the location of the weeds in the field. Once the weed data are delivered, the UGV performs targeted intervention, where the nature of the treatment may also be decided by the user, and updates the field map. The whole data are then sent to the farmer to support high-level decision making.

## 1.4. Contributions

This section summarizes the key contributions of this thesis. It further highlights the connections between the individual results and points to related video and open-source code contributions. In total, the research presented in this thesis has been published in five peer-reviewed conference publications and two journal publications. These works received two nominations for the **Best Student Paper Award Finalist**. The relevant algorithms and datasets developed within this work are available as open-source[1].

### 1.4.1. Crop/Weeds Classification

In this section, I present algorithms that reduce the human effort required to train an effective and robust classification system for crop/weed segmentation.

[1] www.dis.uniroma1.it/~labrococo/fsd

### Fast and accurate crop and weed identification with summarized train sets for precision agriculture

1. **C. Potena**, D. Nardi, and A. Pretto. "Fast and Accurate Crop and Weed Detection with Summarized Train Sets for Precision Agriculture". In Intelligent Autonomous Systems 14 (IAS), 2016, pp. 105-121, **Best Student Paper Award Finalist**, doi: 10.1007/978-3-319-48036-7_9

Deep neural networks have been setting new records in almost all of the computer vision challenges and continue to gain interest in the field of artificial intelligence. Such networks have a big potential in Precision Agriculture since they allow to train highly discriminative visual models capable to distinguish among different plant species with great accuracy. On the other hand, the level of expressivity of such networks is limited by the size and the annotation quality of the training dataset. To deal with these issues, I propose an unsupervised dataset summarization algorithm that automatically summarizes a large, non-annotated, dataset by taking the most informative part of the initial dataset. This enables to streamline and speed-up the manual dataset labeling process, otherwise extremely time consuming, while preserving good classification accuracy.

### Automatic Model Based Dataset Generation for Fast and Accurate Crop and Weeds Detection

1. M. Di Cicco, **C. Potena**, G. Grisetti, and A. Pretto. "Automatic model based dataset generation for fast and accurate crop and weeds detection". In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 5188-5195, doi: 10.1109/IROS.2017.8206408

To cope with limited available data, there are several approaches to aid the training of deep networks. A novel approach is to synthetize training samples through computer-generated imagery. One of the main concerns in the use of synthetic data for the training is whether the source domain is close enough to the target domain (i.e., the domain of real-world images) to make it possible an effective application. Modern graphic engines meet these constraints, by offering high realism and high variability in the generation procedure by changing viewpoint, illumination, weather conditions, and the level of detail which provides a valuable augmentation of the data. I present a model-based dataset generation for training crop/weed classifiers. By parameterizing the generation procedure with the key aspects of the target environment (i.e., number of leaves, plants growth stage, and so on), the proposed approach is able to generate a potentially infinite number of annotated realistic scenes with a few starting real-world textures. The reported experiments show how to use these data for augmenting a small real-world dataset to sensibly improve the classification accuracy. Some synthetic images generated by the proposed approach are available as open-source. This work results from a fruitful collaboration with Ph.D. Maurilio Di Cicco under the joint supervision of my Prof. Alberto Pretto and Prof. Giorgio Grisetti. My contribution mainly lies in the idea on which the work is built and on the deep neural network architecture setting up and testing.

**Related Datasets**
http://www.dis.uniroma1.it/ labrococo/fsd/syntheticdatasets.html

**Related Video**
https://www.youtube.com/watch?v=jMcxquK6G8k

## 1.4.2. Localization and Mapping

In this section, I present our localization and mapping system, and its application on the Bosch Bonirob. With this work I achieved a few centimeter localization errors, thus enabling a farming robot to safely navigate through the field.

### An Effective Multi-Cue Positioning System for Agricultural Robotics

1. M. Imperoli*, **C. Potena***, D. Nardi, G.Grisetti, and A. Pretto. "An Effective Multi-Cue Positioning System for Agricultural Robotics". In IEEE Robotics and Automation Letters (RAL), 2018, vol. 3, num. 4, pp. 3685-3692, doi: 10.1109/LRA.2018.2855052

One of the main capabilities that autonomous ground farming robots must have is to navigate the environment without harming the valuable crop. Modern mapping and localization frameworks have proven to be accurate and reliable in several robotics scenarios. However, localizing a robot within an agricultural scenario is a complex task, and conventional approaches might easily fail. To deal with the above-mentioned issues, I present a multi-cue positioning system specifically tailored for farming robots. To achieve the required localization accuracy, the proposed approach fuses several heterogeneous sensor data with a few context-based constraints. The entire optimization is cast as a pose-graph optimization and runs in real-time. The results show how this approach allows achieving a small centimetric error, even in case of a GPS failure. The proposed algorithm and together with two datasets annotated with ground truth are available as open-source. This work results from a fruitful collaboration with Ph.D. Marco Imperoli under the joint supervision of my Prof. Alberto Pretto, Prof. Giorgio Grisetti, and Prof. Daniele Nardi. My contribution is equally split with Ph.D. Marco Imperoli and, in particular, lies in the whole system development, testing, and data acquisition on the field.

**Related Software**
https://bitbucket.org/cirpote/pose_graph
https://bitbucket.org/Imperoli/mcaps

**Related Datasets**
http://www.dis.uniroma1.it/ labrococo/fsd/mappingdatasets.html

**Related Video**
https://www.youtube.com/watch?v=iumZcOM49wo

### 1.4.3. Aerial-Ground Robots Cooperation

In this section, I present our collaborative mapping framework for heterogeneous robots. With this work I achieved cutting edge registration performance between UAV and UGV maps, also when the maps to register have a significant scale difference.

#### AgriColMap: Aerial-Ground Collaborative 3D Mapping for Precision Farming

1. **C. Potena**, R. Khanna, J. Nieto, R. Siegwart, D. Nardi, and A. Pretto. "AgriColMap: Aerial-Ground Collaborative 3D Mapping for Precision Farming". In IEEE Robotics and Automation Letters (RAL), 2019, vol. 4, num. 2, pp. 1085-1092, doi: 10.1109/LRA.2019.2894468

In a robotic multi-platform application like the Flourish project, both aerial and ground robots must share a common environment representation. This property is essential to let the robots collaborate toward the common goal (e.g., updating the field map with newly acquired information). On the other hand, merging exteroceptive information from heterogeneous robots presents a few challenges: the drastic viewpoint change, data scale discrepancies, and strong visual aliasing. To deal with the above-introduced issues, I present a method for an effective and robust map merging. The proposed method firstly transforms the 3D colored maps into a multimodal environment representation, and then it casts the data association problem as a Large Displacement Optical Flow (LDOF). The results show that the resulting non-rigid transformation allows to effectively merge maps with large scale discrepancies and initial guess errors.

**Related Software**
https://github.com/cirpote/agricolmap

**Related Datasets**
http://www.dis.uniroma1.it/ labrococo/fsd/collaborativemapping.html

**Related Video**
https://www.youtube.com/watch?v=F3FtxcB1kOM

### 1.4.4. Perception-Based Control

In this section, I present control approaches that allow incorporating visual and avoidance constraints, allowing for safe and effective navigation through the environment. Despite their generality, the presented method provides several benefits also in farming activities, particularly when performing monitoring tasks at the resolution of the single plant.

#### Effective target aware visual navigation for UAVs

1. **C. Potena**, D. Nardi, and A. Pretto. "Effective target aware visual navigation for UAVs". In European Conference on Mobile Robots (ECMR), 2017, pp. 1-7.

doi: 10.1109/ECMR.2017.8098714

In this paper, I propose an effective vision-based navigation method that allows a multirotor vehicle to simultaneously reach a desired goal pose in the environment while constantly facing a target object or landmark. Standard techniques such as Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) in some cases (e.g., while the multirotor is performing fast maneuvers) do not allow to constantly maintain the line of sight with a target of interest. Instead, the proposed approach computes the optimal trajectory by solving a non-linear optimization problem that minimizes the target reprojection error, while meeting the UAV's dynamic constraints. The desired trajectory is then tracked by means of a real-time Non-linear Model Predictive Controller (NMPC): this implicitly allows the multirotor to satisfy both the required constraints. The method has been successfully evaluated in many real and simulated experiments, making an exhaustive comparison with a standard approach.

### Non-Linear Model Predictive Control with Adaptive Time-Mesh Refinement

1. **C. Potena**, B. Della Corte, D. Nardi, G. Grisetti, and A. Pretto. "Non-linear model predictive control with adaptive time-mesh refinement". In IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2018, pp. 74-80, doi: 10.1109/SIMPAR.2018.8376274

In this paper, I present a novel solution for real-time, Non-Linear Model Predictive Control (NMPC) exploiting a time-mesh refinement strategy. The proposed controller formulates the Optimal Control Problem (OCP) in terms of *flat* outputs over an adaptive lattice. In common approximated OCP solutions, the number of discretization points composing the lattice represents a critical upper bound for real-time applications. The proposed NMPC-based technique refines the initially uniform time horizon by adding time steps with a sampling criterion that aims to reduce the discretization error. This enables a higher accuracy in the initial part of the receding horizon, which is more relevant to NMPC while keeping bounded the number of discretization points. By combining this feature with an efficient Least Square formulation, our solver is also extremely time-efficient, generating trajectories of multiple seconds within only a few milliseconds. The performance of the proposed approach has been validated in a high fidelity simulation environment, by using a UAV platform. An open-source C++ implementation of the proposed method has been also released.

### Related Software

https://bitbucket.org/gnomicSolver/gnomic

### Joint Vision-Based Navigation, Control and Obstacle Avoidance for UAVs in Dynamic Environments

1. **C. Potena**, D. Nardi, and A. Pretto. "Joint Vision-Based Navigation, Control and Obstacle Avoidance for UAVs in Dynamic Environments". In European

Conference on Mobile Robots (ECMR), 2019, **remaining details to appear soon**

This work addresses the problem of coupling vision-based navigation systems for Unmanned Aerial Vehicles (UAVs) with robust obstacle avoidance capabilities. The former problem is solved by maximizing the visibility of the points of interest, while the latter is modeled through ellipsoidal repulsive areas. The whole problem is transcribed into an Optimal Control Problem (OCP) and solved in a few milliseconds by leveraging state-of-the-art numerical optimization. The resulting trajectories are well suited for reaching the specified goal location while avoiding obstacles with a safety margin and minimizing the probability of losing the route with the target of interest. Combining this technique with a proper ellipsoid shaping (i.e., by augmenting the shape proportionally with the obstacle velocity or with the obstacle detection uncertainties) results in a robust obstacle avoidance behavior. The reported experiments validate the approach through extensive simulated experiments that show effective capabilities to satisfy all the constraints even in challenging conditions. An open-source C++ implementation of the proposed method has been also released.

**Related Software**
https://github.com/cirpote/rvb_mpc

# Chapter 2

# Crop/Weed Segmentation

In this chapter, I focus on the crop/weed detection module. Such a module is an essential component for farming robots since it allows the platform to properly perceive the environment and to carry out weed control by either autonomous removal or spraying. For example, in the Flourish project context, an effective plant segmentation module would enable the robot to properly distinguish between sugarbeets and weeds, allowing the sprayer to autonomously apply treatments to the target weeds by using visual feedback. However, detecting crop and weeds represents one of the most challenging problems: the differences between plant categories are minimal, and plants are often overlapping. An illustration of an image-based classification input and output is reported in Fig. 2.5 (more specifically, second column).

The most promising state-of-the-art approaches in this area usually make use of machine learning techniques, such as Convolutional Neural Networks (CNNs) ([127][132][85]) or random forest classifiers [96]. The usage of such techniques, especially CNNs, allows training highly discriminative visual models capable to distinguish among different plant species with great accuracy. The major drawback of these data-driven approaches is that the level of expressivity is limited by the size of the training dataset. The bigger the training dataset is the bigger the generalization capabilities of the model are. However, in the context of precision agriculture, the requirement for large datasets usually leads to a significant effort. More specifically, datasets should be acquired across different plant growth stages and weather conditions. In addition, the training images must be provided with an accurate semantic pixel-wise annotation. Despite the images can be pre-processed and segmented, thus reducing the annotation effort at the segment level [89], the manual pixel-wise annotation is still a challenging and extremely time-consuming task. Actually, due to this problem, the size of the semantic datasets is usually relatively small [164].

In this chapter, I report the two major contributions of this thesis which address the crop/weed segmentation and the minimization of the labeling effort problems under two different perspectives:

1. A fast and accurate crop and weed identification with summarized train sets for precision agriculture;
2. An automatic model-based dataset generation for fast and accurate crop and

weed detection.

## 2.1. Related Work

### 2.1.1. Plants Classification/Segmentation

The problem of plant classification can be considered an instance of the so-called *fine-grained visual classification* (FGVC) problem, where the purpose is to recognize subordinate categories such as species of animals, models of cars, etc. FGVC problems are intrinsically difficult since the differences between similar categories (in the considered case, plant species) are often minimal, and only in recent works, the researchers obtained noteworthy results (e.g., [119, 167]).
Burks *et al.* [23] proposed to use Color Co-occurrence methods (CCM) as texture statistics as input variables for a backpropagation (BP) neural network for weed classification. Feyaerts and van Gool [46] presented a performance of a classifier based on multispectral reflectance in order to distinguish the crop from weeds. The best classifier, based on neural networks (NN), reached a classification rate of 80% for sugar beet plants and 91% for weeds. Also Aitkenhead *et al.* [7] proposed a NN based approach: the captured images were first segmented into cells that are successively classified, achieving a final accuracy up to 75%. In Haug *et al.* [65] a Random Forest (RF) classifier was proposed. It uses a large number of simple features extracted from a large overlapping neighborhood around sparse pixel positions. This approach achieves strong classification accuracies, due to its ability to discriminate also crops that are very similar to weeds. This approach has been improved in Lottes *et al.* [96] by extending the features set and including a relative plant arrangement prior that helps to obtain better classification results.
Recently Han Lee *et al.* [91] presented a leaf-based plant classification system that uses convolutional neural networks to automatically learn suitable visual features. Also Reyes *et al.* [132] used CNN for fine-grained plant classification: they used a deep CNN with the architecture proposed by Krishevsky *et al.* [85], first initialized to recognize 1000 categories of generic objects, then *fine-tuned* (i.e., specialized) for the specific task to recognize 1000 possible plant species.

### 2.1.2. Synthetic Dataset Generation

Modern data-driven classification approaches like CNN architectures require a large amount of data to obtain the best performance. One recently proposed solution to address this issue is to generate synthetically the training datasets. Many approaches addressed this issue by exploiting modern graphic engines. [134, 144] present approaches based on synthetic data extracted from computer video games, showing how a merged training dataset composed of synthetic and real-world images performs better than a real one. Modern video games are a compelling source of high quality annotated data but they also have an important drawback. They are usually closed-source, so it is not possible to customize and modify the output data stream. In other approaches, the artificial world is totally handcrafted using modern graphic tools. Mancini *et al.* [100] presented an urban scenario developed by means of Unreal Engine 4 for the monocular depth estimation. Experiments using the synthetic data

for the training phase, without any fine-tuning, show good generalization properties on real data. A similar technique has been used in [64] to build a city environment for pedestrian detection, showing how using purely synthetic data is able to outperform models trained on a limited amount of real-world scenes. In that kind of solution, the great human effort required in the dataset acquisition and labeling is transferred into the synthetic world crafting. Even if the latter involves a minor effort, it is still a time-consuming activity and the amount of generated data depends on the size of the generated artificial world.
Another approach used to mitigate the labeling effort is based on transfer learning [117]. For instance Yosinski et al. [168] show how better is the transferability of features depending on the distance between the base and the target task.

## 2.2. Fast and Accurate Crop and Weed Identification with Summarized Train Sets for Precision Agriculture

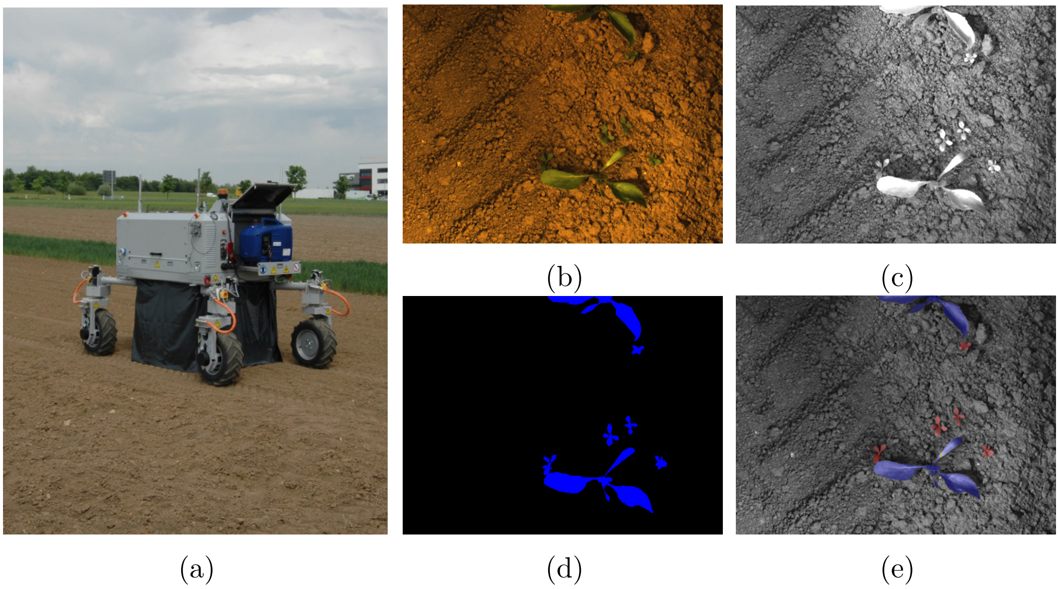


**Figure 2.1.** (a) One of the BOSCH Bonirob employed to acquire the datasets used in the experiments; (b),(c) An example of an RGB+NIR image pair acquired by the multispectral camera mounted on the robot; (d) The segmented image obtained using the proposed vegetation detection algorithm: blue pixels represent projections of 3D points that belong to green vegetation; (e) The results of the proposed pixel-wise classification algorithm: pixels that belong to crop are highlighted in violet, pixels that belong to weeds are highlighted in red.

In this section of the thesis, I present a robust and efficient weed identification system that leverages the effectiveness of convolutional neural networks (CNNs) in both the detection and classification steps. The proposed system takes as input 4-channels RGB+NIR images (e.g., Fig. 2.1(b),(c)), provided by a multispectral camera mounted on a farm robot (e.g., Fig. 2.1(a)) that autonomously monitors the crop and can apply selective weed treatments. The weed identification task includes, before plant classification, a plant detection step. Detection is generally a more challenging and time-consuming task compared with classification, since it may require an exhaustive search in the whole image, with variable bounding box sizes. In the context of green plants, the detection task can be simplified by exploiting the Normalized Difference Vegetation Index (NDVI) [135], extracted from the RGB+NIR images: NDVI enables to obtain a simple, fast and pixel-wise segmentation between green vegetation and soil (e.g., [65], [96]). Unfortunately, being threshold-based, this technique is not robust against illumination changes and different soil conditions: a careful tuning of the threshold and an outlier removal process are necessary to get a good segmentation [96]. To overcome these limitations, the proposed method combines the NDVI based segmentation with a trained lightweight CNN (henceforth referred to as *sNet*) that takes as input small patches of the RGB+NIR images. The idea is to use a very conservative threshold in order to select through the NDVI most of the true positive pixels (i.e., pixels that represent vegetation). The CNN is then used to validate each selected pixel, pruning most of the false positives (e.g., Fig. 2.1(d)). The reported experiments show that this hybrid technique outperforms the NDVI based segmentation while preserving a good computational speed.

Pixels marked as vegetation in the segmentation step are then processed with a deeper 3-classes CNN (henceforth referred to as *cNet*) in order to recognize the category (crop, weeds or soil). Despite *cNet* processes only pixels classified as vegetation in the previous step, including also the class '*soil*' in the *cNet* CNN helps to prune at no cost the remaining false positives not detected by the *sNet* CNN. In order to meet the real-time constraints required by the proposed system, i.e. the classification process should terminate within one second from the image acquisition time[1], the proposed method also employs a *blob-wise* voting scheme, where blobs are connected regions extracted from the segmentation mask. The reported experiments will show that: (a) the classification stage achieves state-of-the-art results; (b) the pipeline composed by the two sequential CNNs (*sNet* + *cNet*) obtains similar results if compared with a single *cNet*, but with a considerable gain in speed.

In the last part of this section, I also address a relatively new problem that I call *unsupervised dataset summarization*. It is well known that CNNs to be effective require large manual labeled training datasets [143]. Unfortunately, plant identification requires a challenging and extremely time consuming per-pixel labeling process. The proposed idea is to reduce the size of the dataset *before* the manual labeling stage, in order to streamline and speed-up the manual dataset labeling process while preserving good classification performances. The proposed algorithm automatically selects a subset of $K$ images that contain the most informative features over the $N$ images of the whole dataset, $K \ll N$, in order to summarize in the best possible way the original dataset. The labeling process will then involve only these $K$ images. The features based subsets selection method is different from other max-relevance and min-redundancy feature selection methods (among the others, [121, 158]) since it is *unsupervised*, i.e. it does not require the labels as input. The unsupervised dataset

[1] In the considered setup, one second represents a reasonable time constraint in order to enable the robot to actively remove the weeds as soon as they are detected.

summarization problem is formulated as a combinatorial optimization problem, using as a reward a submodular set function inspired by the coverage set functions used in text document summarization problems [93]. The reported experiments will show that the proposed dataset selection algorithm outperforms in all the experiments both the random dataset selection and the supervised manual selection strategies.

## 2.2.1. Vision-Based Plant Classification

### Vegetation Detection

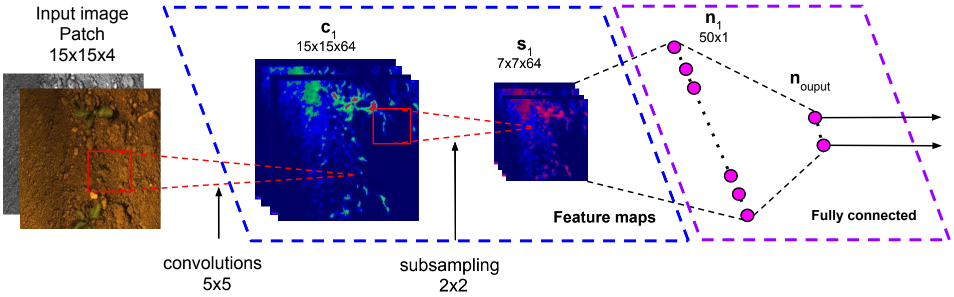


**Figure 2.2.** Architecture of the *sNet* CNN.

The goal of the vegetation detection task is to discriminate in the RGB+NIR images between pixels that represent projections of 3D points that belong to green vegetation and the other pixels. This process enables to simplify and speed up the subsequent plant detection and classification tasks.

Due to the photosynthesis, healthy green plants absorb more solar energy in the visible spectrum, causing a low reflectance level in the RGB channels. Similarly, the reflectance of the near-infra-red spectrum is affected by the same phenomena with opposite results and, as a direct consequence, with a low reflectance level in the NIR channel.

A well-known indicator that is used to measure the reflectance properties of the plants is the Normalized Difference Vegetation Index (NDVI) [135], which is calculated as follows for each pixel $(u, v)$:

$$\mathcal{I}_{NDVI}(u,v) = \frac{\mathcal{I}_{NIR}(u,v) - \mathcal{I}_{R}(u,v)}{\mathcal{I}_{NIR}(u,v) + \mathcal{I}_{R}(u,v)} \tag{2.1}$$

where $\mathcal{I}_R(u,v)$ and $\mathcal{I}_{NIR}(u,v)$ stand for the spectral reflectance measurements taken from the R channel (visible red) and from the near-infrared channel, respectively. The vegetation detection task is typically solved by means of a thresholding operation on the NDVI image: a pixel $(u, v)$ is classified as vegetation if $\mathcal{I}_{NDVI}(u,v) > th_V$, with $th_V$ a fixed threshold. Unfortunately, a single threshold usually is not robust against illumination changes and different soil conditions, even inside a single image. To address this problem, the idea is to combine the NDVI with a lightweight convolutional neural network. The proposed approach firstly performs a thresholding operation on the NDVI using a conservative threshold, that allows preserving most of the pixels that belong to vegetation. For each pixel classified as vegetation, it exploits a trained CNN applied to a 15×15 pixels 4 channels patch around the pixel. This network (*sNet*, Fig. 2.2) includes a single convolutional layer with rectified linear unit (ReLU) activation function, followed by a max-pooling layer and a local response normalization step. The strides in the convolutional layer are set to 1 and the strides in the pooling layer are set to 2, where a max pool operator is applied to 2×2 patches. The normalized neurons provided as output from the convolutional and pooling layers are used as inputs for a fully connected layer. The final neurons are then fully connected to the output labels '*plant*' (i.e., vegetation) and '*soil*' (i.e., not vegetation), that are normalized through a softmax layer. The architectural choices made for this CNN represent a good experimental trade-off between the sake of efficiency and the segmentation performances (see Sec. 2.2.3).

### Pixel-Wise Crop/Weed Classification

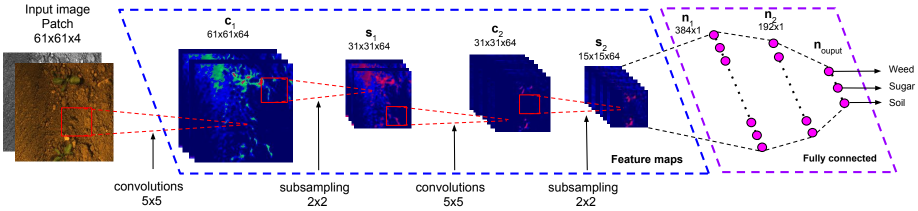


**Figure 2.3.** Architecture of the *cNet* CNN.

The detection system described so far provides an accurate vegetation mask of the input image. Pixels that belong to vegetation need now to be classified between crops and weeds. In this plant classification task, there are a lot of possible error sources, among others the similarity between plant species and the partial overlapping between different plants. To learn more specific features that help to disambiguate in these challenging conditions, the second step of the proposed approach exploits a slightly deeper network with input patches of 61×61 pixels over the 4 RGB+NIR channels and, accordingly, a higher number of output neurons for every layer. The final network *cNet* (Fig. 2.3) includes two convolutional layers with ReLU activation function, each followed by a max pooling layer and a local normalization layer. As in the *sNet*, both the max pooling layers of *cNet* operate on 2×2 patches with strides of 2 pixels. The normalized feature maps are then used as inputs for two fully connected layers before passing through a softmax activation function.

Despite the *cNet* processes only pixels classified as vegetation by *sNet* (i.e, pixel classified as "plant"), *cNet* also keeps the class '*soil*' as a possible output of *cNet*. Through experimental evidence, I found that this helps pruning at no cost the remaining false positives not detected by the *sNet* CNN.

### Blob-Wise Crop/Weed Classification

The plant detection and classification pipeline previously presented provides state-of-the-art results, but it still suffers from some limitations: (a) A pure pixel-wise

```
Data: The input image $\mathcal{I}$ and a conservative threshold $th_{NDVI}$ for the
      NDVI.
Result: A set of classified blobs $\mathbf{B}_c$
/* Compute the vegetation mask $\mathcal{I}_v$                                */
foreach $(u, v) \in \mathcal{I}$ do
    $\mathcal{I}_v \leftarrow$ 'soil';
    if $\mathcal{I}_{NDVI}(u, v) \geq th_{NDVI}$ then
        if sNet$(u, v)$ = 'plant' then
            $\mathcal{I}_v \leftarrow$ 'plant';
        end
    end
end
Extract from $\mathcal{I}_v$ a set of blobs (i.e., connected regions) $\mathbf{B} = \{b_i, \ldots, b_n\}$ of
  pixel classified as 'plant';
$\mathbf{B}_c \leftarrow \{\}$;
foreach $b_i \in \mathbf{B}$ do
    /* A number of pixel is randomly sampled from the blob,
       where $s$ depends on the blob size $|b_i|$                            */
    Sample $s$ pixel $(u, v)_j$ from the blob $b_i$, $s < |b_i|$;
    Classify each pixel $(u, v)_j$ using cNet;
    if The majority of the s pixels have been classified as 'sugar' then
        $\mathbf{B}_c \leftarrow \mathbf{B}_c \cup \{b_i, sugar\}$;
    else if The majority of the s pixels have been classified as 'weed' then
        $\mathbf{B}_c \leftarrow \mathbf{B}_c \cup \{b_i, weed\}$;
    end
end
```

**Algorithm 1:** Blob-Wise Crop/Weed Detection and Classification

approach can lead to the detection of false-positive plants composed by very few misclassified pixels; (b) Differently from *sNet*, *cNet* does not meet the real-time constraints required by the proposed system.

In order to address these problems, the proposed method employs a blob-wise based voting scheme that speeds-up the processes while removing most of the small false positives plants. The pseudo-code of the proposed method is reported in Algorithm 1: it first computes the vegetation mask $\mathcal{I}_v$ as described in Sec. 2.2.1 (lines 1-8), it extracts all the connected regions whose pixels are classified as '*plant*' (line 9) and, finally, it classifies the blobs by applying *cNet* on a subset of pixels (lines 10-19): each pixel "votes" for a class, the majority decides the class of the whole blob, blobs classified as '*soil*' are discarded.

### 2.2.2. Unsupervised Dataset Summarization

The CNNs described above should be trained using pixel-wise labeled datasets: unfortunately, pixel-wise data annotation is an extremely time-consuming process, even if the user can exploit specific labeling tools that allow to quickly detect pixels belonging to vegetation by means of local thresholding operations based on

NDVI. The first solution to this problem would be to extract and label only a subset of $K$ images, randomly selected between the $N$ images of the original dataset, $K \ll N$. Experimental evidence (Sec. 2.2.3) indicates that a randomly selected subset often does not well describe the original dataset, i.e. the subset provides a poor information "coverage" of the original dataset. Alternatively, the subset selection process could be done manually, by looking for a "good" subset of the sample images that well represent the original dataset: this strategy usually enables to obtain better classification results compared with randomly select subsets. I introduce here a simple but effective algorithm that enables to automatically select a subset of the training set that shows very good coverage properties over the original dataset. I call this problem *unsupervised dataset summarization*, where unsupervised means that the subset is extracted before the labeling process and summarization means that the subset must be very informative about the original dataset. This problem can be formulated as a special case of the *Knapsack Problem*, that given a set $\mathbf{V}$ of $N$ elements, each one with a given weight $c_i$, asks for the subset $\mathbf{S}^*$ that maximize a *set* function $\mathcal{F} : 2^{\mathbf{V}} \rightarrow \mathbb{R}$ subject to a constraint that requires the total weight of the subset to be less or equal than a given threshold $K$:

$$\mathbf{S}^* = \operatorname*{argmax}_{\mathbf{S} \subseteq \mathbf{V}} \mathcal{F}(\mathbf{S}) \ \textit{subject to} \ \sum_{i \in \mathbf{S}} c_i \leq K \tag{2.2}$$

The set function $\mathcal{F}$, also called *objective function*, measures the "quality" of a given subset. In this case, the set $\mathbf{V}$ is the original dataset that contains $N$ images, the constraint is represented by an equality constraint where for each $i$ you have $c_i = 1$, while the set function $\mathcal{F}$ should tell how well the subset $\mathbf{S}$ summarizes the original dataset $\mathbf{V}$. It is well known that this class of problems is NP-hard, so the computation of the optimal solution $\mathbf{S}^*$ is often not feasible. Despite that, a good approximated solution can be obtained if the objective function $\mathcal{F}$ is *monotone submodular*. A set function $\mathcal{F}$ is submodular if for each $\mathbf{A} \subseteq \mathbf{B} \subseteq \mathbf{V}$ and for some element $x \notin \mathbf{B}$, you have that:

$$\mathcal{F}(\mathbf{A} \cup x) - \mathcal{F}(\mathbf{A}) \geq \mathcal{F}(\mathbf{B} \cup x) - \mathcal{F}(\mathbf{B}) \tag{2.3}$$

A submodular set function is monotone if for each $\mathbf{A} \subseteq \mathbf{B}$ you have $\mathcal{F}(\mathbf{A}) \leq \mathcal{F}(\mathbf{B})$. Submodular functions have a very attractive property [109]: it can be proven that if $\mathcal{F}$ is monotone submodular, then $\mathcal{F}(\hat{\mathbf{S}}) \geq \left(1 - \frac{1}{e}\right) \mathcal{F}(\mathbf{S}^*) \approx 0.632\ \mathcal{F}(\mathbf{S}^*)$[2], with $\hat{\mathbf{S}}$ an approximated solution computed using a greedy algorithm.

### Subset Selection as a Document Summarization

The proposed method is inspired by the document summarization task that, given a set, $\mathbf{V}$ that contains all the sentences of a text document, searches for a subset of sentences $\mathbf{S} \subseteq \mathbf{V}$ that well represents the original document. Typically this task is subject to some constraints, such as the maximum number of words or the maximum number of sentences that comprise the subset.

Let a dataset acquired by a robot moving in the field be the original "document" $\mathbf{V}$,

[2] This is a lower bound: in most of the practical cases the approximated solution ensures much better results.

possibly composed by thousands of images. Let each image be a "sentence" of $\mathbf{V}$, each one composed by a set of "visual words" [148], the problem of subset selection can be reduced as a standard document summarization problem. Lin and Bilmes [93] faced the document summarization problem by proposing a class of submodular set functions that measure both the similarity of the subset $\mathbf{S}$ to the document to be summarized (also called "coverage" of the original document) and the "diversity" of the sentences that compose the subset $\mathbf{S}$. Since the goal is to encourage subset $\mathbf{S}$ that well describe $\mathbf{V}$, the proposed approach employs as objective function a simple but effective coverage set function:

$$\mathcal{L}(\mathbf{S}) = \sum_{i \in \mathbf{V}, j \in \mathbf{S}} w_{ij} \tag{2.4}$$

where $w_{ij} \geq 0$ represents a similarity between the image (i.e., "sentence") $i$ and the image $j$. $\mathcal{L}(\mathbf{S})$ is clearly monotone submodular.

### Bag-of-Visual-Words from the CNN

In the document summarization task, sentences are usually represented using bag-of-terms vectors: in a similar way, the proposed method represents each image using bag-of-visual-words vectors [148]. Since the goal is to train a CNN (in this case, the CNN  of Fig. 2.3) using a very informative subset of the original dataset, the aim is to extract the visual words *directly* from the trained CNN. In a typical CNN architecture, the sequence of convolutional layers usually computes a $n$-dimensional vector $f$, used as input of a sequence of fully connected layers: the decision over the output classes depends only on $f$. Such a vector represents a descriptor, or *signature*, of the input image or patch. In this specific case, the proposed method applies the *cNet* of Fig. 2.3 to 61×61 possibly overlapping patches of the input image. After two convolutional + pooling layers (blue dotted box in Fig. 2.3) the patch is reduced to a 384-dimensional vector $f$. The idea is to represent an image as a collection of $m$ visual words, derived from the vectors $f_i$, $i = 1, \ldots, m$ provided by the CNN applied to $m$ patches. If the cardinality of the vocabulary trough visual words is denoted with $W$, it becomes possible to quantize the descriptors $f$ into visual words exploiting the k-means clustering algorithm [16]. The bag-of-visual-words vector for a given image is simply the $W$-dimensional histogram that reports the number of times that each visual word $\alpha$ appears in the image.
$w_{ij}$ is computed by using the following cosine similarity:

$$w_{ij} = \frac{\sum_{\alpha \in \mathbf{S}_i} (h_{\alpha,i} \cdot h_{\alpha,j} \cdot ih_{\alpha}^2)}{\sqrt{\sum_{\alpha \in \mathbf{S}_i} (h_{\alpha,i}^2 \cdot ih_{\alpha}^2)} \sqrt{\sum_{\alpha \in \mathbf{S}_j} (h_{\alpha,j}^2 \cdot ih_{\alpha}^2)}} \tag{2.5}$$

where $h_{\alpha,i}$ and $h_{\alpha,j}$ are the number of times that the visual word $\alpha$ appears in the image, and $ih_{\alpha}$ is the inverse document frequency, that is calculated as the logarithm of the ratio of the number of images where $\alpha$ appears, over the total number of images $N$ that compose the input dataset.

### The Proposed Algorithm

The proposed method is not directly applicable: In this work, it is tacitly assumed that the considered CNN is already able to provide valid results even if it is still

```
Data: The input dataset $\mathbf{V}$ with $N$ images $\mathcal{I}$, the size $W$ of the visual word
      vocabulary, the size $K$ of the output subset
Result: The selected subset $\mathbf{S}$
foreach $\mathcal{I} \in \mathbf{V}$ do
    Extract in a fixed grid a number of $m$ patches;
    For each patch, compute the descriptor $f$ provided as output of the
      convolutional layers of the pre-trained CNN;
end
Quantize all the descriptors into $W$ visual words using the k-means
  algorithm;
foreach $\mathcal{I} \in \mathbf{V}$ do
    Compute the $W$-dimensional histogram that reports the numbers of
      times that each visual word $\alpha$ appears in $\mathcal{I}$;
end
$\mathbf{S} \leftarrow \{\}$;
for $k \leftarrow 1$ to $K$ do
    $\mathcal{I}^* \leftarrow \operatorname*{argmax}_{\mathcal{I} \in \mathbf{V} \setminus \mathbf{S}} \mathcal{L}(\mathbf{S} \cup \{\mathcal{I}\})$;
    $\mathbf{S} \leftarrow \mathbf{S} \cup \{\mathcal{I}^*\}$;
end
```

**Algorithm 2:** Unsupervised Dataset Summarization

in the training phase (i.e., the approach is looking for a good subset of the original data set to be used for training). This issue is solved by pre-training the CNN using a general labeled auxiliary dataset or a randomly selected, manually labeled subset of the input dataset.

The pseudo-code of the Unsupervised Dataset Summarization technique is reported in Algorithm 2: firstly, the CNN descriptors from a set of patches (lines 1-4) are computed, then the bag-of-visual-words vectors (lines 5-8) are extracted and finally the subset **S** is selected by using a simple greedy algorithm that exploits the coverage set function reported in Eq. 2.4 and the similarity between images reported in Eq. 2.5 (lines 9-13).

### 2.2.3. Experimental results

The experimental results presented in the following are designed to show the accuracy of the proposed classification system. They also confirm the performances reached by a CNN trained on a small and very representative dataset, build-up by the proposed unsupervised dataset summarization approach.

#### Experimental setup

Two datasets are considered, both collected from a BOSCH Bonirob farm robot (Fig. 2.1(a)) moving on a sugar beet field. Both the datasets are composed of a set of images taken by a 1296×966 pixels 4-channel JAI AD-130 camera mounted on the Bonirob. During the acquisition, the camera pointed downwards on the field and took images with a frequency of 1 Hz.

The first dataset (Dataset $A$) is composed of 700 images and it has been collected in the first growth stage of the plants when both crop and weeds have not yet developed their complete morphological features. The second dataset (Dataset $B$) is composed of 900 images and it has been collected after 4 weeks: plants, in this case, are in an advanced growth stage. From each dataset different subsets have been extracted, each one manually labeled.

The performance of the proposed classification approach has been measured by using two widely used metrics: the mean accuracy (MA, Eq. 2.6) and the mean average precision (MaP, Eq. 2.7):

$$MA = \frac{1}{N}\sum_{n=1}^{N}\frac{T_{pos}+T_{neg}}{T_{pos}+F_{pos}+T_{neg}+F_{neg}} \tag{2.6}$$

$$MaP = \frac{1}{Q}\sum_{q=1}^{Q}AP(q) \tag{2.7}$$

where $T_{pos}$ and $F_{pos}$ are the numbers of true and false positives, $T_{neg}$ and $F_{neg}$ are the numbers of true and false negatives, and $AP(q)$ is the average precision.

The proposed CNNs *cNet* and *cNet* have been implemented and trained by using the open source library TensorFlow [6].

## Vegetation Detection

The first set of experiments is designed to show the performances of the vegetation detection approach that makes use of a conservative NDVI segmentation as initial pixel segmentation (see Sec. 2.2.1).

Different networks, in terms of the amount and sizes of convolutional and fully connected layers, have been tested by using the same training set taken from the dataset $A$. The results are shown in Table 2.1. The best mean average precision and accuracy (96.8% and 91.3%, respectively) is achieved with the biggest networks, composed of two convolutional and two fully connected layers. Nevertheless, the final choice is to use the *sNet1c10-1f20*, being it a perfect trade-off between average time and accuracy. The performance of this network is then compared with the standard NDVI based vegetation detection algorithm for some fixed thresholds (Table 2.2): the results of *sNet* are remarkable since it outperforms NDVI in all cases while it does not depend on any threshold.

## Crop/Weed Classification

To show the classification accuracy of the proposed pipeline, different experiments for both the pixel-wise and blob-wise approaches have been carried out. The results of a comparison among different networks in the case of pixel-wise classification are reported in Table 2.3(a). As described in Sec. 2.2.1, the proposed network is a combination of a *sNet* followed by a *cNet*. Average timing results are reported for sample steps of 1 (i.e., the *cNet* is applied to each active pixel) and 3 pixels (i.e., the *cNet* is applied on a grid with spacing 3 by 3 pixels). The best trade-off in terms of accuracy, precision and computational time is obtained by the combination *sNet1c10-1f20 + cNet2c64-2f192* , where the *cNet* is composed by four layers, equally

**Table 2.1.** Vegetation detection results for different *sNet* networks. The network names follow the convention: $sNet < x > c < y > - < z > f < w >$, where x: number of convolutional layers, y: size of output feature maps, z: number of fully connected layers, w: size of the fully connected layers.

| Net Type | MA | MaP | Average Time[s] |
|---|---|---|---|
| sNet1c10-1f20 | 96.7% | 91.2% | 0.43 |
| sNet1c5-1f10 | 96.6% | 90.8% | **0.34** |
| sNet1c20-1f40 | 96.7% | 91.2% | 0.45 |
| sNet2c10-2f20 | **96.8%** | **91.3%** | 1.05 |
| sNet2c5-2f10 | **96.8%** | **91.3%** | 0.98 |
| sNet2c20-2f40 | **96.8%** | **91.3%** | 1.23 |

**Table 2.2.** Comparison between the NDVI threshold based vegetation detection and the *sNet1c10-1f20*

| Net Type | *sNet* | $NDVI_{160}$ | $NDVI_{170}$ | $NDVI_{180}$ | $NDVI_{190}$ | $NDVI_{200}$ |
|---|---|---|---|---|---|---|
| Mean Accuracy | **96.7%** | 90.2% | 95.6% | 96.4% | 95.2% | 92.3% |

divided into two convolutional and two fully connected layers. This network reaches a MaP of 96.1% with a lower computational time with respect to the others. A further comparison is made between the combinations of *sNet* and *cNet* with the *cNet* network used alone. In this case the *cNet* has to be applied to the whole image, and the complete image classification is done in 23 seconds without any significant increase in precision. Examples of pixel-wise and grid classification are shown in Fig. 2.4(a) and 2.4(b).

Table 2.3(b) reports classification performance obtained using the proposed blob-wise classification algorithm (Seq. 2.2.2). The results are remarkable since the reported statistics refer only to the image pixels classified as vegetation by the *sNet*. These results are obtained without employing any plant position prior. Moreover, the timing results meet the real-time constraints required by the proposed system. Some qualitative results are reported in Fig. 2.4(c).

### Unsupervised Dataset Summarization

This last part report the performance evaluation of the unsupervised dataset summarization algorithm. To do so, the pixel-wise classification results are compared by using a CNN similar to the *cNet* depicted in Fig. 2.3, with a descriptor size of 384 entries (i.e. the size of the first fully connected layer). The chosen subset size is composed of $K = 50$ images for both dataset $A$ and dataset $B$: The CNN is trained by using $K$ randomly chosen images taken only from the dataset $A$ (*cNetRandomA* in Table 2.4) and $K$ randomly chosen images taken only from the dataset $B$ (*cNetRandomB* in Table 2.4). The training steps are repeated by using $K$ images manually chosen from the dataset $A$ (*cNetManualA* in Table 2.4) and $K$ images manually chosen from the dataset $B$ (*cNetManualB* in Table 2.4): in both cases, the proposed architecture looks for subsets that well represent the original dataset. Finally, the

**Table 2.3.** Classification results for different *cNet* networks. The network names follow the convention: $cNet < x > c < y > - < z > f < w >$, where x: number of convolutional layers, y: size of output feature maps, z: number of fully connected layers, w: size of the fully connected layers.

(a) Pixel-wise and 3×3 grid classification

| Net Type | MA | $\mathrm{MA}_{3\times3}$ | MaP | $\mathrm{Map}_{3\times3}$ | T[s] | $\mathrm{T}_{3\times3}$[s] |
|---|---|---|---|---|---|---|
| scNet2c64-2f192 | 92.3% | 93.3% | 96.2% | 95.6% | ∼200 | ∼23 |
| sNet1c10-1f20 + cNet2c64-2f192 | 91.7% | 91.8% | 96.1% | 94.3% | ∼31 | ∼3.1 |
| sNet1c10-1f20 + cNet2c32-2f100 | 90.8% | 90.7% | 95.2% | 94.1% | **∼30** | **∼2.8** |
| sNet1c10-1f20 + cNet2c96-2f384 | 91.7% | 91.7% | 97.2% | 94.5% | ∼34 | ∼2.9 |
| sNet1c10-1f20 + cNet3c64-3f192 | 91.8% | 91.8% | **97.4%** | **95.7%** | ∼41 | ∼4 |
| sNet1c10-1f20 + cNet3c96-3f384 | **92%** | **91.9%** | **97.4%** | 94.9% | ∼42 | ∼4.3 |

(b) Blob-wise classification

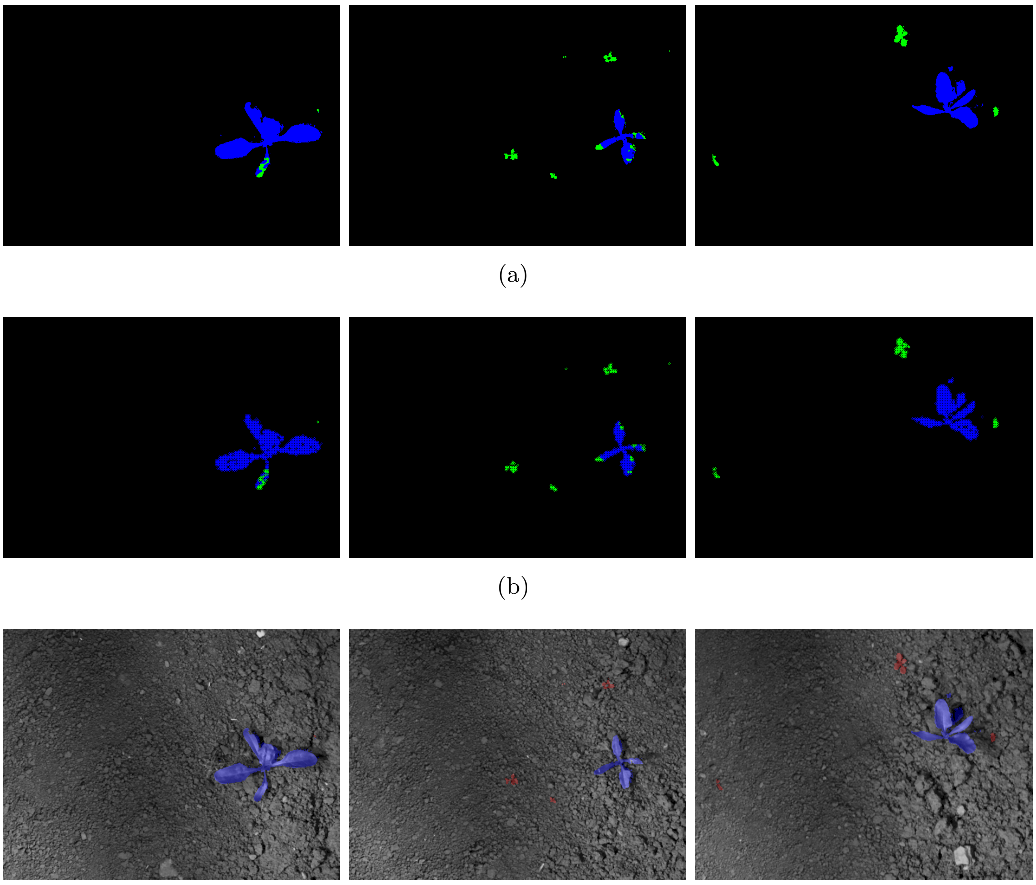
| Net Type | MA | MaP | T[s] |
|---|---|---|---|
| scNet2c64-2f192 | 92.3% | 96.2% | 23 |
| sNet1c10-1f20 + cNet2c64-2f192 | 97.1% | 98.3% | 0.99 |
| sNet1c10-1f20 + cNet2c32-2f100 | 95.6% | 97.8% | **0.93** |
| sNet1c10-1f20 + cNet2c96-2f384 | 97.2% | 98.3% | 1.02 |
| sNet1c10-1f20 + cNet3c64-3f192 | **98%** | 98.3% | 1.74 |
| sNet1c10-1f20 + cNet3c96-3f384 | **98%** | **98.7%** | 2.01 |

CNN is trained by using the automatically selected subsets obtained by applying Algorithm 2 (*cNetUdsA* and *cNetUdsB*), where *cNetRandomA* and *cNetRandomB* are used as pre-trained CNNs, respectively, and a vocabulary of $W = 4096$ visual words. In addition, cross-validation of the trained CNNs is carried out, evaluating a CNN trained with the dataset $A$ with a validation set extracted from dataset $B$, and vice versa. As shown in Table 2.4, the network trained by using subsets selected by the proposed unsupervised dataset summarization algorithm outperform in all the evaluations the network trained with the manually and the randomly chosen training sets, in both datasets A and B. The relatively poor classification results (59.4%) obtained with a CNN tested with a subset of the dataset $B$ and trained using samples taken from dataset $A$ is due to the fact that the dataset $A$ includes only plants that are in their first growth stage, thus without their complete morphological features.

Globally, the results are comparable with the ones recently reported in [96], obtained using the same datasets but, differently from [96], the proposed architecture *does not* exploit any row arrangement. It is expected to obtain even better results by integrating also this type of information.

**Table 2.4.** Pixel-wise classification performances comparison for both datasets $A$ and $B$ for a *cNet* trained with different trainings sets.

| TrainSet & Dataset | MaP A | MaP B |
|---|---|---|
| cNetRandomA | 94.5% | 57.7% |
| cNetManualA | 95.4% | 57.9% |
| cNetUdsA | **96.1%** | **59.4%** |
| cNetRandomB | 78.1% | 97.5% |
| cNetManualB | 79.1% | 98.6% |
| cNetUdsB | **82.3%** | **99.4%** |



(a)

(b)

(c)

**Figure 2.4.** (a)(b) Pixel-wise and 3x3 grid-based classification mask outputs from the sNet1cm1fm + cNet2cm2fm network: in black, green and blue are represented, respectively, pixels that belong to soil, weed, and crop; (c) Final blob-wise classification outputs from sNet1cm1fm + cNet2cm2fm network: pixels that belong to crop are highlighted in violet, pixels that belong to weeds are highlighted in red.

### Manual Annotation Effort Reduction

The reduction in the manual annotation effort $E$ is easily quantifiable by the subset size $K$ and the whole training dataset size through $E = (1 - \frac{K}{size(Dataset_i)}) \times 100$. It results to be around 93% for *DatasetA* and around 94.5% for *datasetB*. A different choice of $K$ would trivially lead to a different effort reduction and, in general, an increase in $K$ would lead also to a slightly better classification performance due to an increased number of images in the training subset. The choice of $K$ is, therefore, a trade-off between the network performance and the reduction of the manual annotation effort.

## 2.3. An automatic model based dataset generation for fast and accurate crop and weed detection

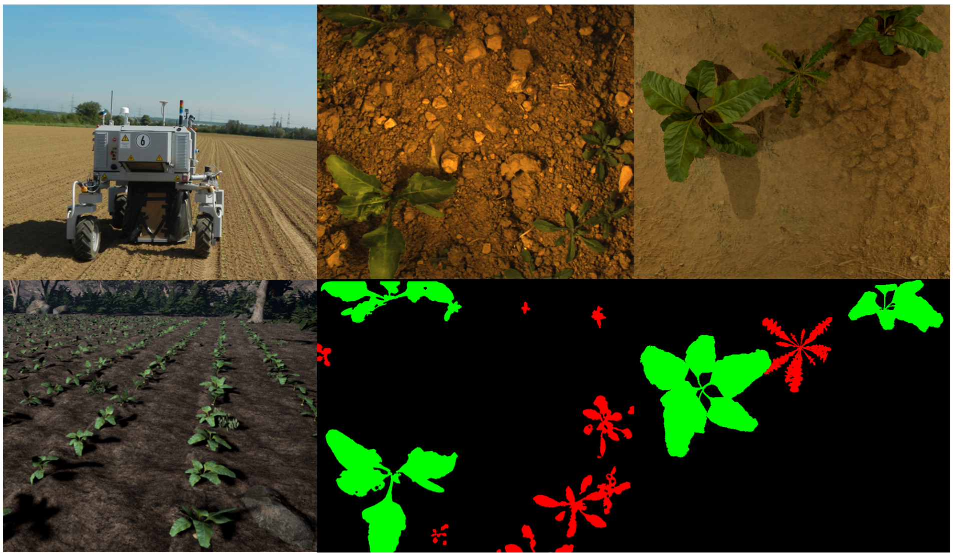


**Figure 2.5.** In the top left, the BOSCH BoniRob employed to acquire the datasets used in the experiments, on top right an artificial sugar beet field generated using the proposed procedure, and on the bottom right the synthetic ground truth annotation mask. In the second column, a real-world image of sugar beets and *Capsella Bursa-Pastoris* with its hand-made ground truth. An example of a synthetic generated field is reported in the bottom left picture.

In this section of the thesis, I explore the use of an open-source graphics engine as a solution for the above-mentioned problem, i.e. by creating data algorithmically. What makes this problem challenging is the realism and the fidelity required to reproduce the key aspects of the target environment, i.e. the virtual scenario must resemble as close as possible to the real one. This requires precise modeling in terms of texture, 3D models and light conditions. Previous work on virtual dataset creation have focused mainly on handcrafted virtual worlds, moving the human effort from the annotation process to the synthetic dataset creation (e.g., [100, 64]). Unlike these approaches, in this thesis I focus on the procedural generation of virtual datasets, allowing us to potentially create an infinite number of synthetic images without any manual labor. More specifically, the proposed method parametrizes the target environment with a set of key rules. Each synthetic scene is then generated by using few real-world textures (e.g., plant leaf textures, soil textures, . . . ) and by

modulating the chosen environmental parameters (e.g., weather conditions, size of plants, . . . ).
The quality[3] of the synthetic datasets is evaluated using them to train a modern deep learning architecture, and then testing it on real data. As the results suggest, even if the virtual scene does not contain all the weed species, the level of accuracy approximates the accuracy reached with real data. An additional test is carried out by performing the same tests training the testbed network with a synthetic dataset augmented with a small number of real images, obtaining even better results. Here the idea was to emulate a real use-case, where only a limited amount of annotated data is available.

### 2.3.1. Procedural Dataset Generation

Procedural generation is a widely used technique in computer graphics, and it has been exploited in several scenarios, such as virtual city generation [54] and virtual dungeons creation [94]. The goal of the procedural dataset generation is to build a randomized rendering pipeline. In this case, the procedural generation can be viewed as a generative model from which fully labeled training images of agricultural scenes can be sampled. The main goals in building this pipeline were twofold: (i) *Realism*, the synthetic agricultural data has to closely resemble the real one, letting the trained model to work in real agricultural environment; (ii) *variety*, the artificial dataset must guarantee a good coverage over the appearance variations, resulting in an unpredictable range of possible scenes. In the following, an exhaustive explanation about how these issues are taken into account is provided. Firstly, the generic kinematic model of a leaf prototype that used to generate leaves of different plant species is described. Then, the entire artificial plant assembling procedure is reported. Finally, the scattering scheme with which the realistic soil and the rendering procedure that allows obtaining at no cost annotated data from the virtual crop field is described.

#### Description of the Leaf Model

The general kinematic model of the simulated leaf is devised as a *kinematic tree* where the root node is the first reference system of the stem, ST1 (see Fig. 2.6). The chain follows the stem direction until the base of the leaf (B), and then the leaf's principal vein, reaching finally the peak (PK). Each joint onto the leaf (VNi) can be used as a starting point for two mirrored branches (STLi, STRi) with respect to the leaf principal axis, following the secondary veins.
The posture of the jth leaf joint with respect to the parent one is parametrized, according to the Denavit-Hartenberg convention, with a single rotation around the z-axis (see Fig. 2.6). The benefit of using such a method is twofold. Firstly, it allows covering a wide variety of crops and weeds realistic leaves in different growth stages just choosing the number of vein joints and their relative distances.
Secondly, acting on the kinematic chain angles as input parameters it is possible to bend the artificial leaf to resemble physical effects such as gravity. In addition,

[3] The term "quality" denotes here the amount and quality of the information transferred to the trained model.

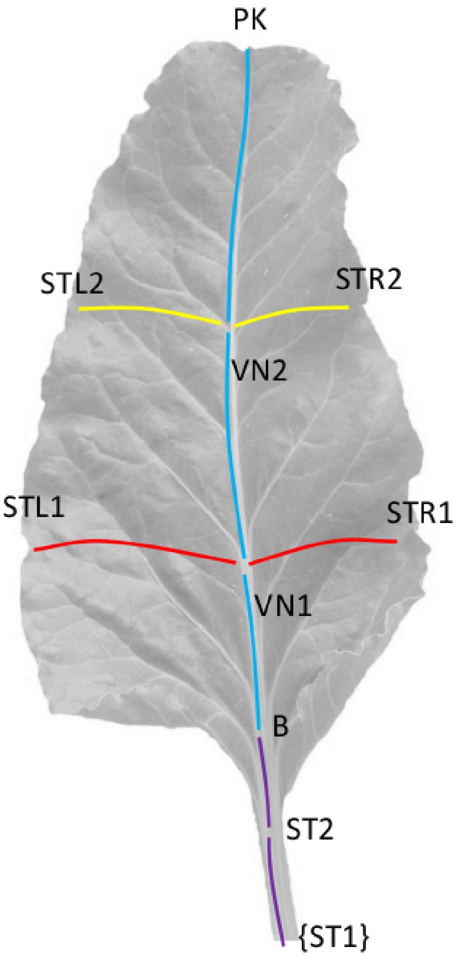


**Figure 2.6.** Leaf kinematic model representing a leaf.

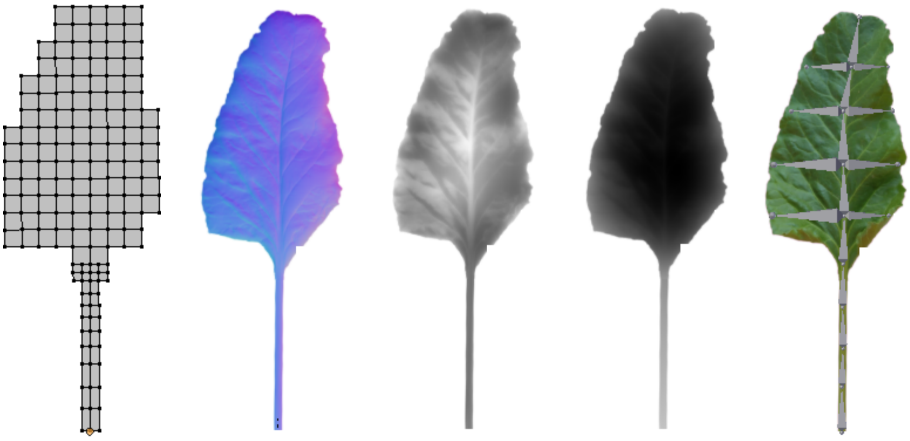


**Figure 2.7.** (a) Shows the generic planar surface mesh. The number of polygons' subdivisions can be kept as low as possible at the expense of deformation quality while acting on the skeleton. (b) Highlights the Normal Map while (c) and (d) respectively represent the Ambient Occlusion and the Heightmap. (e) Shows the final shading results as well as the skeleton.

and most importantly, adding a random component to such angles in the generation phase leads to an unpredictable range of possible leaves.
Once the leaf kinematic model is ready, the skeleton obtained from such a model is associated with a planar surface mesh. The artificial leaf model is then physically-based shaded [126] by means of a high definition RGBA texture taken from real-world pictures. The alpha channel represents the opacity mask.

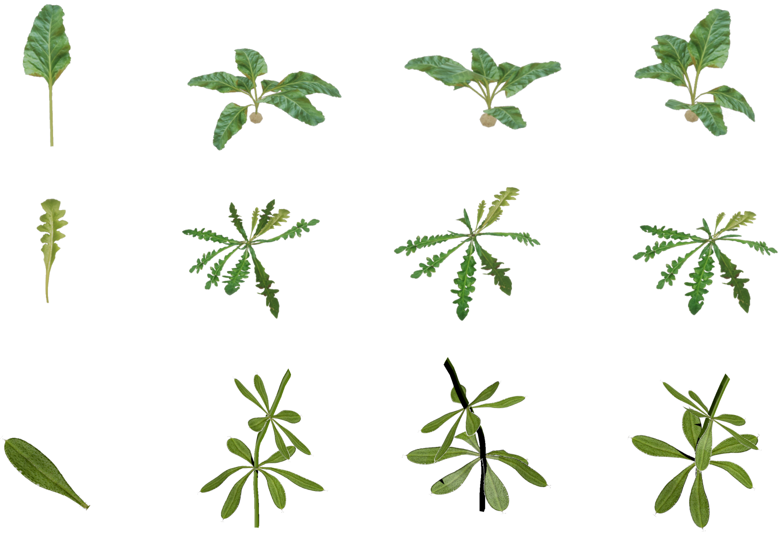


**Figure 2.8.** Examples of three different weeds obtained from a single leaf texture. In the top row three instances of sugar beets, in the second row three different *Capsella Bursa-Pastoris* weeds, in the last row three *Galium Aparine* specimens.

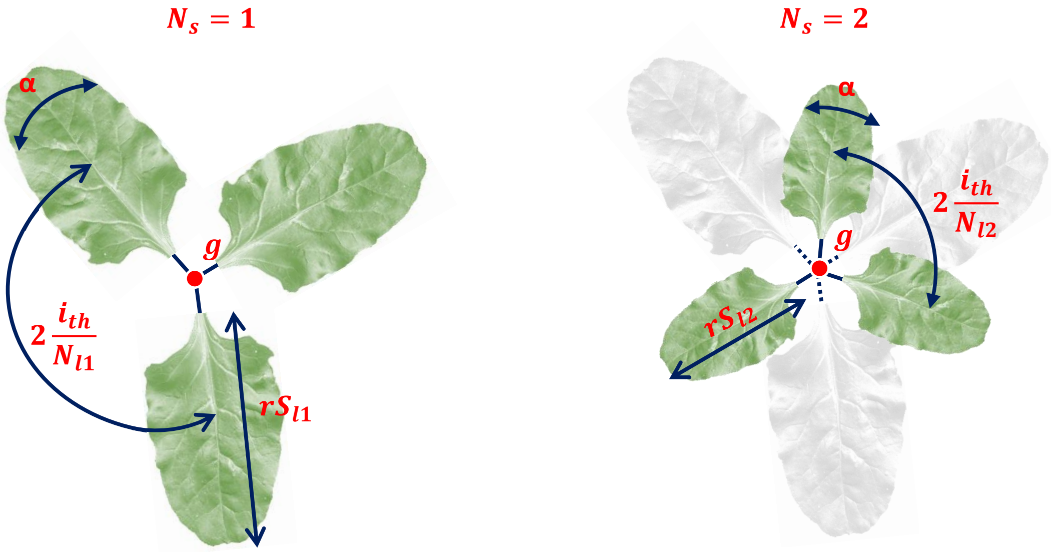


**Figure 2.9.** (a) Shows an example of a plant composed by a single layer leaves, while (b) represents the case of two layers arranged around the growth-axis $g$.

To enhance the rendering quality as well as the real-world fidelity, the proposed method exploits an approximation of the Normal Map (used to simulate high-resolution details on a low-resolution model, see Fig. 2.7(b)), an Ambient Occlusion map (used to approximate how bright light should be shining on any specific part of a surface, see Fig. 2.7(c)) and a Height Map (used to provide extra definition to the leaf model, see Fig. 2.7(d)).

### Plant Modeling

The modeling of an artificial plant follows a procedural scheme too. In order to resemble as much as possible the semblance of real plants, the synthetic plant model needs to take into account properties such as the average number of leaves per plant, their relative distribution, and the number of leaf layers. The whole set of plant species is thus modeled with a generic multi-layer radial distribution of leaves.

According to fig. 2.9, an artificial plant is composed of three main entities:

- *growth stage axis*: it is the axis (called $g$ in Fig. 2.9) around which the leaves are placed. Such axis is parametrized by the 3D position $p$ in the scene and the relative direction $d$ with respect to the gravity vector.

- *layers number*: the number of leaf layers $N_s$, which is usually related to the growth stage.

- *leaves per layer*: an average number of $N_l$ leaves is distributed around the growth stage axis, where $N_l$ depends on the specific plant species. The ith leaf is placed with an offset of $2\frac{i}{N_l} + \alpha$, where $\alpha$ represents a random component used to diversify the leaf arrangement.

- *size of leaves per layer*: as the previous parameters, the size $S_l$ is constrained to the age of the specimen. Conversely to the *layers number*, a random variation is intrudoced on the leaf size by means of a multiplicative factor $r * S_l$

Among the above-mentioned parameters, the number of leaves per layer $N_l$, the number of the layers $N_s$ and the size $S_l$ are kept with the same values, which depends on the plant species, choosing them accordingly to [103].
In this way, modulating the remaining parameters $(p, d, \alpha, r)$, the proposed system is able to procedurally generates a large variety of realistic crop and weeds instances. In order to give a further increment to the virtual environment variation, in each dataset, plants belonging to two different growth stages are generated. Since the initial growth stage is the most important in the targeted treatments, it is noteworthy to highlight how this model allows covering most of the plant species that can be found in a real agricultural environment. Examples of different instances of virtual weeds obtained following the proposed generation procedure are shown in Fig. 2.8.

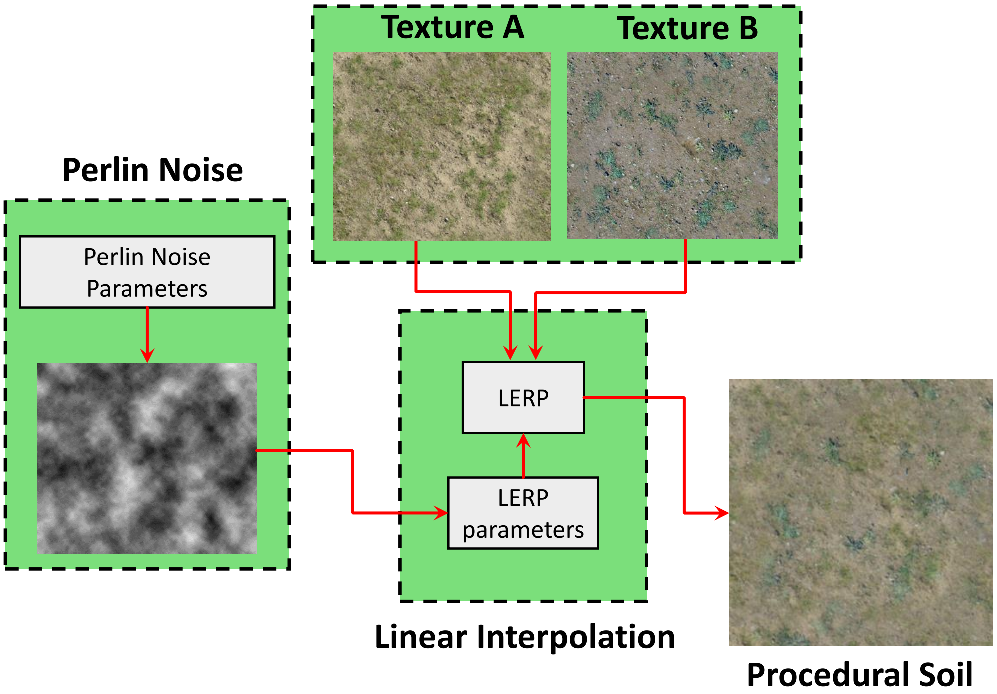


**Figure 2.10.** Example of terrain generation using two inputs. The image shows how two textures are blended together. A linear interpolation node has two textures and an intensity image as inputs. The intensity image is generated via Perlin Noise. The linear interpolation node blends the input images using the intensity input as a selector.

### Virtual Dataset Generation

The final step required to render a realistic agricultural scene is represented by the virtual environment generation procedure, which consists of the following parts: soil generation, lightning, and plants spawning. The ground is simulated as a simple planar surface, wherefrom the fragments point of view different real-world textures have been used (see Fig. 2.10). Each texture can represent different kinds of soil, i.e

dirt, cracked dirt, stony..., and for each soil texture, the Normal Map, the Ambient Occlusion and the Height Maps are generated. These textures are blended together via Perlin Noise linear interpolation [122] to finally generate the soil, as shown in Fig. 2.10.
Following a similar blending procedure between Height Maps and the Normal Maps of the respective textures, a realistic vertices displacement is obtained. By modulating the Perlin Noise parameters, it becomes possible to actually generate different configurations of the terrain both on the fragments and vertices' point of view. To resemble the light conditions of real-world datasets, different light sources can be placed into the scene. In this way, the choice of the light source directly affects the scene illumination conditions, resulting in different shadowing behaviors.
Objects, such as plants, weeds, rocks, sticks.., are then spawned around the scene via a random distribution. The normal direction of the soil surface sampled in the plant position is then used as the initial growth stage direction corrupted by a small white noise perturbation. The whole scene is finally projected into the image plane, using intrinsic and extrinsic camera parameters as close as possible to the real camera used for the actual crop/weeds detection.

#### Artificial Ground Truth Generation

To generate the ground truth labeled images, the target object material is set as unlit and, as an emissive component, it is just required to put the color of the belonging class. Thus, just turning off the anti-aliasing on the camera and all the lights in the scene, it is possible to easily obtain the annotation mask required to train the model. An example of a final rendered sample is shown in Fig. 2.11, where (a) is the artificial terrain RGB image and (b) is the synthetic ground truth.

### 2.3.2. Experiments

The experiments reported in the following are designed to support the claims made in section2.3: a synthetic dataset obtained by means of a procedural generative model and few real-world textures can be used to train a modern machine learning framework, obtaining comparable results with respect to the same framework trained with a real dataset. This result enables a dramatic reduction of the human effort required to acquire and label real data. Furthermore, as the results suggest, a synthetically generated dataset can also be used to supplement small real datasets, enabling cutting-edge results.

#### Experimental Setup

Two datasets are considered, both collected from a BOSCH Bonirob farm robot (Fig. 2.5) moving on a sugar beet field. Both the datasets are composed of a set of images taken by a 1296×966 pixels 4-channels (RGB-NIR) JAI AD-130 camera mounted on the Bonirob. During the acquisition, the camera pointed downwards on the field and took images with a frequency of 1 Hz.
The first dataset (*Real A*) is composed of 700 images and it has been collected in the first growth stage of the plants when both crop and weeds have not yet developed their complete morphological features. The second dataset (*Real B*) is composed of

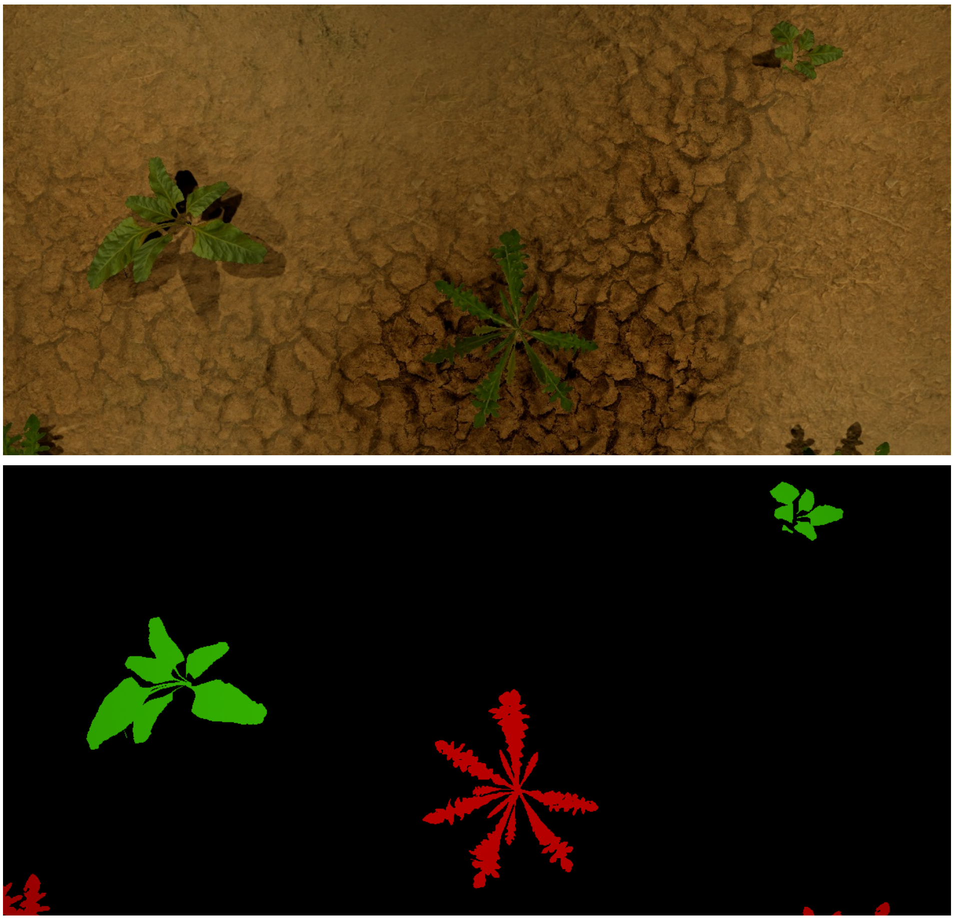


**Figure 2.11.** (a) Shows an example of a generated RGB image the proposed system is capable to provide. In (b) each class has been labeled by using a different color. In the example, the system automatically highlights in green the objects that belong to the *sugar beet* class, in red the *weeds* (like the *Capsella Bursa-Pastoris*) while the soil is turned to black.

900 images and it has been collected after 4 weeks: plants, in this case, are in an advanced growth stage. Each dataset has been manually labeled: the annotation procedure typically takes 5 to 30 minutes per image.

For the procedural dataset generation, the proposed approach exploits the Unreal Engine 4[4] as the graphic engine. Since the proposed architecture aims to make a direct comparison against the real-world data, the evaluation has been carried out by using a generated synthetic dataset of comparable size with respect to the real ones. 4 synthetic datasets are rendered in total, each one composed of 1300 images, using the method presented above: (i) The first one (called *Synthetic A* in the tables) includes only sugar beet plants and some random weeds. (ii) *Synthetic B* includes sugar beet plants and many instances of the *Capsella Bursa-Pastoris* weed, that is the most common weed found in the real datasets. (iii) In *Synthetic C*

[4] Unreal Engine 4 is a complete open-source creation suite for game developers https://www.unrealengine.com/en-US/blog.

**Table 2.5.** Image segmentation results.

| | | Test Set A | | | Test Set B | | |
|---|---|---|---|---|---|---|---|
| Net Variant | Train Set | GA | CA | I/U | GA | CA | I/U |
| RGB SegNet | Real A | 98.6 | 59.6 | 57.5 | 94.3 | 56.0 | 51.7 |
| | Real B | 94.7 | 67.1 | 64.3 | 96.2 | 72.3 | 66.2 |
| | Synthetic A | 99.2 | 66.8 | 47.4 | 96.3 | 74.9 | 55.4 |
| | Synthetic B | 97.8 | 62.2 | 53.3 | 95.5 | 60.2 | 55.1 |
| | Synthetic C | 98 | 61.3 | 53.9 | 96.1 | 59.7 | 55.3 |
| | Synthetic D | 97.6 | 62.5 | 53.1 | 95.6 | 60.1 | 55.2 |
| | Real-Augmented | 98.1 | 63.7 | 52.6 | 96.3 | 59.6 | 55.5 |
| RGB Basic SegNet | Real A | **99.7** | **88.9** | **80.3** | 97.5 | 88.4 | 72.4 |
| | Real B | 99.6 | 82.1 | 72.9 | 98.1 | 91.0 | **83.1** |
| | Synthetic A | 99.4 | 80.3 | 58.9 | 96.4 | 80.9 | 58.3 |
| | Synthetic B | 99.5 | 86.8 | 60.2 | 96.8 | 83.3 | 58.3 |
| | Synthetic C | 99.5 | 70.2 | 55.1 | 96.5 | 55.9 | 52.5 |
| | Synthetic D | 99.6 | 78.7 | 59.8 | 96.7 | 84.2 | 61.1 |
| | Real-Augmented | 99.6 | 84.6 | 74.1 | **99.8** | **91.3** | 76.2 |

another kind of weed called *Galium Aparine* is included. (iv) *Synthetic D* contains sugar beet plants and all the aforementioned weed species. To address the case of a limited amount of annotated data, an additional dataset called *Real-Augmented* has been generated. This dataset is assembled by adding a random sample of 100 images from *Real A* and 100 images from *Real B* to *Synthetic D*. Since simulating in a realistic manner the effect of the light reflection in the Near Infrared (NIR) channels has never been addressed in the literature, only the RGB data is considered for the experiments. As test sets, two subsets removed from *Real A* and *Real B* have been used (called *Test Set A* and *Test Set B* in the tables, respectively). All the images have been resized to 480 × 360 pixels. The performance has been measured by exploiting widely used metrics: global classification accuracy (acronym *GA* in the tables) that provides the number of correct predictions divided by the number of all predictions; per-class average accuracy (*CA*) that provides the mean over all classes of the number of correct predictions made for a specific class divided by the actual number of samples of this class; average intersection over union (*I/U*, see Eq. 2.8); precision and recall (P and R, see Eq. 2.9):

$$I/U = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{T_{pi}}{T_{pi}+F_{pi}+F_{ni}}\right) \tag{2.8}$$

$$P = \frac{T_{pi}}{T_{pi}+F_{pi}}, R = \frac{T_{pi}}{T_{pi}+F_{ni}} \tag{2.9}$$

where $i$ represents the class number, $N$ is the total number of classes, $T_{pi}$, $F_{pi}$ and $F_{ni}$ the number of true positives, false positives and false negatives, respectively, for the class $i$. Finally, all metrics are multiplied by 100 to turn them into percentages.

### SegNet

Differently from my previous work described in Sec.2.2, the performance, and quality of the generated synthetic databases have been evaluated by using an effective pixel-wise classification CNN, *SegNet* [12]. The main difference with respect the CNN described in 2.2, where the entire image is first divided into patches and then classified blob-wise with a voting scheme, *SegNet* makes a pixel-wise semantic classification. This leads to higher accuracy, since *SegNet* is able to discriminate even in the case of overlapping plants. Such advantages make *SegNet* particularly suitable for the purposes of this work. For the evaluation phase, a smaller version of *SegNet* called *Basic SegNet* has been used. While the former is composed of 26 layers, the latter has only 8 layers. The main motivation that stands behind the choice of two different variants of the same network is the avoidance of overfitting. The bigger network has been developed for semantic segmentation of a large number of classes, prone to overfit in case of a small number of output classes.

### Crop/Weed Classification and Vegetation Detection

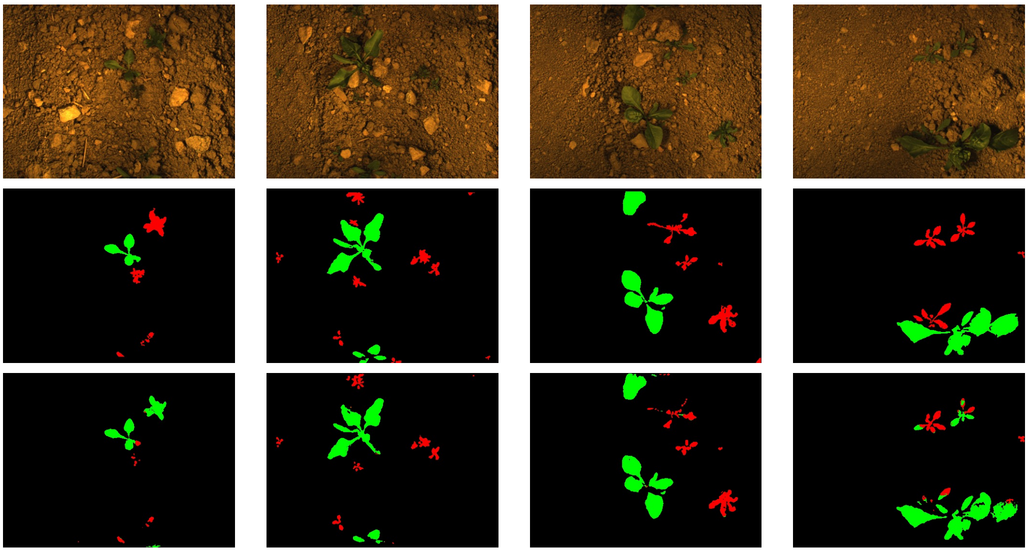


**Figure 2.12.** Some examples of real image segmentation using a *RGB Basic SegNet* trained using synthetic datasets. First row: input real RGB images. Second row: ground truth labels. Third row: segmentation results.

The first set of experiments compare the performance among different crop/weeds classifiers. The difference is made in terms of training dataset and network typology, while the comparison is made by applying the trained models on both *Test Set A* and *Test Set B*. Table 2.5 reports numerical results of the pixel-wise soil-crop-weeds classification, while in Fig. 2.12 some qualitative results are depicted. In this experiment, the most significant metrics are the per-class average accuracy and the average intersection over union. The global accuracy can be sometimes a misleading

**Table 2.6.** Vegetation detection results

| | | Test Set A | | Test Set B | |
|---|---|---|---|---|---|
| Net Variant | Train Set | P | R | P | R |
| RGB SegNet | Real A | 99.0 | 47.7 | 97.1 | 40.8 |
| | Real B | 98.2 | 52.9 | 94.9 | 69.0 |
| | Synthetic A | 58.0 | 90.4 | 53.8 | 49.3 |
| | Synthetic B | 67.5 | 49.6 | 71.7 | 68.5 |
| | Synthetic C | 63.2 | 64.2 | 73.8 | 75.6 |
| | Synthetic D | 67.5 | 67.6 | 77.9 | 76.9 |
| | Real-Augmented | 68.1 | 65.1 | 83.3 | 84.2 |
| RGB Basic SegNet | Real A | 89.2 | 98.4 | 90.3 | 98.6 |
| | Real B | 90.7 | 93.1 | 91.0 | 94.5 |
| | Synthetic A | 80.9 | 90.9 | 78.5 | 86.2 |
| | Synthetic B | 81.3 | 94.1 | 79.9 | 89.2 |
| | Synthetic C | 77.6 | 91.4 | 69.2 | 87.1 |
| | Synthetic D | 82.2 | 94.0 | 80.4 | 89.7 |
| | Real-Augmented | 83.1 | 94.2 | 83.6 | 92.8 |

**Table 2.7.** Infer runtime

| Net Variant | time (s) |
|---|---|
| RGB SegNet | 0.14 |
| RGB Basic SegNet | 0.08 |

metric: for instance, in the considered test sets the majority of pixels represents soil in the scene, hence also a classifier that predicts only soil will still obtain good GA performance.

The best performance are generally obtained by the *RGB Basic SegNet* network trained with real datasets (*Real A* and *Real B*) and with the *Real-Augmented* dataset. Particularly, the latter often outperforms the real ones, achieving very good average intersection over union results for both *Test Set A* and *Test Set B* and a remarkable per-class average accuracy of 91.3% on *Test Set B*.

Moreover, also the results obtained by *RGB Basic SegNet* trained *solely* with the synthetic datasets are noteworthy, especially for the *Synthetic B* and *Synthetic D* datasets. These surprising results are also implicitly confirmed in [64], where the authors claimed that using synthetic data can outperform models trained on a limited amount of real scene-specific data. Another interesting result is the significant performance difference between *Synthetic B* and *Synthetic C*. The main motivation that probably stands behind these fluctuating results is the actual weed distribution in the real-world datasets. Indeed, in such data, the *Capsella Bursa-Pastoris* plays the role of the most common weed while there are just a few instances of *Galium Aparine*.

It is also important to note how the incremental inclusion of different weed species in the artificial datasets yields a monotone positive trend in the classification performance, converging toward the real-data performance when real data is included in the training dataset. This fact implicitly confirms the goodness of the proposed architecture.

On the other hand, the results obtained by the complete SegNet version *RGB SegNet* are generally poor, also for real data, mainly due to overfitting phenomena.

Table 2.6 reports the numerical results of the pixel-wise soil-vegetation classification. Also, this evaluation confirms the previous results: also, in this case, the synthetically generated datasets are able to approximate the results obtained using the real datasets, while as before *RGB Basic SegNet* performs much better in all cases compared with *RGB SegNet*.

Here is noteworthy to highlight that the information coming from the NIR channel is not exploited. Due to the high reflectance of green plants in this spectrum, this information is strictly related to the vegetation detection: specific models to synthetically generate the NIR channel are currently under investigation.

Finally, Table 2.7 reports the infer time of each network using an NVidia GPU GTX 1070.

### Manual Annotation Effort Reduction

Also in this case, the reduction in the manual annotation effort $E$ can be derived by through $E = (1 - \frac{K}{size(Dataset_i)}) \times 100$. Particularly, for datasets *Real A* and *Real B*, it can be derived by assuming that the number of real images is equal to the subset size $K$ described in Sec. 2.2.2. By doing so, the reduction in the manual annotation effort results to be around 85.7% for *Real A* and around 88.8% for *Real B*.

## 2.4. Summary of Results and Lessons Learned

This chapter of the thesis focuses on perception challenges that arise when developing crop/weed segmentation modules for farming robotics. Besides proposing two different Convolutional Neural Networks to perform plant segmentation, two important contributions have been proposed, namely "*using a training dataset summarization technique*" and "*generating realistic synthetic data to perform a better data augmentation*". The purpose is to reduce the image annotation effort while keeping a sufficient segmentation accuracy to perform agronomic interventions and weed control. In particular, the first proposed architecture uses inner features from a pre-trained network to summarize a large real-world dataset while keeping ad adequate level of informativeness, *i.e.* ensuring that the summarized dataset allows the network to properly generalize the segmentation problem. Conversely, the second proposed approach addresses the problem by generating a huge number of realistic images through a modern graphics engine. Through the analysis of the segmentation accuracy, the reported experiments show how a CNN trained with summarized or synthetic datasets allows for a high segmentation accuracy.

These procedures provide simple and effective ways to deal with the big annotation effort involved in the annotation of large datasets and, moreover, these concepts can be applied in other robotics contexts.

# Chapter 3

# Mapping and Localization

The crop/weed perception module introduced in the previous chapter enables a farming robot to measure crop health indicators. However, to carry out a targeted weeding or a disease control of the field, the semantic information provided by the former module should be fed inside a map of the target environment. Indeed, such a map would provide the user with a general overview of the crop status, and help the farmer with high-level decision making. Moreover, locating the infested areas allows the robot to perform targeted treatments in a fully autonomous manner. Therefore, this chapter focuses on robot mapping and localization module.

Those two complementary capabilities are strictly required, and almost every robot with autonomous navigation capabilities is equipped with this module. Specifically, in a farming scenario, an accurate global pose estimation system is an essential component to handle several tasks: (i) navigation and path planning; (ii) autonomous ground intervention; (iii) acquisition of relevant semantic information. However, self-localization inside an agricultural environment is a complex task: the scene is rather homogeneous, visually repetitive and often poor of distinguishable reference points. For this reason, conventional landmark-based SLAM architectures are likely to yield poor results. On the other hand, most farming navigation systems rely on high-end Real-Time Kinematic Global Positioning Systems (RTK-GPSs) to localize the UGV on the field with high accuracy [162][113]. This technology bases its great accuracy (*i.e.* a few centimetric errors) on geo-localized ground stations that calculate the estimation error. The latter is then sent to the GPS rover stations which are mounted on the moving platforms aiding their localization estimation.

The relatively high cost and the effort in setting up the base station usually prevented this technology to be used in real farming applications in the past. However, in the last years, its cost became affordable while some services providing the correction signals wirelessly becomes available. An example of this service is located in NRW/Germany and is called SAPOS [1]. Despite this technology presents fewer drawbacks nowadays, it is still vulnerable to outages or signal loss. In addition, the wireless signal is penalized in some specific environments, such as in hazelnut precision farming, where trees and their canopy may block the wireless signal.

On the other hand, consumer-grade GPSs[1] usually provide noisy data, thus not

[1] In this section, the term GPS is used as a synonym of the more general acronym GNSS (Global Navigation Satellite System) since almost all GNSSs use at least the GPS system, included the two

guaranteeing enough accuracy and reliability for safe and effective operations. Moreover, a GPS cannot provide the full state estimation of the vehicle, i.e. its attitude, that is an essential piece of information to perform a full 3D reconstruction of the environment.

In this chapter, I present one of the major contribution of this thesis which addresses the self-localization of farming robots. The problem is tackled through a multi-sensor fusion approach, and the results show how the method allows the robot to safely navigate within the environment. Moreover, in this study, a benchmark and assessment of different sensors in the precision farming context are presented, highlighting both their weaknesses and strengths.

## 3.1. Related Work

The problem of global pose estimation for UGVs has been intensively investigated, especially in the context of self-driving vehicles and outdoor autonomous robots moving in urban environments. The task is commonly approached by integrating multiple sources of information. Most of the state-of-the-art systems rely on IMU-aided GPS [44], while they differ in the other sensor cues they use in the estimation process. Cameras are used primarily in [141][120][131][159][142], while LIDARs have been used in [87].

In urban scenarios, the presence of a prior map allows improving the estimation by constraining the robot motion. [50][22] use 2D road maps, while [38] propose to use more rich Digital Elevation Models (DEMs). The sensors fusion is usually carried out by means of parametric [50] or discrete [87] filtering, pose graph optimization [131][159], set-membership positioning [38], or hybrid topological/filtering [141].

As stated in the introduction, these approaches cannot be used effectively in agricultural environments, since a prior map is typically not available. In addition, crops exhibit substantially a less stable structure than an urban environment, and their appearance varies substantially over time. Hence, the localization inside an agricultural field, by using a map built on-line, turns out to be extremely difficult since stable features are hard to find. For this reason, most of the available localization methods for farming robots are based on expensive global navigation satellite systems [151][155][113]. However, relying on the GPS as the primary localization sensor exposes the system to GPS related issues: potential signal losses, multi-path, and a time-dependent accuracy influenced by the satellite positions.

The main task of an agricultural robot is to follow the crop rows and take some action along the way. To this extent, English *et. al* [40], proposed a vision based crop-row following system. While effective, this system assumes that the crops are clearly visible from the camera of the robot, and this is not true at all growth stages of the plants. Furthermore, the estimate of a crop row tracking tends to accumulate drift along the row direction.

To gain robustness and relax the accuracy requirements on the GPS, it is natural to use the plants as landmarks to build a map using a SLAM algorithm. To this extent, Cheein *et al.* [29] propose to find and to use as landmarks, in a SLAM system, olive tree stems. The stem detection algorithm uses both camera and laser data.

GNSS r used in our experiments.

Other approaches are based on the detection of specific plant species and thus they address very specific use cases. Jin *et al.* [76] focuses on the individual detection of corn plants by using RGB-D data. In [162], the authors propose a MEMS-based 3D LIDAR sensor to map an agricultural environment by means of a per-plant detection algorithm. Gai *et al.* [6] proposed an algorithm that follows leaf ridges detected in RGB images to the center. Similar approaches rely on Stem Emerging Points (SEPs) localizations: Mitdiby *et al.* [105] follow sugar beet leaf contours to find the SEPs. In [65] the authors perform machine learning-based SEP localization in an organic carrot field. Kraemer *et al.* [84] proposed an image-based plant localization method that exploits a CNN to learn time-invariant SEPs.

## 3.2. An Effective Multi-Cue Positioning System for Agricultural Robotics

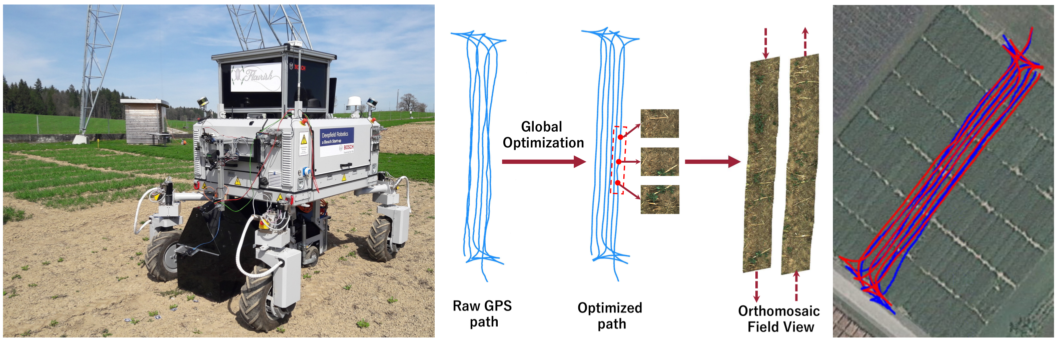


**Figure 3.1.** (Left) The Bosch BoniRob farm robot used in the experiments; (Center) Example of a trajectory (Dataset B, see Sec. 3.2.3) optimized by using our system: the optimized pose graph can be then used, for example, to stitch together the images acquired from a downward looking camera; (Right) The obtained trajectory (red solid line) with respect to the trajectory obtained using only the raw GPS readings (blue solid line). Both trajectories have been over-imposed on the actual field used during the acquisition campaign.

In this section, I present a robust and accurate 3D global pose estimation system for UGVs (Unmanned Ground Vehicles) designed to address the specific challenges of an agricultural environment. Our system effectively fuses several heterogeneous cues extracted from low-cost, consumer-grade sensors, by leveraging the strengths of each sensor and the specific characteristics of the agricultural context. The global localization problem is cast as a pose graph optimization problem (Sec. 3.2.1): the constraints between consecutive nodes are represented by motion estimations provided by the UGV wheel odometry, local point-cloud registration, and a visual odometry (VO) front-end that provides a full 6D ego-motion estimation with a small cumulative drift[2]. Noisy, but drift-free GPS readings (i.e., the GPS *pose solution*), along with a pitch and roll estimation extracted by using a MEMS Inertial Measurement Units (IMU), are directly integrated as prior nodes. Driven by the fact that both GPS and visual odometry provide poor estimates along the $z$-axis, i.e. the

[2] In VO open-loop systems, the cumulative drift is unavoidable.

axis parallel to the gravity vector, the proposed localization and mapping architecture improve the state estimation along this challenging direction by introducing two additional altitude constraints:

1. An altitude prior, provided by a Digital Elevation Model (DEM);

2. A smoothness constraint for the altitude of adjacent nodes[3].

Both the newly introduced constraints are justified by the assumption that, in an agricultural field, the altitude varies slowly, i.e. the soil terrain can be approximated by piecewise smooth surfaces. The smoothness constraints exploit the fact that a farming robot traverses the field by following the crop rows, hence, by using the Markov assumption, the built pose graph can be arranged as a Markov Random Field (MRF). The motion of the UGV is finally constrained using an Ackermann motion model extended to the non-planar motion case. The integration of such constraints not only improves the accuracy of the altitude estimation but it also positively affects the estimate of the remaining state components, i.e. $x$ and $y$ (see Sec. 3.2.3).
The optimization problem (Sec. 3.2.2) is then iteratively solved by exploiting a graph-based optimization framework [88] in a sliding-window (SW) fashion (Sec. 3.2.2), i.e., optimizing the sub-graphs associated with the most recent sensor readings. The SW optimization allows obtaining on-line localization results that approximate the results achievable by an off-line optimization over the whole dataset.
In order to validate our approach (Sec. 3.2.3), I used and made publicly available with this work two novel challenging datasets acquired using a Bosch BoniRob UGV (Fig. 3.1, left) equipped with, among several others calibrated sensors, two types of low-cost GNSS receivers: a Precise Point Positioning (PPP) GPS and a consumer-grade RTK-GPS. Exhaustive experiments with several sensors setups are reported, showing remarkable results: the global localization accuracy has been improved up to 37% and 76%, compared with the raw localization obtained by using only the raw RTK-GPS and PPP-GPS readings, respectively (e.g., Fig. 3.1). Moreover, the reported experiments also show that our approach allows localizing the UGV even though the GPS performances temporarily degrade, e.g. due to a signal loss.

### 3.2.1. Multi-Cue Pose Graph

The challenges that must be addressed to design a robust and accurate global pose estimation framework for farming applications are twofold: (i) the agricultural environment appearance, being usually homogeneous and visually repetitive; (ii) The high number of cues that have to be fused together. In this section, the formulation of a robust pose estimation procedure able to face both these issues is presented.

The proposed system handles the global pose estimation problem as a pose graph optimization problem. A pose graph is a special case of factor graph[4], where the factors $\langle \cdot \rangle$ are only connected to variables (i.e., nodes) pairs, and variables are only

[3] The term "adjacent“ denotes nodes that are temporally or spatially close.

[4] A factor graph is a bipartite graph where nodes encode either variables or measurements, namely the factors.

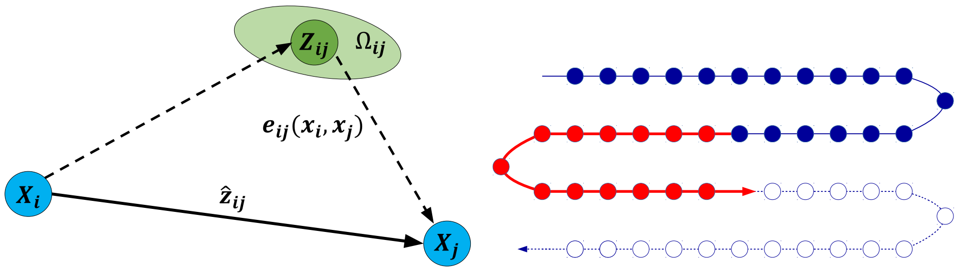


**Figure 3.2.** (Left) Illustration of an edge connecting two nodes $x_i$ and $x_j$. The error $e_{i,j}$ is computed between the measurement $z_{i,j}$ and the predicted measurement $\hat{z}_{i,j}$. In addition, each edge encodes an information matrix $\Omega_{i,j}$ that represents the uncertainty of the measurement; (Right) Sliding-window sub-graph optimization: nodes that belong to the current sub-graph are painted in red, old nodes no more optimized are painted in blue, while nodes that will be added in the future are painted in white.

represented by robot poses. For this reason, it is common to represent each factor with an edge. Solving a factor graph means finding a configuration of the nodes for which the likelihood of the actual measurements is maximal. Since all the involved noises are assumed to follow a Gaussian distribution, this problem can be solved by employing an iterative least square approach.

Let $X = \{x_0, ..., x_{N-1}\}$ represents the vector of graph nodes that represents the robot poses at discrete points in time, where each $x_i = (T_i, R_i)$ is represented by the full 3D pose in terms of a translation vector $T_i = [t_{x,i}\ t_{y,i}\ t_{z,i}]'$ and, using the axis-angle representation, an orientation vector $R_i = [r_{x,i}\ r_{y,i}\ r_{z,i}]'$, both in $\mathbb{R}^3$. This pose is defined with respect to a global reference centered in $x_0$[5]. Let $z$ be the sensor measurements that can be related to pairs or single nodes. Let $z_{i,j}$ be a relative motion measurement between nodes $x_i$ and $x_j$, while $z_i$ be a global pose measurement associated to the node $x_i$. Additionally, let $\Omega_{i,j}$ and $\Omega_i$ represent the information matrices encoding the reliability of such measurements, respectively. From the poses of two nodes $x_i$ and $x_j$, it is possible to compute the expected relative motion measurement $\hat{z}_{i,j}$ and the expected global measurement $\hat{z}_i$ (see Fig. 3.2, left). The errors between those quantities is then formulated as:

$$e_{i,j} = z_{i,j} - \hat{z}_{i,j},\ e_i = z_i - \hat{z}_i, \tag{3.1}$$

Thus, for a general sensor $\mathcal{X}$ providing a relative information, an edge can be characterized (i.e., a *binary factor* $\langle e^{\mathcal{X}}_{i,i+1}, \Omega^{\mathcal{X}}_{i,i+1} \rangle$) by the error $e^{\mathcal{X}}_{i,i+1}$ and the information matrix $\Omega^{\mathcal{X}}_{i,j}$ of the measurement, as described in [60]. In other words, an edge represents the relative pose constraint between two nodes (Fig. 3.2, left). In order to take into account also global pose information, unary constraints are used, namely a measurement that constrains a single node. Hence, for a general sensor $\mathcal{Y}$ providing an absolute information, $\langle e^{\mathcal{Y}}_i, \Omega^{\mathcal{Y}}_i \rangle$ is defined as the prior edge (i.e., an *unary factor*) induced by the sensor $\mathcal{Y}$ on node $x_i$. Fig. 3.3 depicts a portion of a pose graph highlighting both unary and binary edges. Each edge acts as a directed spring with elasticity inversely proportional to the relative information matrix associated with the measurement that generates the link. Our pose graph is built by adding

[5] Each global measurement (e.g., GPS measurements) is transformed in the reference frame $x_0$.

an edge for each sensor reading, for both relative (e.g., wheel odometry readings) and global (e.g., GPS readings) information. In addition, the proposed architecture integrates other prior information that exploits both the specific target environment and the assumptions that have been formerly made. In the following, the full list of edges exploited in this work is reported, and they divided between local (relative) and global measurements (the acronyms used in Fig. 3.3 are reported in brackets):

**Local measurements**: Wheel odometry measurements (WO), Visual odometry estimations (VO), Elevation constraints between adjacent nodes (MRF), Ackermann motion model (AMM), Point-clouds local registration (LID).

**Global measurements**: GPS readings (GPS), Digital Elevation Model data (DEM), IMU readings (IMU)[6].

Let $\langle e^{VO}_{i,i+1}, \Omega^{VO}_{i,i+1} \rangle$ be the relative constraint induced by a visual odometry algorithm, $\langle e^{WO}_{i,i+1}, \Omega^{WO}_{i,i+1} \rangle$ as the relative constraint induced by the wheel odometry, and $\langle e^{LID}_{i,i+1}, \Omega^{LID}_{i,i+1} \rangle$ as the relative constraint obtained by aligning the local point-clouds perceived by the 3D LIDAR sensor.

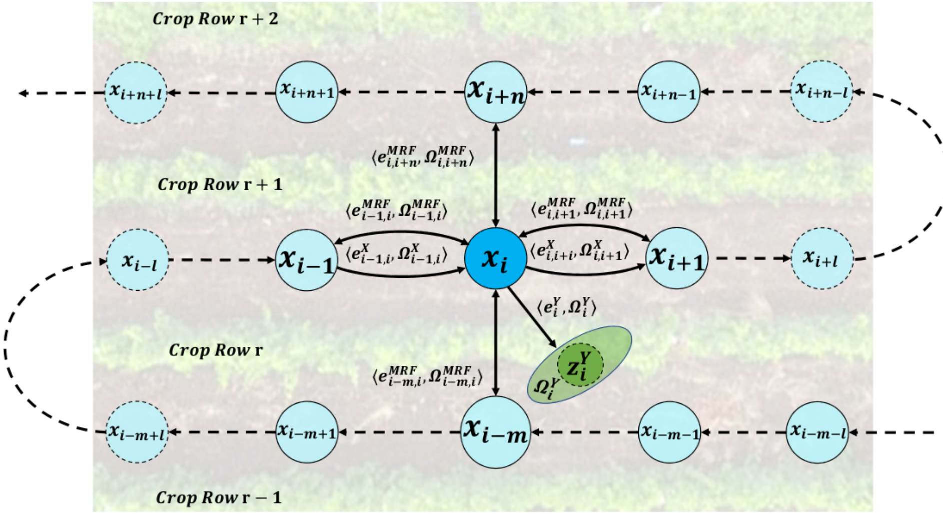

Crop Row r + 2
$x_{i+n+l}$
$x_{i+n+1}$
$x_{i+n}$
$x_{i+n-1}$
$x_{i+n-l}$
$\langle e^{MRF}_{i,i+n}, \Omega^{MRF}_{i,i+n} \rangle$
Crop Row r + 1
$\langle e^{MRF}_{i-1,i}, \Omega^{MRF}_{i-1,i} \rangle$
$\langle e^{MRF}_{i,i+1}, \Omega^{MRF}_{i,i+1} \rangle$
$x_{i-l}$
$x_{i-1}$
$\langle e^{X}_{i-1,i}, \Omega^{X}_{i-1,i} \rangle$
$x_i$
$\langle e^{X}_{i,i+i}, \Omega^{X}_{i,i+1} \rangle$
$x_{i+1}$
$x_{i+l}$
$\langle e^{Y}_{i}, \Omega^{Y}_{i} \rangle$
Crop Row r
$\langle e^{MRF}_{i-m,i}, \Omega^{MRF}_{i-m,i} \rangle$
$z^{Y}_{i}$
$\Omega^{Y}_{i}$
$x_{i-m+l}$
$x_{i-m+1}$
$x_{i-m}$
$x_{i-m-1}$
$x_{i-m-l}$
Crop Row r − 1

**Figure 3.3.** Overview of the built pose graph. Solid arrows represent graph edges, that encode conditional dependencies between nodes, dotted arrows temporal relationships between nodes. For the sake of clarity, only edges directly connected with the node $x_i$ are shown, by representing only one instance for each class of edges: (i) the binary non directed MRF constraint $\langle e^{MRF}_{i,i+1}, \Omega^{MRF}_{i,i+1} \rangle$; (ii) the binary directed edge $\langle e^{\mathcal{X}}_{i,i+1}, \Omega^{\mathcal{X}}_{i,i+1} \rangle$ induced from sensor $\mathcal{X} \in \{VO, WO, AMM, LID\}$; (iii) the unary edge $\langle e^{\mathcal{Y}}_{i}, \Omega^{\mathcal{Y}}_{i} \rangle$ induced by sensor $\mathcal{Y} \in \{GPS, DEM, IMU\}$. The graph is superimposed on a cultivated field to remark the relationship between the graph structure and the crop rows arrangement.

Often, GPS and visual odometry provide poor estimates of the robot position along the $z$-axis (i.e, the axis that represents its *elevation*). In the GPS case, this low accuracy is mainly due to the Dilution of Precision, multipath or atmospheric disturbances, while in the visual odometry this is due to the 3D locations of the

[6] In this approach, the IMU is referred as global measurement due to its properties to estimate the heading angle with respect to the magnetic north and the roll and pitch angles with respect to the Earth's gravity.

tracked points. In a typical agricultural scenario, most of the visual features belong to the ground plane. Hence, the small displacement of the features along the z-axis may cause a considerable drift. On the other hand, agricultural fields usually present locally flat ground levels and, moreover, a farming robot usually traverses the field by following the crop rows. Driven by these observations, the local ground smoothness assumption is enforced by introducing an additional type of local constraints that penalizes the distance along the $z$-coordinate between adjacent nodes. Therefore, the built pose graph can be augmented by a 4-connected MRF [18]: each node is conditionally connected with the previous and the next nodes in the current crop row, and with the spatially closest nodes that belong to the previous and next crop rows, respectively. This constraint is from now on referred as $\langle e^{MRF}_{i,i+1}, \Omega^{MRF}_{i,i+1}\rangle$ in Fig. 3.3 (e.g., the set $\{x_{i-1}, x_i, x_{i+1}, x_{i-m}, x_{i+n}\}$). A further type of local constraint based on the Ackermann steering model is then added. As reported in Sec. [**?** ], the robot used in this work navigates within the environment according to Ackermann steering kinematics. despite this, the local constraint can be formulated to fit, case by case, the specific architecture of the employed farming robot. Particularly, the Ackermann constraint used in the proposed architecture assumes that the robot is moving on a plane. In this work, this assumption is relaxed to local planar motions between temporal adjacent nodes. Such a motion plane is updated with the attitude estimation of the subsequent node. This constraint is integrated by means of a new type of edge, namely $\langle e^{AMM}_{i,i+1}, \Omega^{AMM}_{i,i+1}\rangle$.

Local constraints are intrinsically affected by a small cumulative drift: to overcome this problem, some global measurements are integrated into the graph drift-free as position prior information. In particular, a GPS prior $z_i^{GPS}$ and an IMU prior $z_i^{IMU}$ with associated information matrices $\Omega_i^{GPS}$ and $\Omega_i^{IMU}$ are defined. The IMU is used as a drift-free roll and pitch reference[7], where the drift resulting from the gyroscopes integration is compensated by using the accelerometers data.

Finally, an additional global measurement is introduced by means of an altitude prior, provided by a DEM. A DEM is a special type of Digital Terrain Model that represents the elevation of the terrain at some location, by means of a regularly spaced grid of elevation points [69]. The DEM maps a 2D coordinate to an absolute elevation. Since in this chapter I made the assumption that the altitude varies slowly, the current position estimate $T_i$ (i.e., the $t_{x,i}$ and $t_{y,i}$ components) can be used to query the DEM for a reliable altitude estimation $z_{DEM,i} = f(t_{x,i}, t_{y,i})$, with associated information matrix $\Omega_i^{DEM}$. The cost function is then assembled as follows:

$$J_i = \sum_{i=1}^{N-1} \Big( \underbrace{\sum_{\mathcal{X}} e^{\mathcal{X}}_{i,i-1} \Omega^{\mathcal{X}}_{i,i-1} e^{\mathcal{X}'}_{i,i-1}}_{Binary\ constraints} + \underbrace{\sum_{\mathcal{Y}} e_i^{\mathcal{Y}} \Omega_i^{\mathcal{Y}} e_i^{\mathcal{Y}'}}_{Unary\ constraints} + \underbrace{\sum_{j \in \mathbb{N}_i} e^{MRF}_{i,j} \Omega^{MRF}_{i,j} e^{MRF'}_{i,j}}_{MRF\ constraint} \Big) \tag{3.2}$$

where $\mathcal{X}$ and $\mathcal{Y}$ represent respectively the set of binary and unary constraints defined above (see Fig. 3.3), and $\mathbb{N}_i$ stands for the 4-connected neighborhood of the node $x_i$.

[7] I experienced that integrating the full inertial information inside the optimization did not positively affect the state estimation: our intuition is that the slow, often unimodal, motion of our robot makes the IMU biases difficult to estimate and sometimes predominant over the motion components.

### 3.2.2. Pose Graph Optimization

In this section, the solution of the cost function reported in Eq. 3.2 is explained, describing the error computation, the weighting factors assignment procedure and the on-line and off-line versions of the optimization. Finally, some key implementation insights are reported.

#### Error Computation

For each measurement $z$, given the current graph configuration, a prediction $\hat{z}$ needs to be provided in order to compute errors in Eq. 3.2. $\hat{z}$ represents the expected measurement, given a configuration of the nodes, which are involved in the constraint. Usually, for a binary constraint, this prediction is the relative transformation between the nodes $x_i$ and $x_j$, while for an unary constraint it is just the full state $x_i$ or a subset of its components. Let $\mathbf{X}_i$ be a general homogeneous transformation matrix related to the full state of the node $x_i$ (e.g., the homogeneous rigid body transformation generated from $T_i$ and $R_i$) and $\Phi(\cdot)$ as a generic mapping function from $\mathbf{X}_i$ to a vector; now, $\hat{z}_{i,j}$ and $\hat{z}_i$ can be expressed as:

$$\hat{z}_{i,j} = \Phi(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j), \;\; \hat{z}_i = \Phi(\mathbf{X}_i) \tag{3.3}$$

In this work not all the constraints belong to $SE(3)$: indeed, most of used sensors (e.g., WO, IMU) can only observe a portion of the full state encoded in $x$. Therefore, in the following, the process with which the expected $\hat{z}$ is obtained for each involved cue is presented (for some components, the subscripts $i$ and $j$ are omitted by using the relative translations $dt$ and rotations $dr$ between adjacent nodes):

**VO and LID**: these front-ends provide the full 6D motion: $\hat{z}^{VO}$ and $\hat{z}^{LID}$ are built by computing the relative transformation between the two connected nodes as in Eq. 3.3;

**WO**: the robot odometry provides the planar motion by means of a roto-translation $z_{WO} = (dt_x, dt_y, dr_z)$: $\hat{z}^{WO}$ is built as $\Phi(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j)|_{t_x,t_y,r_z}$, the subscripts after $\Phi(\cdot)$ specify that the map to the vector $\hat{z}$ involves only such components;

**MRF and DEM**: they constrain the altitude of the robot, the estimated measurements are obtained as:

$$\hat{z}_{i,j}^{MRF} = (0, 0, t_{z,i} - t_{z,j}, 0{,}0, 0) \tag{3.4a}$$

$$\hat{z}_i^{DEM} = (0, 0, t_{z,i}, 0, 0, 0) \tag{3.4b}$$

**GPS**: this sensor only provides the robot position:

$$\hat{z}_i^{GPS} = (T_i, 0_{3\times1}) \tag{3.5}$$

**IMU**: from this measurement the proposed approach actually exploits only the *roll* and *pitch* angles, being the rotation around the $z$ axis provided by the IMU usually affected by not negligible inaccuracies. Therefore, you have $\hat{z}_i^{IMU} = \Phi(\mathbf{X}_i)|_{r_{x,i},r_{y,i}}$;

**AMM**: such a constraint is formulated by a composition of two transformation matrices. The first one encodes a roto-translation of the robot around the so-called

*Instantaneous Center of Rotation* (ICR). The proposed approach follows the same formulation presented in [129]:

$$\mathbf{X}(\rho, dr_z) = \begin{bmatrix} cos(\frac{dr_z}{2}) & -sin(\frac{dr_z}{2}) & 0 & \rho \cdot cos(\frac{dr_z}{2}) \\ sin(\frac{dr_z}{2}) & cos(\frac{dr_z}{2}) & 0 & \rho \cdot sin(\frac{dr_z}{2}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

where $\rho$ is the norm of the translation along $dt_x$ and $dt_y$. Additionally, a further rotation along those two axes is added by also taking into account the ground slope, and by rotating the ideal plane on which the vehicle steers following the Ackermann motion model:

$$\mathbf{X}(dr_x, dr_y) = \begin{bmatrix} R(dr_x, dr_y) & 0_{1x3} \\ 0_{3x1} & 1 \end{bmatrix} \tag{3.7}$$

Hence, $\hat{z}^{AMM}$ is obtained as $\Phi(\mathbf{X}(dr_x, dr_y) \cdot \mathbf{X}(\rho, dr_z))$.

### Dynamic Weight Assignment

The impact of each constraint in the final cost function (Eq. 3.2) is weighted by its relative information matrix. As a consequence, such information matrices play a crucial role in weighting the measurements, i.e. giving much reliability to a noisy sensor can lead to errors in the optimization phase. This problem is tackled by dynamically assigning the information matrix for each component as follows:

**WO**: the proposed approach uses as information matrix $\Omega_{i,j}^{WO}$ the inverse of the covariance matrix $\Sigma^{WO}$ of the robot odometry, scaled by the magnitude of the distance and rotation traveled between the nodes $x_i$ and $x_j$, as explained in [154];

**VO**: in this case, the inverse of the covariance matrix $\Sigma^{VO}$ provided as output by the visual odometry front-end is used, weighting the rotational and translational sub-matrices ($\Sigma^{VO,R}$ and $\Sigma^{VO,T}$) with two scalars $\lambda_{VO,R}$ and $\lambda_{VO,T}$, experimentally tuned. Since the VO system internal parameters are not directly tuned, the proposed method employs these ”VO agnostic“ scaling factors that have the analogous effects as injecting a higher sensor noise. In the experiments, $\lambda_{VO,R}$ and $\lambda_{VO,T}$ are set as 5 and 1, respectively;

**MRF**: the information matrix $\Omega_{i,j}^{MRF}$ is set as $diag(0, 0, w_z^{MRF}, 0, 0, 0)$. The weight $w_z^{MRF} = \lambda_{MRF}/|x_i - x_j|_{t_x,t_y}$ is inversely proportional to the distance in the $(x, y)$ plane between the two nodes, while $\lambda_{MRF}$ has been experimentally tuned. $\lambda_{MRF} = 0.8$ in the experiments;

**GPS**: as information matrix $\Omega_i^{GPS}$, the proposed approach uses the inverse of the covariance matrix $\Sigma^{GPS}$ provided by the GPS sensor;

**AMM**: as information matrix $\Omega_{i,j}^{AMM}$, the proposed approach uses an identity matrix scaled by the magnitude of the traveled distance between the nodes $x_i$ and $x_j$, similarly to the wheel odometry constraint. This allows to model the reliability of such a constraint as inversely proportional to the traveled distance;

**IMU**: as information matrix $\Omega_i^{IMU}$, the proposed approach uses the inverse of the covariance matrix $\Sigma^{IMU}$ provided by the IMU sensor;

**DEM**: the information matrix $\Omega_i^{DEM}$ is set as $diag(0, 0, w_z^{DEM}, 0, 0, 0)$, where $w_z^{DEM}$ is empirically tuned. In the experiments $w_z^{DEM}$ is set as 5;

**LID**: the information matrix $\Omega_{i,j}^{LID}$ is set as the inverse of the covariance matrix estimated from the transformation provided by the registration algorithm (e.g., an ICP algorithm), by using the procedure described in [128]. Such an information matrix allows adapting the influence of the point-cloud alignment inside the optimization process, enabling to correctly deal also with the lack of geometrical structure on some dimensions, e.g. in farming scenarios with small plants.

### Sliding-Window Optimization

A re-optimization of the whole pose graph presented above, every time a new node is added, cannot guarantee the real-time performances required for on-line field operations, especially when the graph contains a large number of nodes and constraints. This issue is solved by employing a sliding-window approach, namely performing the optimization procedure only on a sub-graph that includes a sequence of recent nodes. Each time a new node associated with the most recent sensor readings is added to the graph, the sub-graph is updated by adding the new node and removing its oldest one, in a Sliding Window (SW) fashion. The optimization process is performed only on the current sub-graph, while older nodes maintain the state assigned during the last optimization where they were involved. In order to preserve the MRF constraints, the size of the sub-graph is automatically computed so that any adjacent nodes in the previous row are included (see Fig. 3.2, right). Global optimization of the whole pose graph is then carried out off-line, using as initial guess the node states assigned on-line using the SW approach.

### Implementation Details

**Temporal Synchronization**: In the previous sections, I tacitly assumed that all sensor measurements associated with a graph node share the same timestamp. However, in a real context, this is usually not true. In the proposed architecture, the creation of new nodes is triggered every $step_{WO}$ meters (0.3 $m$ in our experiments), by using the wheel odometry as a distance reference. For each node, the proposed architecture associates synchronized estimates of the other sensor readings, which are obtained by means of linear interpolation over the closest readings of each used sensor. This enables to associate to the same node a set of heterogeneous sensor readings that share the same time stamp.

**Visual Odometry Failures**: VO systems are usually tuned by default to provide high accuracy at the expense of the robustness. This limitation is addressed by employing a simple strategy designed to mitigate VO failures. To do so, the local reliability of the WO is exploited: when the difference between WO and VO is greater than a given threshold, a failure of the latter is assumed. In this case, the influence of the VO during the pose graph optimization is reduced by downscaling its information matrix.

**Point-Cloud Registration**: Point-clouds acquired by a 3D LIDAR are typically too sparse to perform a robust alignment: thus, a number of LIDAR readings are accumulated into a single point-cloud by using the motion estimations provided by the VO. The point-cloud registration is finally performed using the Iterative Closest

Point (ICP) algorithm.
**Graph Optimization**: Both the on-line and off-line pose graph optimizations are performed (Sec. 3.2.2) using the Levenberg-Marquardt algorithm implemented in the *g2o* graph optimization framework [88].

### 3.2.3. Experiments

In order to analyze the performance of our system, two datasets[8] with different UGV steering modalities are considered. In *Dataset A* the robot follows 6 adjacent crop rows by constantly maintaining the same global orientation, e.g. half rows have been traversed by moving the robot backward, while in *Dataset B* the robot is constantly moving in the forward direction. Both datasets include data from the same set of sensors: (i) wheel odometry; (ii) VI-Sensor device (stereo camera + IMU) [111]; (iii) Velodyne VLP-16 3D LIDAR; (iv) a low cost U-blox RTK-GPS; (v) an U-blox Precise Point Positioning (PPP) GPS. For a comprehensive description of the UGV farm robot, the sensors setup and the calibration procedure, I refer the interested readers to the on-line supplementary material[9].
In all our experiments, Stereo DSO [161] is employed as the VO subsystem and the ICP implementation provided by the PCL library as point-cloud registration front-end. The IMU, the wheel odometry and both the GPSs provide internally filtered outputs (attitude, relative and absolute positions, respectively), along with covariance matrices associated with the outputs in the IMU and GPSs cases. The DEM of the inspected field is built by using the Google Elevation API that provides, for the target field, measurements over a regularly spaced grid with a resolution of 10 meters. Such measurements are then interpolated to provide denser information. The 3D ground truth reference has been acquired by using a LEICA laser tracker. This sensor tracks a specific target mounted on the top of the robot and provides a position estimation ($x$, $y$ and $z$) with millimeter-level accuracy. Both datasets have been acquired by using the Bosch BoniRob farm robot (Fig. 3.1, left) on a field in Eschikon, Switzerland (Fig. 3.1, right). The Bosch BoniRob is a multi-functional, four-wheel drive agricultural robot. Each wheel has 3 degrees of freedom, that means they can independently turn, steer and change track width. The BoniRob is hence an omnidirectional drive robot that can adapt itself to a wide range of farming fields. For this specific work, the robot has been manually controlled according to Ackerman steering kinematics. In addition to these two datasets, a third dataset (*Dataset C*) have been created. This datasets simulates a sudden RTK-GPS signal loss, e.g. due to a communication loss between the base and the rover stations. In particular, it simulates the accuracy losses by randomly switching for some time to the PPP-GPS readings.

In the following, the quantitative results are reported by using the following statistics build upon the localization errors with respect to the ground truth reference: Root Mean Square Error ($RMSE$ in the tables), maximum and mean absolute error ($Max$ and $Mean$), and mean absolute error along each component ($err_x$, $err_y$ and $err_z$).

[8] www.dis.uniroma1.it/~labrococo/fsd
[9] www.dis.uniroma1.it/~labrococo/fsd/ral2018sup.pdf

### 3.2.4. Dataset A and Dataset B

**Table 3.1.** Error statistics in *Dataset A* and *Dataset B* by using different sensor setups and constraints for the global, off-line and the sliding-window (SW), on-line pose graph optimization procedures. The results of the ORB SLAM 2 system (OS2 in the table) are reported for both type of GPSs.

| GPS | WO | VO | IMU | AMM | ELEV | LIDAR | MRF | SW | **DatasetA** | | | | | **DatasetB** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | $err_x$ | $err_y$ | $err_z$ | $Max$ | $RMSE$ | $err_x$ | $err_y$ | $err_z$ | $Max$ | $RMSE$ |
| PPP | | | | | | | | | 0.349 | 0.582 | 1.577 | 2.959 | 1.710 | 0.306 | 0.501 | 1.484 | 2.875 | 1.621 |
| | ✓ | | | | | | | | 0.311 | 0.520 | 1.537 | 2.954 | 1.630 | 0.246 | 0.416 | 1.424 | 2.829 | 1.504 |
| | ✓ | | | | ✓ | | | | 0.343 | 0.572 | 0.475 | 1.627 | 1.071 | 0.241 | 0.408 | 0.492 | 1.782 | 1.168 |
| | ✓ | ✓ | | | | | | | 0.239 | 0.412 | 0.672 | 1.628 | 0.961 | 0.222 | 0.362 | 1.298 | 2.392 | 1.211 |
| | ✓ | ✓ | | | | ✓ | | | 0.233 | 0.422 | 0.649 | 1.421 | 0.863 | 0.227 | 0.361 | 1.292 | 2.571 | 1.242 |
| | ✓ | ✓ | | | | | ✓ | | 0.239 | 0.411 | 0.528 | 1.398 | 0.719 | 0.221 | 0.364 | 1.019 | 2.362 | 1.119 |
| | ✓ | ✓ | | ✓ | | ✓ | | | 0.224 | 0.411 | 0.551 | 1.375 | 0.726 | 0.201 | 0.397 | 0.881 | 2.019 | 0.951 |
| | ✓ | ✓ | ✓ | | | ✓ | | | 0.222 | 0.389 | 0.531 | 1.281 | 0.729 | 0.229 | 0.407 | 0.652 | 1.613 | 0.829 |
| | ✓ | ✓ | ✓ | ✓ | | | | | 0.239 | 0.361 | 0.523 | 1.272 | 0.739 | 0.231 | 0.369 | 0.641 | 1.461 | 0.732 |
| | ✓ | ✓ | ✓ | ✓ | | ✓ | | | 0.224 | 0.371 | 0.453 | 1.124 | 0.621 | 0.221 | 0.362 | 0.619 | 1.611 | 0.734 |
| | ✓ | ✓ | ✓ | ✓ | | | ✓ | | 0.234 | 0.360 | 0.440 | 1.093 | 0.564 | 0.199 | 0.360 | 0.475 | 1.161 | 0.660 |
| | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | 0.234 | 0.342 | 0.311 | 0.921 | 0.422 | 0.198 | 0.361 | 0.463 | 1.121 | 0.604 |
| | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | | 0.211 | **0.331** | **0.282** | 0.897 | 0.416 | 0.182 | 0.339 | 0.369 | 1.198 | 0.471 |
| | **✓** | **✓** | **✓** | **✓** | **✓** | **✓** | **✓** | | **0.201** | **0.331** | 0.289 | **0.824** | **0.401** | **0.173** | **0.331** | **0.321** | **1.117** | **0.461** |
| | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 0.252 | 0.419 | 0.349 | 0.991 | 0.549 | 0.291 | 0.431 | 0.459 | 1.291 | 0.652 |
| OS2+GPS | | | | | | | | | 0.234 | 0.417 | 0.643 | 1.534 | 0.915 | 0.209 | 0.401 | 0.371 | 2.123 | 1.047 |
| RTK | | | | | | | | | 0.059 | 0.051 | 0.121 | 0.431 | 0.128 | 0.054 | 0.062 | 0.091 | 0.322 | 0.122 |
| | ✓ | | | | | | | | 0.053 | **0.042** | 0.105 | 0.431 | 0.125 | 0.049 | 0.058 | 0.086 | 0.321 | 0.119 |
| | ✓ | ✓ | | | | | | | 0.053 | **0.042** | 0.054 | 0.279 | 0.088 | 0.047 | 0.048 | 0.062 | 0.192 | 0.091 |
| | ✓ | ✓ | ✓ | | | | | | 0.048 | 0.049 | 0.060 | 0.306 | 0.092 | 0.045 | 0.046 | 0.064 | 0.209 | 0.091 |
| | ✓ | ✓ | | ✓ | | | | | 0.046 | 0.047 | 0.061 | 0.279 | 0.090 | 0.045 | **0.045** | 0.064 | 0.211 | 0.090 |
| | ✓ | ✓ | | ✓ | | ✓ | | | 0.046 | 0.047 | 0.061 | 0.278 | 0.089 | 0.045 | **0.045** | 0.062 | 0.197 | 0.090 |
| | ✓ | ✓ | ✓ | ✓ | | | | | 0.046 | 0.050 | 0.056 | **0.248** | 0.088 | 0.045 | 0.046 | 0.039 | 0.165 | 0.075 |
| | ✓ | ✓ | ✓ | | | ✓ | ✓ | | 0.047 | 0.049 | **0.034** | 0.251 | 0.076 | 0.045 | 0.046 | 0.035 | 0.154 | 0.074 |
| | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | 0.051 | 0.049 | 0.068 | 0.312 | 0.097 | 0.046 | 0.048 | 0.064 | 0.219 | 0.095 |
| | **✓** | **✓** | **✓** | **✓** | | **✓** | **✓** | | **0.045** | 0.048 | **0.034** | 0.260 | **0.075** | **0.044** | 0.046 | **0.034** | **0.151** | **0.073** |
| | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | 0.053 | 0.051 | 0.042 | 0.272 | 0.084 | 0.051 | 0.051 | 0.035 | 0.172 | 0.084 |
| OS2+GPS | | | | | | | | | 0.051 | 0.045 | 0.059 | 0.293 | 0.097 | 0.051 | 0.054 | 0.068 | 0.231 | 0.102 |

This set of experiments shows the effectiveness of the proposed method and the benefits introduced by each cue. Tab. 3.1 reports the results obtained by using different sensor combinations and optimization procedures over *Dataset A* and *Dataset B*. The table is split according to the type of GPS sensor used; the sensor setups that bring the overall best results are highlighted in bold. The proposed system is also compared with the ORB SLAM 2 system [107], a best-in-class Visual SLAM system, with its mapping and loop closures back-ends activated. For a fair comparison, the GPS information (PPP and RTK) is added as a global constraint at each key-frame triggered by ORB SLAM 2.

A first result is the positive impact of including the new proposed constraints in the optimization: both the ELEV and MRF cues individually integrated lead to noteworthy improvements in the estimation along the $z$ when a noisy GPS is used (PPP-GPS case). Another remarkable result is the decreasing error trend, almost monotonic: the more sensors are introduced in the optimization process, the smaller the resulting $RMSE$ and $Max$ errors are. This behavior occurs in both *Dataset*

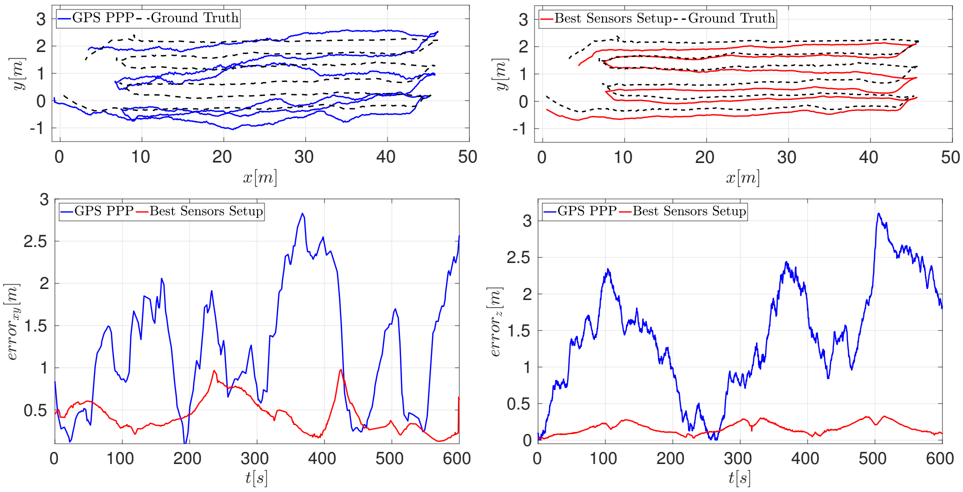


**Figure 3.4.** Dataset A, PPP-GPS: (Top) Qualitative top view comparison between the raw GPS trajectory (left) and the optimized trajectory (right); (Bottom): absolute $x, y$ (left) and $z$ (right) error plots for the same trajectories.

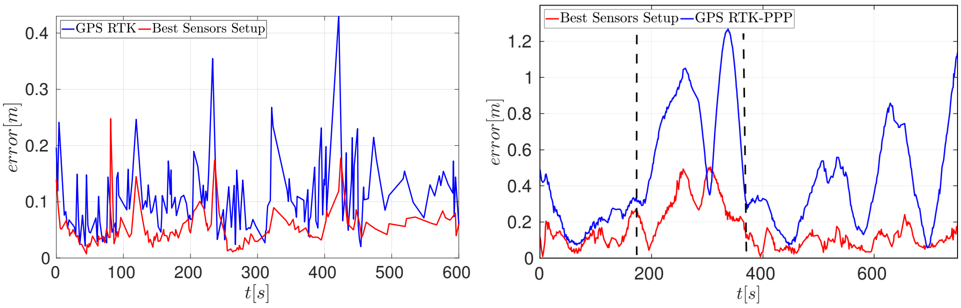


**Figure 3.5.** (Left) Dataset A, RTK-GPS: Absolute error plots for the raw GPS trajectory and the optimized trajectory obtained by using the best sensors configuration (see Tab. 3.1). (Right) Dataset C, absolute error plots for the raw GPS trajectory and the optimized trajectory (see Tab. 3.3). The time interval when the signal loss happens is bounded by the two dashed lines.

*A* and *Dataset B*, and proves how the proposed method properly handles all the available sources of information. Another important outcome is the relative *RMSE* improvement obtained between the worst and the best set of cues, which is around the 37% for RTK case, and 76% for the PPP case; in both these setups our system outperforms the ORB SLAM 2 system. A noteworthy decrease of the error also happens to the *Max* error statistic, respectively, 40% and 70%: this fact brings a considerable benefit to agricultural applications, where spikes in the location error might lead to harming crops. For the best performing sensor setup, this experimental section also reports the results obtained by using the SW, on-line pose graph optimization procedure (Sec. 3.2.2): also, in this case, the relative improvement is remarkable (32% and 67%, respectively), enabling a safer and more accurate real-time UGV navigation.

Fig. 3.4 (top) depicts a qualitative top view comparison between the raw PPP-GPS trajectory (top-left) and the trajectory (top-right) obtained after the pose graph

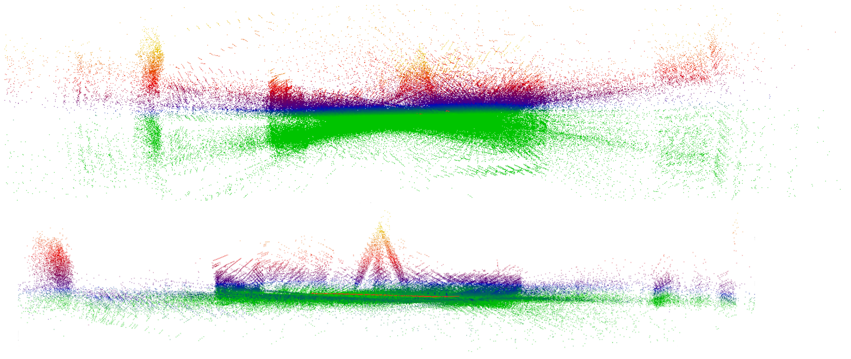


**Figure 3.6.** Comparison between output point-clouds: (left) without IMU and LIDAR and (right) with IMU and LIDAR in the optimization.

optimization, using the best sensors configuration in *Dataset A*. The error plots (bottom) show how the introduction of additional sensors and constraints allows to significantly improve the pose estimation. Similar results for *Dataset A* and RTK-GPS are reported in Fig. 3.5 (left).
For both GPSs, the maximal error reduction happens when all the available cues are used within the optimization procedure except for the low-cost RTK-GPS case, where the ELEV constraint worsens the error along the $z$ axis. Actually, the RTK-GPS usually provides an altitude estimate, which is more accurate than the one provided by the interpolated DEM. It is also noteworthy to highlight the propagation of the improvements among state dimensions: the integration of constraints that only act on a part of the state (e.g., IMU, LIDAR, ELEV) also positively affects the remaining state components.
As a further qualitative evaluation, Fig. 3.6 reports the global point-cloud obtained by rendering LIDAR scans at each estimated position, with and without the IMU and LIDAR contributions within the optimization procedure: the attitude estimation greatly benefits from these contributions. The runtimes of our system are reported in Tab. 3.2, for both the off-line and on-line, sliding-window cases.

**Table 3.2.** Runtime performance for the global, off-line and the sliding-window (SW), on-line pose graph optimization (Core-i7 2.7 GHz laptop).

| | SW | *#Nodes* | *#Edges* | *#Iters* | *time(s)* |
|---|---|---|---|---|---|
| *Dataset* | | 786 | 8259 | 24 | 13.493 |
| *A* | ✓ | 98 | 763 | 4 | 0.0989 |
| *Dataset* | | 754 | 8032 | 22 | 12.993 |
| *B* | ✓ | 104 | 851 | 5 | 0.1131 |

### 3.2.5. Dataset C

This set of experiments is designed to prove the robustness of the proposed system against sudden losses in the GPS sensor accuracy. Tab. 3.3 reports the quantitative results of our system over *Dataset C* by means of *RMSE* and *Max* errors. Even

**Table 3.3.** Error statistics in *Dataset C* by using different sensor setups and constraints in the optimization procedure.

| | | | | | | | | **DatasetC** | |
|---|---|---|---|---|---|---|---|---|---|
| GPS | WO | VO | IMU | AMM | ELEV | LIDAR | MRF | *Max* | *RMSE* |
| RTK+PPP | | | | | | | | 1.313 | 0.647 |
| | ✓ | | | | | | | 1.291 | 0.613 |
| | ✓ | ✓ | | | | | | 1.259 | 0.552 |
| | ✓ | ✓ | | ✓ | | ✓ | | 1.171 | 0.431 |
| | ✓ | ✓ | ✓ | | | ✓ | | 0.882 | 0.356 |
| | ✓ | ✓ | | ✓ | ✓ | ✓ | | 0.551 | 0.223 |
| | ✓ | ✓ | ✓ | | | | ✓ | 0.655 | 0.204 |
| | ✓ | ✓ | ✓ | ✓ | | | ✓ | 0.521 | 0.201 |
| | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 0.534 | 0.181 |
| | **✓** | **✓** | **✓** | **✓** | | **✓** | **✓** | **0.419** | **0.168** |

in the presence of a RTK-GPS signal loss that lasts for more than one crop row, the best sensors setup leads to a remarkable *RMSE* of 0.166 m and a relative improvement around the 72%. Moreover, also in *Dataset C* the *RMSE* and the *Max* error statistics follow the same decreasing trend shown in Tab. 3.1. Fig. 3.5 (right) compares the absolute error trajectories for the best sensors configuration against the error trajectory obtained by using only the GPS measurements: the part where the signal loss occurs is affected by a higher error. Another interesting observation regards the non-constant effects related to the use of the ELEV constraint. As shown in Tab. 3.3, in some cases it allows decreasing the overall error, while in other cases it worsens the estimate. The latter happens when the pose estimation is reliable enough, i.e. when most of the available constraints are already in use. As explained in section 3.2.4, in such cases the ELEV constraint does not provide any additional information to the optimization procedure, while with a less accurate PPP-GPS its use is certainly desirable. The last remark concerns the best *RMSE*, which is around $7.5cm$, achievable when the best sensor setup is available. A farming robot navigating within the environment with an error similar to the one achieved with the proposed architecture can safely move without the risk of damaging the crop. Indeed, planting patterns exploited in agriculture applications usually consider a minimum intra-row space around $25 - 30cm$.

## 3.3. Summary of Results and Lessons Learned

Conversely from the related state-of-the-art, where the proposed solutions exploit the GPS as the main source of information, and from the SLAM literature where, similarly, only a few sensors are used to perceive and map the environment, I propose a multi-cue localization a mapping system. Indeed, when navigating within complex environments, as in a flat and landmark-less farm field, each individual sensor, if used as a stand-alone system, would probably lead the navigation system to fail. On the other hand, the exploitation of multiple sources of information and context-based constraints is fundamental to aid the robot localization and mapping capabilities. Particularly, as described in Sec. 3.2.1, the problem is formulated as a pose-graph optimization, where each new heterogeneous measurement acts as an additional constraint on a specific part of the robot state. Such a formulation of the problem also allows taking advantage of the specificity of the scenario by introducing context-based constraints which aim to enhance the strengths of each integrated information. Through the analysis of the localization accuracy, this chapter shows how the whole system ensures a sufficient precision that enables for autonomous and safe navigation, *i.e.* guaranteeing the robot to not harm the crop.

# Chapter 4

# Collaborative Mapping

The module described in the previous chapter enables a farming robot to navigate in a previously unknown farming environment by building an accurate map and, at the same time, localizing within it. However, an additional capability for a farming robot would be to build a map of the same farmland in a cooperative manner. This capability proves to be essential and offer several benefits in automatizing farming activities. For instance, in the case of a vast environment, the constructed maps, if partially overlapping, can be merged together. This would allow using more robots in a collaborative manner [81], reducing the working time. Moreover, registering maps of the same farmland acquired in different days or hours allows observing and measuring the evolution of the crops and the appearance of infested areas or diseases.

A collaborative module would also provide benefits when using heterogeneous robots. In the specific use-case scenario considered in this thesis, the Flourish project, a collaborative mapping module would allow the UAV to quickly provide a coarse reconstruction of a large area [82], which can be updated with a more detailed and higher resolution map portions generated by the UGV when inspecting some specific areas.

This chapter focuses on the collaborative mapping between heterogeneous robots, such as a UAV and a UGV. There are two classes of methods designed to generate multi-robot environment representations: (i) multi-robot Simultaneous Localization and Mapping (SLAM) algorithms (e.g., [72][58]), that concurrently build a single map by fusing raw measurements or small local maps generated from multiple robots; (ii) map registration algorithms (e.g., [15][21]) that align and merge maps independently generated by each robot into a unified map. On the one hand, the lack of distinctive visual and 3D landmarks in an agricultural field, along with the difference in the robots' point-of-views (e.g., Fig. 4.1), prevent direct employment of standard multi-robot SLAM pipelines, either based on visual or geometric features. On the other hand, merging maps independently generated by the UAVs and UGVs in an agricultural environment is also a complex task, since maps are usually composed of similar, repetitive patterns that easily confuse conventional data association methods [57]. Furthermore, due to inaccuracies in the map building process, the merged maps are usually affected by local inconsistencies, missing data, occlusions, and global deformations such as directional scale errors, that negatively affect the performance of standard alignment methods. Geolocation information associated with (i) sensor readings or (ii) maps often can not solve the limitations of conventional methods in agricultural environments, since the location and orientation accuracy provided by standard reference sensors[1] [74] is not suitable to prevent such systems from converging towards sub-optimal solutions (see Sec. 4.2.4)

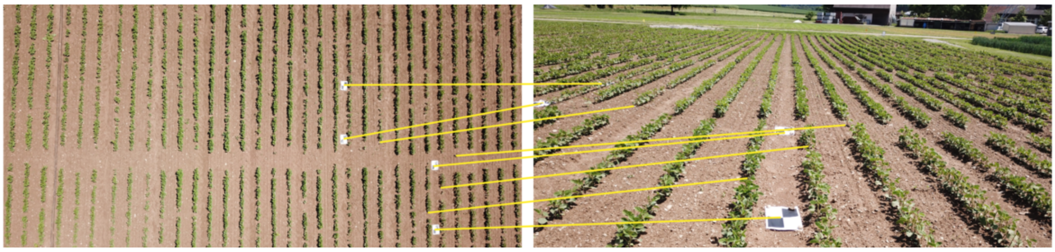


**Figure 4.1.** Pictures of the same portion of field seen from the UAV point-of-view (left) and from the UGV point-of-view (right). The local crop arrangement geometry, such as the missing crop plants, is generally not visible from the UGV point-of-view. The yellow solid lines represent an example of manually annotated correct point matches. It is important to underline the complexity required in obtaining correct data association, also from a human point-of-view. The fiducial markers on the filed have been used to compute the ground truth alignments between the maps.

In this chapter, I present one of the major contributions of this thesis which addresses the collaborative mapping between aerial and ground robots in farming scenarios.

## 4.1. Related Work

The field of multi-robot cooperative mapping is a recurrent and relevant problem in literature and, as previously introduced, several solutions have been presented by means of either multi-robot SLAM algorithms or map merging/map registration strategies, in both 2D ([15, 19, 139]) and 3D ([21][51][75]) settings. Registration of point cloud based maps can also be considered as an instance of the more general point set registration problem [30][47]. In this work, I mainly review methods based on map registration, since the heterogeneity of the involved robots and the lack of distinctive visual and geometrical features on an agricultural environment prevent the employment of standard multi-robot SLAM methods; a comprehensive literature review about this class of methods can be found in [140].

Map registration is a challenging problem especially when dealing with heterogeneous robots, where data is gathered from different points-of-view and with different noise characteristics. It has been intensively investigated, especially in the context of urban reconstruction with aerial and ground data. In [145], the authors focus on the problem of geo-registering ground-based multi-view stereo models by proposing a novel viewpoint-dependent matching method. Wang *et al.* [160] deal with aligning 3D structure-from-motion point clouds obtained from Internet imagery with existing geographic information sources, such as noisy geotags from input Flickr photos and

[1] Global Positioning Systems (GPSs) and Attitude and Heading Reference Systems (AHRSs)

geotagged city models and images collected from Google Street View and Google Earth. Bódis-Szomorú *et al.* [25] propose to merge low detailed airborne point clouds with incomplete street-side point clouds by applying volumetric fusion based on a 3D tetrahedralization (3DT). Früh *et al.* [52] propose to use Digital Surface Models obtained from a laser airborne reconstruction to localize a ground vehicle equipped with 2D laser scanners and a digital camera, detailed ground-based facade models are hence merged with a complementary airborne model. Michael *et al.* [152] propose a collaborative UAV-UGV mapping approach in earthquake-damaged contexts. They merge the point clouds generated by the two robots using a 3D Iterative Closest Point (ICP) algorithm, with an initial guess provided by the (known) UAV takeoff location; the authors make the assumption that the environment is generally described by flat planes and vertical walls, also called the "Manhattan world" assumption. The ICP algorithm has also been exploited in [49] and [68]. Forster *et al.* [49] align dense 3D maps obtained by a UGV equipped with an RGB-D camera and by a UAV running dense monocular reconstruction: they obtain the initial guess alignment between the maps by localizing the UAV with respect to the UGV with a Monte Carlo Localization method applied to height-maps computed by the two robots. Hinzmann *et al.* [68] deal with the registration of dense LiDAR-based point clouds with sparse image-based point clouds by proposing a probabilistic data association approach that specifically takes the individual cloud densities into consideration. In [56], Gawel *et al.* present a registration procedure for matching LiDAR point-cloud maps and sparse vision keypoint maps by using structural descriptors.

Although much literature addresses the problem of map registration for heterogeneous robots, most of the proposed methods make strong context-based assumptions, such as the presence of structural or visual landmarks, "Manhattan world" assumptions, etc. Registering 3D maps in an agricultural setting, in some respects, is even more challenging: the environment is homogeneous, poorly structured and it usually gives rise to strong sensor aliasing. For these reasons, most of the approaches mentioned above cannot directly be applied to an agricultural scenario. Localization and mapping in an agricultural scenario is a topic that is recently gathering great attention in the robotics community [162][40][74]. Most of these systems, however, deal with a single robot, and the problem of fusing maps built from multiple robots is usually not adequately addressed and a little, very recent research exists on this topic. Dong *et al.* [37] propose a spatio-temporal reconstruction framework for precision agriculture that aims to merge multiple 3D field reconstructions of the same field across time. They use single row reconstructions as starting points for the data association, that is actually performed by using standard visual features. This method uses images acquired by a single UGV that moves in the same field at different times and, being based on visual features, cannot manage drastic viewpoint changes or large misalignments when matching aerial and ground maps. A local feature descriptor designed to deal with large viewpoint changes has been proposed by Chebrolu *et al.* in [28]. The authors propose to encode with such descriptor the almost static geometry of the crop arrangement in the field. Despite the promising results, this method suffers from the presence of occluded areas when switching from the UAV to the UGV point-of-view.

## 4.2. AgriColMap: Aerial-Ground Collaborative 3D Mapping for Precision Farming

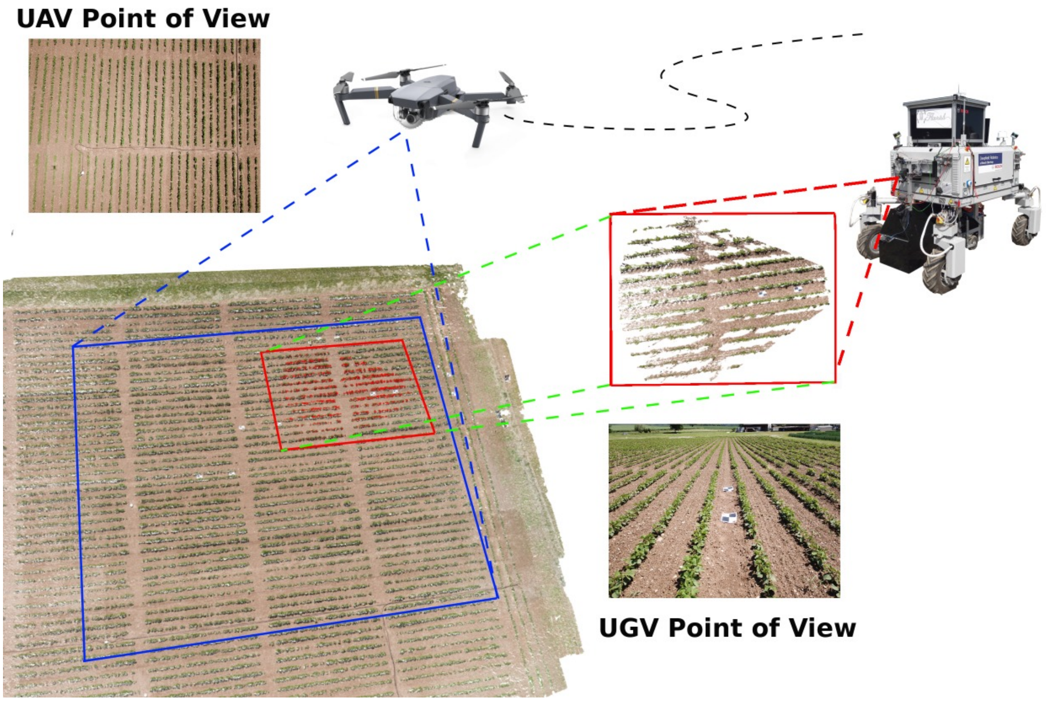


**Figure 4.2.** An overview of AgriColMap. Both the UGV and UAV generate, using data gathered from their onboard cameras, colored point clouds of the cultivated field. The proposed method aims to accurately merge these maps by means of an affine transformation that registers the UGV submap (red rectangular area) into the UAV aerial map (blue rectangular area), taking into account possible scale discrepancies.

In this section, I introduce AgriColMap, an Aerial-Ground Collaborative 3D Mapping pipeline, which provides an effective and robust solution to the cooperative mapping problem with heterogeneous robots, specifically designed for farming scenarios. This problem is addressed by proposing a non-rigid map registration strategy able to deal with maps with different resolutions, local inconsistencies, global deformations, and relatively large initial misalignments. It is assumed that both a UAV and a UGV can generate a colored, geotagged point cloud of a target farm environment, e.g., by means of photogrammetry-based 3D reconstruction. (Fig. 4.2). To solve the data association problem between the input point clouds, The proposed system switches from a 3D problem to a 2D one, solved by using a global, 2D dense matching approach. The key intuition behind this choice is that points belonging to a cloud locally share similar displacement vectors that associate such points with points in the other cloud. Therefore, the idea is to employ a regularized 2D matching strategy that penalizes the displacement vectors discontinuities for each point neighborhood[2]. With this formulation, good correspondences are iteratively improved and

[2] In other words, a regularized matching enforces the smoothness of the displacement vectors for neighboring points.

spread through cooperative search among neighboring points. This approach has been inspired by the Large displacement Dense Optical Flow (LDOF) problem in computer vision and, actually, the data association problem is formulated as a LDOF problem. To this end, the colored point clouds are converted into a more suited, multimodal environment representation that allows one to exploit two-dimensional approaches and to highlight both the semantic and the geometric properties of the target map. The former is represented by a vegetation index map, while the latter through a Digital Surface Model (DSM). More specifically, each input point cloud is converted into a grid representation, where each cell stores (i) the Excess Green index (ExG) and (ii) the local surface height information (e.g., the height of the plants, soil, etc.). Then, the data provided by the GPS and the AHRS are used to extract an initial guess of the relative displacement and rotation between grid maps to match. Hence, a dense set of point-to-point correspondences are computed between matched maps by exploiting a modified, state-of-the-art LDOF system [73], tailored to the precision agriculture context. To adapt this algorithm to the considered environment representation, the proposed architecture uses a different cost function that involves both the ExG information and the local structure geometry around each cell. The proposed approach selects, by means of a voting scheme, the bigger subset of correspondences with coherent, similar flows, to be used to infer a preliminary alignment transformation between the maps. In order to deal with directional scale errors, a non-rigid point-set registration algorithm is used to estimate an affine transformation. The final registration is obtained by performing a robust point-to-point registration over the input point clouds, pruned from all points that do not belong to the vegetation. A schematic overview of the proposed approach is depicted in Fig. 4.3.

An exhaustive set of experiments (Sec. 4.2.4) on data acquired by a UAV and a handheld camera are reported, simulating the UGV, on crop fields in Eschikon, Switzerland. The results show that the proposed approach is able to guarantee with a high probability a correct registration for an initial translational error up to 5 meters, an initial heading misalignment up to 11.5 degrees, and a directional scale error of up to 30%. The proposed method achieves similar registration performance across fields with three different crop species, showing that it generalizes well across different kinds of crop species. Finally, is also reported an additional comparison with state-of-the-art point-to-point registration and matching algorithms, showing that the proposed approach outperforms them in all the experiments.

### 4.2.1. Problem Statement and Assumptions

Given two 3D colored point clouds $\mathcal{M}_A$ and $\mathcal{M}_G$ of a farmland (Fig. 4.3, first column), built from data gathered from a UAV and a UGV, respectively, the goal is to find a transformation $F : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ that allows to accurately align them. $\mathcal{M}_A$ and $\mathcal{M}_G$ can be generated, for instance, by using an off-the-shelf photogrammetry-based 3D reconstruction software applied to sequences of geotagged images. The proposed method makes the following assumptions:

1. The input maps built form UAVs and UGVs data can have different spatial resolutions but they refer to the same field, with some overlap among them;

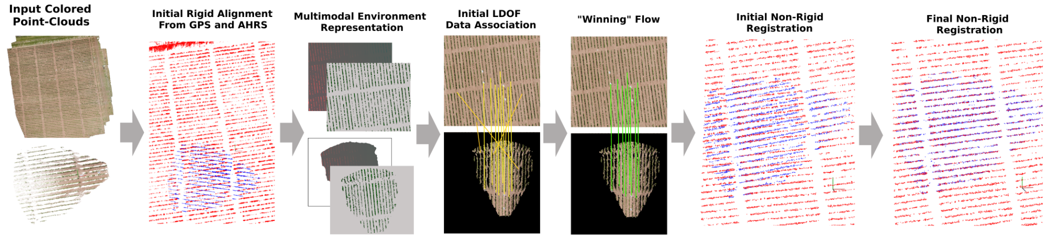

Input Colored Point-Clouds
Initial Rigid Alignment From GPS and AHRS
Multimodal Environment Representation
Initial LDOF Data Association
"Winning" Flow
Initial Non-Rigid Registration
Final Non-Rigid Registration

**Figure 4.3.** Overview of the proposed approach. For visualization purposes, in columns 2,6 and 7 the UGV and UAV point clouds are colored in blue and red, respectively, pruned from all points that do not belong to vegetation, according to a thresholding operator applied to the ExG index. Starting from the left side: (i) the input colored point clouds gathered by the UAV and UGV; (ii) the initial noisy and biased rigid alignment provided by the GPS and the AHRS; (iii) the generated multimodal grid maps; (iv) the initial LDOF data associations, i.e. the point-to-point correspondences, in yellow; (v) the ”winning“ data associations (flows), in green, selected by a voting scheme; (vi) the aligned point clouds according to the initial affine transform; (vii) the final non-rigid registration after the refinement step.

2. The data used to build the maps were acquired at approximately the same time;
3. The maps are roughly geotagged, possibly with noisy locations and orientations;
4. They can be affected by local inconsistencies, missing data, and deformations, such as directional scale errors.
5. $\mathcal{M}_A$ is not affected by any scale inconsistencies.

Hypotheses 1, 2, and 3 are essential data requirements. Hypothesis 4) implies the violation of the typical rigid-body transformation assumption between the two maps: therefore, $F$ describes an affine transformation that allows anisotropic (i.e., non-uniform) scaling between the maps. Hypothesis 5) is an acceptable assumption, since the map created by the UAV is usually wider than $\mathcal{M}_G$, and generated by using less noisy GPS readings, so the scale drift effect tends to be canceled: hence, the proposed approach looks for a transformation that aligns $\mathcal{M}_G$ with $\mathcal{M}_A$ by correcting the scale errors of $\mathcal{M}_G$ with respect to $\mathcal{M}_A$.

### 4.2.2. Data Association

In order to estimate the transformation $F$ that aligns the two maps, the proposed approach seeks a set of point correspondences, $m_{A,G} = \{(p, q) : p \in \mathcal{M}_A, q \in \mathcal{M}_G\}$ between $\mathcal{M}_A$ and $\mathcal{M}_G$, that represent points pairs belonging to the same global 3D position. As introduced before and shown in the experiments (see Sec. 4.2.4), conventional sparse matching approaches based on local descriptors are unlikely to provide effective results due to the big amount of repetitive and non-distinctive patterns spread over farmlands. Instead, inspired by the fact that when the maps are misaligned, points in $\mathcal{M}_A$ locally share a coherent ”flow“ towards corresponding points in $\mathcal{M}_G$, the proposed method casts the data association estimation problem as a *dense*, *regularized*, matching approach. This problem resembles the dense optical flow estimation problem for RGB images: in this context, global methods (e.g., [70]) aim to build correspondences pixel by pixel between a pair of images by minimizing a cost function that, for each pixel, involves a data term that measures the point-wise similarity and a regularization term that fosters smoothness between nearby flows (i.e., nearby pixel to pixel associations).

### Multimodal Grid Map

The goal is to estimate $m_{A,G}$ by computing a ”dense flow“ that, given an initial, noisy alignment between the maps provided by a GPS and a AHRS (Fig. 4.3, second column), associates points in $\mathcal{M}_A$ with points in $\mathcal{M}_G$. Unfortunately, conventional methods designed for RGB images are not directly applicable to colored point clouds: I introduce here a multimodal environment representation that allows exploiting such methods while enhancing both the semantic and the geometrical properties of the target map. A cultivated field is basically a globally flat surface populated by plants. A DSM[3] can well approximate the field structure geometry, while a vegetation index can highlight the meaningful parts of the field and the visual relevant patterns: in the considered environment representation, both these intuitions are exploited. The proposed approach generates a DSM from the point cloud; for each cell of the DSM grid, the model also provides an ExG index that, starting from the RGB values, highlights the amount of vegetation. More specifically, a colored point cloud $\mathcal{M}$ is transformed into a two-dimensional grid map $\mathcal{J} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ (Fig. 4.3, third column), where for each cell it is provided both the surface height and the ExG index, with the following procedure:

1. A rectangle that bounds the target area by means of minimum-maximum latitude and longitude is selected;

2. The selected area is discretized into a grid map $\mathcal{J}$ of $w \times h$ cells, by using a step of $s$ meters. In practice, each of the $w \times h$ cells represents a square of $s \times s$ meters. Each cell is initialized with $(0, 0)$ pairs;

3. Remembering that $\mathcal{M}$ is geotagged (see Sec. 4.2.1), each 3D point of $\mathcal{M}$ can be associated to one cell of $\mathcal{J}$ just using the $x, y$, and yaw information;

4. For each cell with associated at least one 3D point, the proposed model computes: (a) the height as a *weighted* average of the $z$ coordinates of the 3D points that belong to such cell; (b) the ExG index as a *weighted* average of the ExG indexes of the 3D points that belong to such cell, where for each point $p$:
$$ExG(p) = 2g_p - r_p - b_p. \tag{4.1}$$
with $r_p$, $g_p$ and $b_p$ the RGB components of the point; (c) the 3D global position of the nearest point is stored in the original colored point cloud.

Both the averages use as weighting factor a circular, bivariate Gaussian distribution with standard deviation $\sigma_{avg}$: points with $x, y$ coordinates close to the center of the cell get a higher weight.

[3] A DSM is a raster representation of the height of the objects on a surface.

### Multimodal Large displacement Dense Optical Flow

The proposed architecture generates from both the $\mathcal{M}_A$ and $\mathcal{M}_G$ the corresponding multimodal representations $\mathcal{J}_A$ and $\mathcal{J}_G$. In the ideal case, with perfect geotags and no map deformations, a simple geotagged superimposition of the two maps should provide a perfect alignment: the ”flow“ that associates cells between the two maps should be zero. Unfortunately, in the real case, due to the inaccuracies of both the geotags and the 3D reconstruction, non zero, potentially large displacements are introduced in the associations. These offsets are locally consistent but not constant for each cell, due to reconstruction errors. To estimate the offsets map, a modified version of the Coarse-to-fine PatchMatch (CPM) framework is employed [73]. CPM is a recent LDOF system that provides cutting edge estimation results even in presence of very large displacements, and is more efficient than other state-of-the-art methods with similar accuracy.
For efficiency, CPM looks for the best correspondences of some seeds that are refined by means of a dense, iterative neighborhood propagation: the seeds are a set of points regularly distributed within the image. Given two images $\mathcal{I}_0, \mathcal{I}_1 \in \mathbb{R}^2$ and a collection of seeds $S = \{s_1, \ldots, s_n\}$ at position $\{p(s_1), \ldots, p(s_n)\}$, the goal of this framework is to determine the flow of each seed $f(s_i) = M(p(s_i)) - p(s_i) \in \mathbb{R}^2$, where $M(p(s_i))$ is the corresponding matching position in $\mathcal{I}_1$ for the seed $s_i$ in $\mathcal{I}_0$. The flow computation for each *seed* is performed by an iterative, coarse-to-fine random search strategy that minimizes a cost function:

$$f(s_i) = \underset{f_{s_j}}{\operatorname{argmin}}(C(f(s_j))), s_j \in s_i \cap \mathcal{N}_i \tag{4.2}$$

where $C(f(\cdot))$ denotes the match cost between the patch centered at $p(s_i)$ in $\mathcal{I}_0$ and the patch centered in $p(s_i) + f(\cdot)$ in $\mathcal{I}_1$, while $\mathcal{N}_i$ is a set of spatially adjacent neighbors seeds around $s_i$ whose flow has already been computed in the current iteration with Eq. 4.2. For a comprehensive description of the flow estimation pipeline, I refer the reader to [73].

The goal is to use the CPM algorithm to compute the flow between $\mathcal{J}_A$ and $\mathcal{J}_G$. To exploit the full information provided by the grid maps (see Sec. 4.2.2), I modified the CPM matching cost in order to take into account both the height and ExG channels. The cost function is split in two terms:

$$C_{flow}(f(s_i)) = \alpha \cdot C_{DY}(f(s_i)) + \beta \cdot C_{FPFH}(f(s_i)) \tag{4.3}$$

$C_{DY}(f(s_i))$ is the DAISY [156] based match cost as in the original CPM algorithm: in this case the DAISY descriptors have been computed from the ExG channel of $\mathcal{J}_A$ and $\mathcal{J}_G$. $C_{FPFH}(f(s_i))$ is a match cost computed using the height channel. The proposed method makes use of the Fast Point Feature Histograms (FPFH) [137] descriptor for this second term: the FPFH descriptors are robust multi-dimensional features which describe the *local* geometry of a point cloud[4], in this case, they are computed from the organized point cloud generated from the height channel of $\mathcal{J}_A$ and $\mathcal{J}_G$. The parameters $\alpha$ and $\beta$ are the weighting factors of the two terms. As in

[4] It is noteworthy to highlight that the FPFH, being a local descriptor, does not embed global displacements along the axes.

[73], the patch-based matching cost is chosen to be the sum of the absolute difference over all the 128 and 32 dimensions of the DAISY and FPFH flows, respectively, at the matching points. The proposed cost function takes into account both the visual appearance and the local 3D structure of the plants.

Once the dense flow between $\mathcal{J}_A$ and $\mathcal{J}_G$ has been computed (Fig. 4.3, fourth column), the largest set of coherent flows is extracted by employing a voting scheme inspired by the classical Hough transform with discretization step $t_f$; these flows define a set of point-to-point matches $m_{A,G}$ that will be used to infer a preliminary alignment (Fig. 4.3, fifth column).

### 4.2.3. Non-Rigid Registration

The estimation of the non-rigid transformation between the maps is addressed in two steps. A preliminary affine transformation $\hat{F}$ is computed by solving a non-rigid registration problem with known point-to-point correspondences. $\hat{F} = (\hat{s}\hat{R}|\hat{t})$ is computed by solving an optimization problem with cost function the sum of the squared distances between corresponding points (Fig. 4.3, sixth column):

$$C_{reg}(\hat{F}) = \sum_{i=0}^{N} ||p_i - \hat{s}\hat{R}q_i - \hat{t}||^2 \tag{4.4}$$

with $(p_i, q_i) \in m_{A,G}$, $N$ the cardinality of $m_{A,G}$, $\hat{R}$ and $\hat{t}$ the rotation matrix and the translation vector, and $\hat{s}$ is a scaling vector.

To estimate the final registration, two subsets are firstly selected from the input colored point clouds $\mathcal{M}_A$ and $\mathcal{M}_G$, $\mathcal{M}_A^{veg}$ and $\mathcal{M}_G^{veg}$, that includes only points that belong to vegetation. The selection is performed by using an ExG based thresholding operator over $\mathcal{M}_A$ and $\mathcal{M}_G$. This operation enhances the morphological information of the vegetation while reducing the size of the point clouds to be registered. Finally, the target affine transformation $F$ is estimated by exploiting the coherent point drift (CPD) [108] point set registration algorithm over the point clouds $\mathcal{M}_A^{veg}$ and $\mathcal{M}_G^{veg}$, using $\hat{F}$ as initial guess transformation.

### 4.2.4. Experiments

In order to analyze the performance of the proposed system, I acquired datasets on fields of 3 different crop types in Eschikon (Switzerland) - soybean, sugar beet, and winter wheat. For each crop species I collected: (i) one sequence of GPS-IMU tagged images over the entire field from a UAV flying at 10 meters altitude; (ii) 4-6 sequences of GPS/IMU-tagged images of small portions of the field from a UGV point-of-view. Additionally, for the sugar beet field, I acquired an additional aerial sequence of images from 20 meters altitude. More comprehensive details regarding the acquired datasets are reported in Table 4.1.

The UAV datasets were acquired using a DJI Mavic Pro UAV equipped with a 12 MP color camera, while the UGV datasets were acquired moving the same camera by hand with a forward-looking point-of-view, simulating data acquisition by a ground robot. The collected images are first converted into 3D colored point clouds using Pix4Dmapper [3], a professional photogrammetry software suite, which is then aligned using the proposed registration approach. To analyze the performance of the proposed approach, the following error metrics are employed:

$$\delta t = t - \widetilde{t} \quad \delta r = Trace(R^T \cdot \widetilde{R}) \quad \delta s = s \oslash \widetilde{s} \tag{4.5}$$

$$e_t = ||\delta t||_2 \;\; e_R = acos((\delta r - 1)/2) \;\; e_s = ||\delta s||_2 \tag{4.6}$$

where $\oslash$ stands for the element-wise division operator and $(e_t, e_r, e_s)$ are, respectively, the translational, the rotational, and the scale error metrics. Table 4.2 reports the AgriColMap related parameters.

**Table 4.1.** Overview of the Datasets: the global scale error is, in general, bigger in the UGV datasets since the camera is carried by hand, and therefore some GPS satellite signals might be not received.

| Crop Type | Name | # Images | Crop Size (avg.) | Global Scale Error | Recording Height (approx.) |
|---|---|---|---|---|---|
| Soybean | *sUGV A* | 16 | 6 cm | 4% | 1 m |
| | *sUGV B* | 19 | 6 cm | 6% | 1 m |
| | *sUGV C* | 22 | 6 cm | 7% | 1 m |
| | *sUAV* | 89 | 6 cm | 3% | 10 m |
| Sugar Beet | *sbUGV A* | 25 | 5 cm | 6% | 1 m |
| | *sbUGV B* | 26 | 5 cm | 7% | 1 m |
| | *sbUGV C* | 27 | 5 cm | 5% | 1 m |
| | *sbUAV A* | 213 | 5 cm | 3% | 10 m |
| | *sbUAV B* | 96 | 5 cm | 2% | 20 m |
| Winter Wheat | *wwUGV A* | 59 | 25 cm | 9% | 1 m |
| | *wwUGV B* | 61 | 25 cm | 9% | 1 m |
| | *wwUAV* | 108 | 25 cm | 5% | 10 m |

**Table 4.2.** Parameter set

| Parameter | $\alpha$ | $\beta$ | $s$ | $\sigma_{avg}$ | $tf$ |
|---|---|---|---|---|---|
| Value | 1 | .5 | 0.02 $m$ | 0.04 $cm$ | 1 |

### Performance Under Noisy Initial Guess

This experiment is designed to show the robustness of the proposed approach under different noise conditions affecting the initial guess, and different directional scale discrepancies. For each UGV point cloud, an accurate ground truth non-rigid transform is estimated by manually selecting the correct point-to-point correspondences with the related UAV cloud. A random initial alignments between maps is estimated by manually adding noise, with different orders of magnitude, to the ground truth heading, translation, and scale. Then, the clouds are aligend with the sampled initial alignments by using (i) the proposed approach; (ii) a modified version of the proposed approach by moving from the ExG + DSM environment representation to an RGB one (iii) a non-rigid standard ICP, (iv) the coherent point drift (CPD) method [108], (v) a state-of-the art Globally Optimal 3D ICP (Go-ICP) [166], and with standard sparse visual feature matching approaches [13, 136, 26], applied as a data association front-end to the proposed method in place of the proposed LDOF coll_map/imgs/acc_plots_new.png

**Figure 4.4.** Average success registration rate curves by varying the initial guess and the initial scale error: (i) from top left to bottom right, the initial scale error is incrementally increased: $0\%, 10\%, 20\%, 30\%$; (ii) in each plot within the first and third rows, the initial heading error $\delta\psi$ is kept fixed, while the initial translational misalignment $\delta t$ is incrementally increased until 5 meters. For the experimental parts, 5 meters are assumed to be a reasonable upper bound for the initial GPS translational error. (iii) in the second and fourth rows figures $\delta\psi$ is incrementally increased, while the initial translational misalignment $\delta t$ is kept constant. It is important to point out that the successful registration rate of the Go-ICP [166] method is only reported for the cases without an initial scale error since this approach only deals with rigid transformations. For AgriColMap, different results obtained in each dataset are reported (sb: Soybean, sg10: Sugabeet 10m, sg20: Sugabeet 20m, ww: Winter Wheat).

based data association (Sec. 4.2.2): in the last cases, the proposed method exploits only the ExG channel of the grid maps (Sec. 4.2.2). An alignment is considered valid if: $e_t <= 0.05\ m$, $e_r <= 0.1\ rad$, and $e_s <= 2.5\%$.

The results are illustrated in Fig. 4.4. The proposed approach significantly outperforms the other approaches, ensuring an almost 100% success registration rate up to a scale error of 25%, and a high probability of succeeding even with a 30% scale error.

The ICP-based registration methods [108, 166], due to the absence of structural 3D features on the fields, fall into local minima with high probability. The closest methods, in terms of robustness, are based on local feature matching [13][136][26], succeeding in the registration procedure up to a scale error magnitude of 10%. While analyzing the results, however, I verified that, unlike the proposed method, these methods provide a larger number of wrong, incoherent point associations, and such a problem is clearly highlighted for increasing scale deformations above 20% and rotations above 0.1 radians. The superior robustness is also confirmed for noisy initial guesses: unlike the other methods, the proposed approach guarantees a high successful registration rate for a translational error up to 5 meters, and an initial heading error up to 11.5 degrees, enabling it to deal with most errors coming from a GPS or AHRS sensor. The proposed method generalizes well over the different datasets, showing the capability to deal with different crop species, crop growth stages (i.e., the winter wheat crop is in an advanced growth stage compared to the soybean and sugar beet), soil conditions, and point cloud resolution (from different UAV altitudes). An additional important outcome is the higher alignment probability obtained with the ExG/DSM representation over the RGB one.
Table 4.3 reports a comparison between the inliers percentages when using the visual (i.e., the ExG or the RGB) and the geometric terms in the cost function of (4.3). Most of the information is carried by the visual term, especially by the ExG, while the sole geometric term is not able to provide valid results. Nevertheless, when combined, the latter acts as a strong outlier rejection term, improving the robustness properties of the registration procedure. This is true especially for the sugar beet dataset, where the inliers percentage increases quite significantly.

**Table 4.3.** Inliers percentage comparison when changing data terms in the LDOF cost function.

| | Descriptor Type (% inliers) | | | |
|---|---|---|---|---|
| Crop Type | RGB | ExG | Depth | ExG + Depth |
| Soybean | 11.7% ± 4.3% | 53.2% ± 14.9% | 0.2% ± 0.1% | **54.5% ± 13.2%** |
| Sugar Beet | 49.2% ± 11.9% | 64.1% ± 12.8% | 0.4% ± 0.2% | **68.1% ± 13.6%** |
| Winter Wheat | 22.9% ± 9.7% | 51.8% ± 17.4% | 0.1% ± 0.1% | **52.4% ± 16.7%** |

### Accuracy Evaluation

To evaluate the accuracy of the proposed registration approach, the provided results are compared with the ground truth parameters and, by using all the successful registrations, the average accuracy for each crop type and approach is reported. The results are summarized in Tab. 4.5, and are sorted in increasing order of scale error. On average, the proposed method results in a lower registration error as compared to all the other evaluated methods for the same scale error. The difference in the registration error is even more pronounced when comparing the *Sugar Beet 10m* against *Sugar Beet 20m* datasets. Indeed, due to the higher sparseness of the points in the latter, all the other methods tend to perform slightly worse than they do with the *Sugar Beet 10m*. Conversely, the proposed method results in almost the same registration error magnitudes, showing that it correctly deals with the different densities of the initial colored point clouds. Fig. 4.5 also reports some qualitative results.

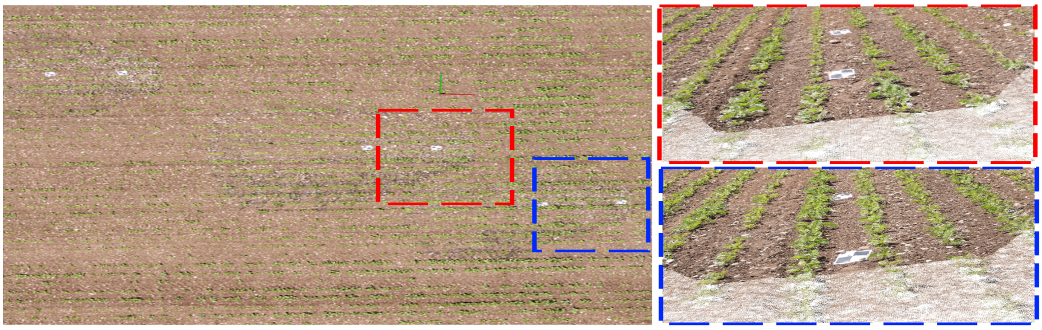


**Figure 4.5.** Qualitative registration results seen from aerial (left) and ground point-of-views. In the former, the UGV clouds are indistinguishable from the UAV, proving the correctness of the registration. Conversely, in the latter, the UGV clouds are clearly visible due to their higher points density.

### Runtime Evaluation

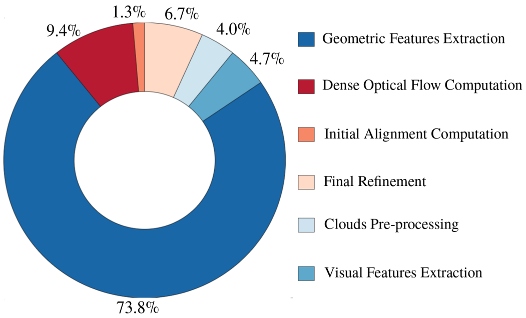


**Figure 4.6.** Average percentage of the total runtime for different parts of the AgriColMap pipeline.

| | Runtime [sec] | | |
|---|---|---|---|
| | Min | Max | Avg |
| *AgriColMap* | 63.7 | 118.6 | 79.8 |
| ICP | 2.1 | 10.6 | 4.5 |
| CPD | 4.9 | 23.2 | 8.2 |
| Go-ICP | 5.3 | 689.2 | 193.1 |
| SURF [13] | 4.6 | 7.2 | 5.3 |
| ORB [136] | 3.9 | 6.7 | 4.8 |
| FAST+BRIEF [26] | 3.7 | 6.4 | 4.5 |

**Table 4.4.** Runtime comparison.

The average, maximum, and minimum computational time are recorded for all tested methods over 100 successful registrations, reporting these values in Tab. 4.4. The method requiring the biggest computational effort is Go-ICP. The proposed approach requires half the computational time as compared to Go-ICP, but turns out to be quite slow compared to the custom-built ICP, and, in general, to all the other matching approaches. Fig. 4.6 shows the runtime percentages for the proposed approach. The biggest component of the computational effort is required to extract the geometric features (i.e., the FPFH features), meaning that the total computational time might be reduced by switching to a less time consuming 3D feature or by using only the visual term.

## 4.3. Summary of Results and Lessons Learned

In this chapter of the thesis, I focus on the collaboration between the ground and aerial vehicles, namely on building and sharing a common environment representation. The related literature is wide, and the problem is usually addressed from a 3D perspective and in an *ad-hoc* fashion, depending on the specific use-case. Conversely, due to the specific characteristic of farming scenarios, I proposed to address the problem from a 2D perspective. Particularly, as described in Sec. **??**, the problem is cast as a non-rigid map registration strategy, where the initial data association problem is addressed in 2D. When registering a map in challenging contexts where the geometric structures are poor, the enhancement of the environment-specific characteristics is crucial. Indeed, by converting each individual map into an *ad-hoc* representation that embeds vegetation density and height, is it possible to filter out most of the redundant data. Such a representation of the environment also allows exploiting 2D registration methods to carry out, in a robust manner, the initial data association. Through the analysis of the success registration rate, I show how the system allows for accurate and reliable results, also in the face of consistent noises in the map scales and in the map geo-location. Moreover, the proposed method may also provide benefits in another context where building a map in a collaborative manner is highly recommended.

**Table 4.5.** Registration accuracy comparison among the proposed approach, the non-rigid ICP, the CPD [108], and the Go-ICP [166] systems. The table reports, for each cell, the average accuracy among all the successful registrations with a specific initial anisotropic scaling error.

| crop type | approach | registration err. (trans/ros/scale) scale error 0% | registration err. (trans/ros/scale) scale error 5% | registration err. (trans/ros/scale) scale error 10% | registration err. (trans/ros/scale) scale error 15% | registration err. (trans/ros/scale) scale error 20% | registration err. (trans/ros/scale) scale error 25% | registration err. (trans/ros/scale) scale error 30% |
|---|---|---|---|---|---|---|---|---|
| Soybean | AgriColMap | 0.03$m$/0.03°/– | 0.03$m$/0.04°/1.9% | 0.04$m$/0.05°/2.0% | 0.04$m$/0.04°/2.1% | 0.03$m$/0.04°/2.2% | 0.05$m$/0.04°/2.2% | 0.05$m$/0.05°/2.3% |
| | ICP | 0.03$m$/0.07°/– | 0.05$m$/0.08°/2.4% | 0.04$m$/0.09°/2.4% | fail | fail | fail | fail |
| | CPD [108] | 0.02$m$/0.03°/– | 0.04$m$/0.07°/2.1% | 0.03$m$/0.08°/2.3% | 0.03$m$/0.08°/2.4% | fail | fail | fail |
| | Go-ICP [166] | 0.03$m$/0.06°/– | - | - | - | - | - | - |
| | SURF [13] | 0.02$m$/0.04°/– | 0.03$m$/0.04°/2.2% | 0.05$m$/0.06°/2.4% | fail | fail | fail | fail |
| | [136] | 0.02$m$/0.04°/– | 0.04$m$/0.04°/2.2% | 0.05$m$/0.05°/2.3% | fail | fail | fail | fail |
| | FAST+BRIEF [26] | 0.04$m$/0.05°/– | 0.05$m$/0.06°/2.3% | 0.05$m$/0.07°/2.4% | fail | fail | fail | fail |
| Sugar Beet 10m | AgriColMap | 0.03$m$/0.04°/– | 0.03$m$/0.04°/2.1% | 0.04$m$/0.04°/2.0% | 0.05$m$/0.06°/2.0% | 0.05$m$/0.07°/2.3% | 0.05$m$/0.1°/2.3% | 0.05$m$/0.1°/2.4% |
| | ICP | 0.04$m$/0.05°/– | 0.05$m$/0.07°/2.1% | 0.05$m$/0.09°/2.4% | fail | fail | fail | fail |
| | CPD [108] | 0.03$m$/0.04°/– | 0.04$m$/0.05°/2.1% | 0.04$m$/0.06°/2.2% | 0.05$m$/0.09°/2.4% | fail | fail | fail |
| | Go-ICP [166] | 0.02$m$/0.05°/– | - | - | - | - | - | - |
| | SURF [13] | 0.03$m$/0.04°/– | 0.03$m$/0.04°/2.1% | 0.04$m$/0.07°/2.3% | fail | fail | fail | fail |
| | ORB [136] | 0.02$m$/0.03°/– | 0.03$m$/0.03°/2.2% | 0.05$m$/0.06°/2.4% | fail | fail | fail | fail |
| | FAST+BRIEF [26] | 0.02$m$/0.04°/– | 0.02$m$/0.03°/2.1% | 0.05$m$/0.06°/2.3% | fail | fail | fail | fail |
| Sugar Beet 20m | AgriColMap | 0.03$m$/0.03°/– | 0.04$m$/0.03°/2.0% | 0.04$m$/0.04°/2.2% | 0.05$m$/0.05°/2.1% | 0.05$m$/0.08°/2.2% | 0.05$m$/0.09°/2.4% | 0.05$m$/0.1°/2.4% |
| | ICP | 0.05$m$/0.06°/– | 0.05$m$/0.09°/2.3% | fail | fail | fail | fail | fail |
| | CPD [108] | 0.04$m$/0.05°/– | 0.05$m$/0.07°/2.3% | 0.05$m$/0.08°/2.4% | 0.05$m$/0.1°/2.5% | fail | fail | fail |
| | Go-ICP [166] | 0.04$m$/0.05°/– | - | - | - | - | - | - |
| | SURF [13] | 0.03$m$/0.04°/– | 0.04$m$/0.05°/2.1% | 0.04$m$/0.06°/2.4% | fail | fail | fail | fail |
| | ORB [136] | 0.04$m$/0.05°/– | 0.04$m$/0.05°/2.2% | 0.04$m$/0.05°/2.4% | fail | fail | fail | fail |
| | FAST+BRIEF [26] | 0.03$m$/0.04°/– | 0.04$m$/0.05°/2.1% | 0.05$m$/0.07°/2.4% | fail | fail | fail | fail |
| Winter Wheat | AgriColMap | 0.04$m$/0.02°/– | 0.04$m$/0.03°/2.0% | 0.04$m$/0.05°/2.1% | 0.04$m$/0.04°/2.2% | 0.05$m$/0.08°/2.3% | 0.05$m$/0.09°/2.4% | 0.05$m$/0.1°/2.4% |
| | ICP | 0.04$m$/0.07°/– | 0.04$m$/0.08°/2.2% | 0.05$m$/0.10°/2.5% | fail | fail | fail | fail |
| | CPD [108] | 0.04$m$/0.05°/– | 0.04$m$/0.05°/1.9% | 0.04$m$/0.05°/2.1% | 0.05$m$/0.09°/2.3% | fail | fail | fail |
| | Go-ICP [166] | 0.03$m$/0.07°/– | - | - | - | - | - | - |
| | SURF [13] | 0.03$m$/0.06°/– | 0.03$m$/0.05°/2.2% | 0.04$m$/0.06°/2.4% | fail | fail | fail | fail |
| | ORB [136] | 0.04$m$/0.05°/– | 0.04$m$/0.04°/2.1% | 0.04$m$/0.06°/2.3% | fail | fail | fail | fail |
| | FAST+BRIEF [26] | 0.03$m$/0.07°/– | 0.04$m$/0.06°/2.3% | 0.05$m$/0.05°/2.4% | fail | fail | fail | fail |

# Chapter 5

# Perception Based Control

In the previous chapters of this thesis, I addressed several perception related challenges in PA contexts, such as a crop/weed segmentation module, a localization, and mapping architecture and a collaborative mapping framework. The above-mentioned methods may already provide a good backbone for a possible multi-robot agricultural application, such as the one described in the Flourish project. Such a robotic system could be already capable of safely moving within the environment, distinguishing between crop and weeds, and building a shared map of the environment.
Nevertheless, problems may arise when the robot is controlled to carry out a targeted task, such as pruning, applying targeted treatments, or performing targeted inspections. In this setting, indeed, the required maneuvers are often carried out to move a vehicle from a specific pose configuration to another one to perform certain agronomic interventions and/or to collect data and, therefore, the goal pose may depend on the agronomic target location into the field. These problems are usually tackled by employing visual serving approaches, where the robot is steered by using feedbacks coming from visual sensors [147]. However, the specific characteristics of farming scenarios, such as the high pattern repetitiveness, and the strong sensor aliasing, might possibly mislead the on-board navigation system. Moreover, obstacles may lie or suddenly appear, along the desired route leading the robot to perform reactive avoiding maneuvers. A viable solution to overcome the above-mentioned issues is to explicitly take into account those constraints in the control loop.
This chapter of the thesis proposes a few possible approaches to address these issues, with a specific focus on UAVs. In recent years, small Unmanned Aerial Vehicles (UAVs) have increasingly gained popularity in many practical applications thanks to their effective survey capabilities and limited cost. For some basic applications, solutions are already available in the market to localize the drone and provide some basic navigation and obstacle avoidance capabilities. However, to safely navigate in the presence of obstacles, an effective and reactive planning algorithm is an essential requirement.

Thanks to the progress in perception and control algorithms [59][110], and to the increased computational capabilities of embedded computers, vision-based optimal control techniques became a standard for UAVs moving in dynamic environments [43][147]. They allow mitigating some of the vision-based perception limitations (e.g. feature tracking failures) through an ad-hoc trajectory planning and have to some

extent solved the vehicle state-estimation problem that, in the last decades, has been commonly faced with motion capture systems. However, the problem of addressing perception and obstacle avoidance together has been rarely investigated [42].
In this chapter, I report three contributions related to perception-based control:

1. An effective target aware visual navigation for UAVs, which splits the control loop into two independent parts, addressing the perception-based planning and the control part, respectively;

2. A non-linear model predictive control with adaptive time-mesh refinement, where we exploit an adaptive strategy to reduce the computational burden that affects optimal controllers when dealing with multiple constraints;

3. A joint vision-based navigation, control and obstacle avoidance for UAVs in dynamic environments, where we develop a full control stack that incorporates both perceptions and obstacles constraints.

An important and preliminary remark concerns the applicability of the abovementioned contributions in a farming setting. Those methods, indeed, have only been tested in indoor and simulated environments, thus the results can be only considered as a preliminary step in the direction of a possible application in precision agriculture. Further tests and evaluation will be subject of future investigations.

## 5.1. Related Work

A vision-based UAV needs three main components to effectively navigate in a dynamic environment:

1. A reactive control strategy, to accurately track a desired trajectory while reducing the motors effort;

2. A reliable collision avoidance module, to safely explore the environment even in the presence of dynamic, unmodeled obstacles;

3. An adaptive, perception aware, on-line planner, to support the vision-based state estimation or to constantly keep the line-of-sight with a possible reference target.

A wide literature addressing individually, or in pairs, these requirements is available, among others: [79][157][11][153]. However, they have rarely been addressed together, in particular when dealing with unexpected and moving obstacles.
Requirement 1) is an essential capability for highly dynamic vehicles such as UAVs, hence extensively covered in literature, and often formulated as an OCP [31]. Model Predictive Controller (MPCs) is a well-known control technique capable to deal with OCPs, and have recently gained great popularity thanks to increased onboard computational capabilities of embedded computers. In [71] and [130] ACADO, a framework for fast Nonlinear Model Predictive Control (NMPC), is presented; [78] uses ACADO for fast attitude control of UAVs. In [59], the authors addressed the reactive control problem by building upon a flatness-based Model Predictive Control:

the approach converts the optimal control problem in a linear convex quadratic program by accounting for the non-linearity in the model through the use of an inverse term. Experiments performed in simulation and real environments demonstrate improved trajectory tracking performance. In [110], the authors propose to employ an iterative optimal control algorithm, called Sequential Linear Quadratic, applied inside a Model Predictive Control setting to directly control the UAV actuation system.

The collision-free trajectory generation (requirement 2) is usually categorized into three main strategies: search-based approaches [77][133], optimization-based approaches [157][114], path sampling and motion primitives [106][118]. In [114] the authors propose a motion planning approach capable to run in real-time and to continuously recompute safe trajectories as the robot perceives the surrounding environment. Although the proposed method allows the UAV to replan at a high rate and react to previously unknown obstacles, it might be vulnerable to vision-based perception limitations.

Steering a robot to its desired state by using visual feedback obtained from one or more cameras (requirement 3) is formally defined as Visual Servoing (VS), with several applications within the UAV domain [123][102][90][42]. Among the others, in Falanga *et al.* [42], the authors address the flight through narrow gaps by proposing an active-vision approach and by relying only on onboard sensing and computing. The system is capable to provide an accurate trajectory while simultaneously estimating the UAV's position by detecting the gap in the camera images. Nevertheless, it might fail in the presence of unmodeled obstacles along the path.

A fully autonomous UAV navigating in a cluttered and dynamic environment should be able to concurrently solve *all* the three problems listed above. A solution could be to combine three of the methods presented above, to deal with each problem individually. Unfortunately, due to poor integration between methods and the overall computational load, this solution is not easily feasible. Jointly addressing a subset of these problem is a topic that is recently gathering great attention: in [147], the authors propose to encode in the NMPC cost function the image feature tracks, implicitly keeping them in the field of view while reaching the desired pose. In Falanga *et al.* [43], the authors propose a different version of NMPC that also takes into account the features velocity in the camera image plane. The controller will eventually steer the vehicle keeping the features as close as possible to the image plane center, while minimizing their motion. This mitigates the blur of the image due to the camera motion, aiding the target detection and the features tracking. However, the methods presented so far in general do not guarantee a fully autonomous flight in cluttered environments or in presence of unmodeled obstacles. In [55], the authors propose a NMPC which incorporate obstacles in the cost function. To increase the robustness in avoiding the obstacles, the UAV trajectories are computed taking into account the uncertainties of the vehicle state. Kamel *et al.* [79] deal with the problem of multi-UAV reactive collision avoidance. They employ a model-based controller to simultaneously track a reference trajectory and avoid collisions. The proposed method also takes into account the uncertainty of the state estimator and of the position and velocity of the other agents, achieving a higer degree of robustness. Both these methods show a reactive control strategy, but might not allow the vehicle to perform a vison-based navigation.

## 5.2. Effective target aware visual navigation for UAVs

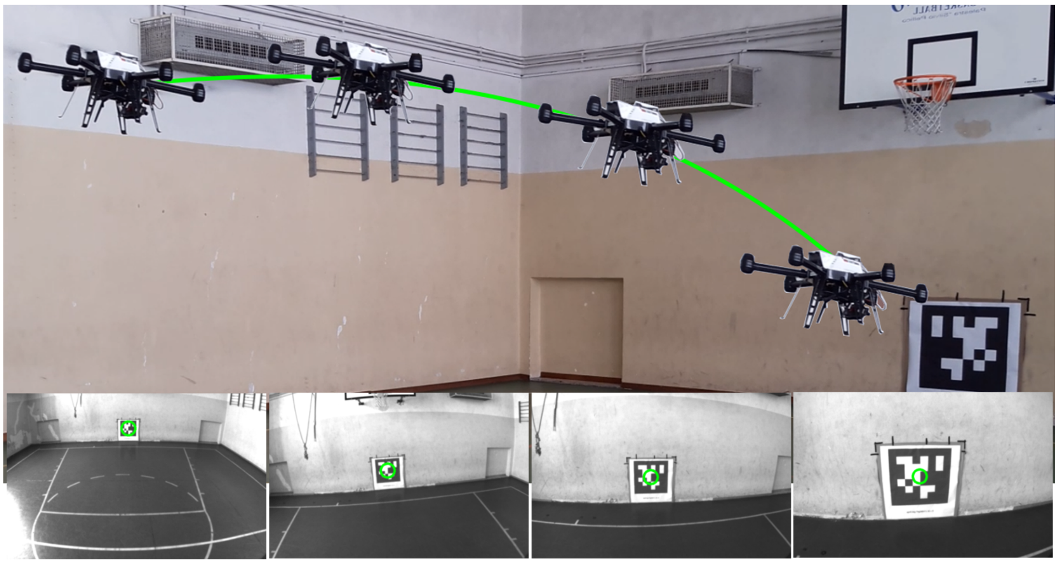


**Figure 5.1.** An example of target aware visual navigation: the UAV is following an optimal trajectory towards the target  while constantly framing the target with the camera.

In this section, I propose an effective and robust VS controller that allows a UAV to perform fast maneuvers without losing the line of sight with the target of interest during the entire duration of the flight.
VS techniques can be split into two parallel branches: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). In PBVS, the 3D goal pose is directly obtained from a complete 3D reconstruction of the surrounding environment or from the 6D position of one or more landmarks placed in it. In contrast, IBVS formulates the problem in terms of image features locations: the goal pose is defined by means of desired features locations in the final image while the control law aims to minimize the features re-projection error during the flight. Even if IBVS does not require any full 3D estimation, it still needs the depth of the target. It has been shown that both strategies have their own weaknesses. In IBVS it is particularly challenging to model the relation between the vehicle dynamics and the feature projection error, especially for under-actuated systems. Furthermore, inaccurate estimation of the object depth leads to instabilities. In PBVS, since the control law is directly designed in the state-space domain, there is a close dependence on the accuracy of the 3D environment reconstruction or on the target pose estimation. In practice, this estimation may be very noisy, leading PBVS to be very sensitive to initial conditions, camera calibration parameters and image noise corruption. Differently from the literature, I propose a procedure that decouples the planning and the control problems. The planning task is addressed by employing a hybrid approach. Firstly, as in PBVS, the proposed method gets the goal pose as the

position and the relative orientation of the vehicle in the environment that allows having the desired view of the target object.

Then, similarly to IBVS, the proposed approach models the trajectory as a non-linear constrained optimization problem with a cost function that penalizes the target's location error, in order to constantly keep the target in the camera field of view.

Once a global optimal trajectory[1] has been found, we employ an NMPC framework as the controller and local planner. Making use of an efficient open-source solver, the proposed control framework is capable to solve an NMPC problem in few milliseconds, allowing us to use at each time step just the initial tuple of control inputs[2] while simultaneously resolving the whole non-linear control problem.
The proposed method is compared against a common PBVS approach in both simulated and real environments, getting in all experiments cutting edge results. Additionally, a preliminary assessment is made with respect to a state-of-the-art Optimal Visual Servoing (OVS) technique, suggesting that the proposed approach can achieve comparable results.

### 5.2.1. UAV Dynamic Model

In the following, the vehicle dynamics equation used as constraints in the optimal trajectory computation will be described.Let $x_Y^Z$ be the position of the reference frame $Y$ expressed with respect to the reference frame $Z$. Furthermore, let express a rotation matrix from the reference system $Y$ to the reference system $Z$ as $R_Y^Z$. For the trajectory planning and the control of the multirotor vehicle, three main coordinate reference systems will be used: (i) the camera frame with $C$; (ii) the world fixed inertial frame with $I$; (iii) the body-fixed frame with $B$, that is the frame attached to the Center of Gravity (CoG) of the UAV. The UAV configuration at each time step is formulated by the position $p_B^I$ and the linear velocities $v_B^I$ of the vehicle CoG, both expressed in the inertial frame, and the vehicle orientation $q_B^I$. More specifically, the whole state of the vehicle is then expressed as $x = \{p_B^I, q_B^I, v_B^I\}$. At each time step, the tuple of control inputs are also defined as $u = \{\phi_{cmd}, \theta_{cmd}, \dot{\psi}_{cmd}, T_{cmd}\}$, where the single terms stands for, respectively, the roll, pitch and yaw rate desired commands and the commanded thrust.

The proposed method employs a widely used dynamic model for multirotor, where the main forces that act on the vehicle are generated from the propellers. More specifically, each propeller generates a thrust force $F_T$ proportional to the square of the motor rotation speed. Moreover, also two other important effects that became relevant in the case of dynamic maneuvers are taken into account, namely *blade flapping* and induced *drag*. Both of them introduce additional forces in the x-y rotor plane [99]. The considered dynamic model formulates them into a single lumped *drag* coefficient $K_D$, as shown in [116][24], leading to the aerodynamic force $F_{aero,i}$:

[1] With abuse of notation, as in other related work, the term "optimal trajectory" is referred to the desired trajectory the multirotor tracks during the flight. Actually, due to the non-linear nature of the cost function, the optimization does not always guarantee the convergence to the optimal, global minimum.

[2] An NMPC provides a sequence of control inputs for a finite temporal horizon.

$$F_{aero,i} = F_{T,i} K_{drag} {R_B^I}^T v_B^I \tag{5.1}$$

where $i$ stands for the propeller index, $K_{drag} = diag\{K_D, K_D, 0\}$, $F_{T,i}$ is the $z$ component of the $i-th$ thrust force and $v_B^I$ is the vehicle's linear velocity (in the next equations, where there is no confusion, superscripts and subscripts $I$ and $B$ will be both omitted). The final dynamic model of the vehicle can be expressed as follows:

$$\dot{p} = v, \tag{2.$a$}$$

$$\dot{v} = \frac{1}{m}\Big(R\sum_{n=0}^{n_p}(F_{T,i} - F_{aero,i}) + F_{ext}\Big) + g, \tag{2.$b$}$$

$$\dot{\phi} = \frac{1}{\tau_\phi}(k_\phi \phi_{cmd} - \phi) \tag{2.$c$}$$

$$\dot{\theta} = \frac{1}{\tau_\theta}(k_\theta \theta_{cmd} - \theta) \tag{2.$d$}$$

$$\dot{\psi} = \dot{\psi}_{cmd} \tag{2.$e$}$$

where $m$ is the mass of the vehicle, $F_{ext}$ are the external forces that act on the multirotor. The proposed system also employs a low-level controller that maps the high-level attitude control inputs in propellers' velocity, as the one provided with the Asctec NEO hexacopter used in the experiments. To achieve better tracking performance, this inner control loop is modeled as first-order dynamic systems, where the model parameters $\tau_i$ and $k_i$ are obtained by a system identification procedure [80].

### 5.2.2. Optimal Visual Servoing (OVS)

In this section, I describe how the dynamics and perception constraints are taken into account when planning a trajectory and controlling the multirotor. The first step is the goal pose computation, namely the position and orientation of the vehicle that allows getting the desired view of the target. The proposed approach splits the Optimal Visual Servoing (OVS) problem into two consecutive stages. First, an optimal global trajectory is computed by solving a non-linear optimization problem. In order to take into account the dynamic and perception constraints, the output trajectory is minimized over the multirotor dynamics and the target re-projection error in the image plane. To track the desired trajectory the proposed method then employs a Receding Horizon NMPC controller, where a smaller non-linear optimization problem is solved every time step and only the first control input is actually sent to the multirotor.

### 5.2.3. Goal Pose Computation

Before computing the optimal trajectory, the multirotor has to retrieve the goal pose it aims to reach. Such a pose depends on the task (e.g., inspection or patrolling) and it usually requires the vehicle to frame a target (e.g., landmark or object) from a specific distance and with a specific point of view. Retrieving a relative

3D transformation from the camera is a well-known problem and has been widely studied in the last decades. A widely used technique is based on the solution of a Perspective-n-Point ($PnP$) problem [125]: such a technique requires prior knowledge about the target object geometry and scale.

Since the choice of the goal pose computation algorithm goes behind the purpose of this work, I assume for the sake of simplicity to have a real-time ”black-box” detection framework that outputs: (i) the $(u, v)$ pixel coordinates of the target $T$ in the camera image plane; (ii) the 3D position of the target in the camera frame $p_T^C$; (iii) the orientation $q_C^T$ of the target object with respect to the camera frame $C$ in terms of $yaw$ angle. The goal pose in world $I$ reference system can be then obtained as follows:

$$p_{goal}{}_B^I = \ p_B^I + q_B^I( \ q_C^B(d_T^C - p_T^C \ ) + p_C^B) \tag{3.a}$$

$$q_{goal}{}_B^I = q_B^I q_C^B q_T^C \tag{3.b}$$

Where $d_T^C$ is the desired position of the target expressed in the camera frame, while $p_C^B$ and $q_C^B$ are the extrinsic calibration parameters between the camera frame and the body frame.

### 5.2.4. Optimal Trajectory Computation

Once the goal pose has been computed, the proposed approach needs to generate a discrete trajectory composed by $N$ tuples of the vehicle state vector $\{x_0, ..., x_N\}$ and control inputs $\{u_0, ..., u_N\}$ that minimize the functional cost $J$ subject to the vehicle dynamics equations $f(x_k, u_k)$ described in Sec. 5.2.1. The time step of such dynamic equations is given by $\frac{t_f - t_0}{N}$, where $t_f$ and $t_0$ are respectively the final time and the initial time, while $N$ is the number of steps. Additionally, the custom choice of the time variable allows us to define also the nominal speed $s_{nom}$ (i.e. $\frac{\Delta_p}{t_f}$), namely the speed the vehicle is expected to fly. Similarly to [147], the cost function is defined as:

$$J(x_{0:N}, u_{0:N-1}) = J_N(x_N) + \sum_{k=0}^{N-1} J_k(x_k, u_k) \tag{4.a}$$

where $J_N$ is the final cost and $J_k$ is the cost along the trajectory. At this point, $J_k$ is split into two main terms. The first one represents the cost over the desired final state and the control effort, and it can be expressed as follow:

$$J'_K(x_k, u_k) = \frac{1}{2}(x_k - x_N)^T Q(x_k - x_N) + \frac{1}{2} u^T R u \tag{4.b}$$

where $Q \geq 0$ and $R \geq 0$ are the matrices that weight the control objectives. In addition, in the second term of $J_k$ a cost that aims to penalize the re-projection error of the target into the camera field of view is introduced. The entire cost in the discrete time step $k$ can be then formulated as:

$$J_K(x_k, u_k) = J'_K(x_k, u_k) + \frac{1}{2} e_i(x_k)^T H e_i(x_k) \tag{4.c}$$

$$e_i(x_k) = \mathcal{P}(x_k, P_i, \pi) - p_i \tag{4.d}$$

where $H \geq 0$ is the penalization term over the target re-projection error and $\mathcal{P}$ is a general camera projection function. Starting from the 3D position of the object in the camera frame $P_i$, the re-projection error is obtained by the knowledge of the intrinsic calibration parameters of the camera, denoted in Eq. 4.d as $\pi$, the extrinsic parameters between the camera reference system $C$ sensor and the body frame $B$, and the desired position of the target object in the image plane $p_i$. Differently from [147], the proposed approach makes use of a weighting matrix H in place of a scalar weighting factor, allowing us to scale the different components of the re-projection error. Ideally, one wants to have a $H$ that penalizes mostly the error along the smaller dimension of the input image. $H$ is set as follows:

$$H = \begin{pmatrix} h_x & 0 \\ 0 & h_y \end{pmatrix} \tag{4.$e$}$$

Let $h_{i=x,y}$ be the scale factor related to the smaller dimension $(d_i < d_j)$, it is set as follows:

$$h_i = h_j \times \sigma, \quad \sigma = \frac{d_j}{d_i} \tag{4.$f$}$$

This enables the UAV to cope with different camera sensor setups. Since the proposed method introduces in the cost function $J$ the re-projection error term of the target with respect to the camera image plane, the optimal solution will implicitly allow the vehicle to constantly face the target, maintaining it as close as possible to the center of the image plane.

### Optimal Control Solver

Once the optimal trajectory has been obtained, the multirotor must closely follow it. To this end, the proposed method employs an NMPC that repeatedly solves the following optimal control problem:

$$\begin{aligned} \min_{u,x} \sum_{k=0}^{K-1} (\|x_k - x_f\|_Q^2 + \|u_k - u_f\|_R^2) + \|x_K - x_f\|_P^2 \\ \text{subject to: } & x_{k+1} = f(x_k, u_k) + f_{ext}(d_k) \\ & d_{k+1} = d_k \\ & U_{min} \leq u_k \leq U_{max} \\ & x_0 = x_{init} \end{aligned} \tag{5}$$

where $Q \geq 0$ is the weight factor over the state, $R \geq 0$ is the weight factor over the control inputs and $P$ is the weight factor over the final state. The controller is implemented in a receding horizon fashion, meaning that the aforementioned optimization problem is solved every time step over the fixed time interval $[i, i+K]$. Once the optimization problem has been solved, the optimization procedure is repeated for the time interval $[i+1, i+K+1]$ starting from the state reached in $i+1$ and by using the previous solution as the initial guess. By solving this optimization procedure in real-time, the proposed framework simultaneously provides a feed-forward trajectory toward the desired state and a discrete set of control inputs which will be used by the low-level onboard controller. This means that, in practice,

**Table 5.1.** Comparison of simulated image error trajectory statistics for each method across different nominal speeds[3].

| | | Avg. Pixel Error | | | Max. pixel error | | |
|---|---|---|---|---|---|---|---|
| $t_f$ | $S_{nom}$ | NMPC OVS | PBVS | Sheckells *et al.*[147][3] | NMPC OVS | PBVS | Sheckells *et al.*[147][3] |
| 10.2 | 2.01 | **32.5** | 89.2 | ~63.8 | **55.05** | 145.36 | ~127.6 |
| 7.5 | 2.73 | **45.03** | 109.4 | ~85.3 | **76.91** | 199.21 | ~195.1 |
| 6.6 | 3.10 | **55.2** | 125.2 | ~90.7 | **98.76** | 223.67 | ~207.9 |
| 5.1 | 4.02 | **62.5** | 146.9 | not available | **115.64** | 323.78 | not available |

**Table 5.2.** Comparison in terms of control effort between a standard PBVS approach and the proposed one.

| | | RMS Thrust (g) | | RMS Roll Ref. (deg) | | RMS Pitch Ref. (deg) | | RMS Yaw Rate (rad/s) | |
|---|---|---|---|---|---|---|---|---|---|
| $t_f$ | $S_{nom}$ | NMPC OVS | PBVS | NMPC OVS | PBVS | NMPC OVS | PBVS | NMPC OVS | PBVS |
| 10.2 | 2.01 | 10.8 | 10.56 | 0.15 | 0.11 | 0.08 | 0.075 | 0.34 | 0.33 |
| 7.5 | 2.73 | 10.95 | 10.79 | 0.31 | 0.25 | 0.33 | 0.29 | 0.43 | 0.46 |
| 6.6 | 3.10 | 11.21 | 10.98 | 0.5 | 0.43 | 1.47 | 0.38 | 0.49 | 0.48 |
| 5.1 | 4.02 | 11.54 | 11.13 | 0.9 | 0.75 | 1.82 | 0.69 | 0.61 | 0.60 |

at the end of each optimization procedure only the first control input tuple is actually sent to the multirotor controller, then the optimization procedure is repeated.

### 5.2.5. Simulation Experiments

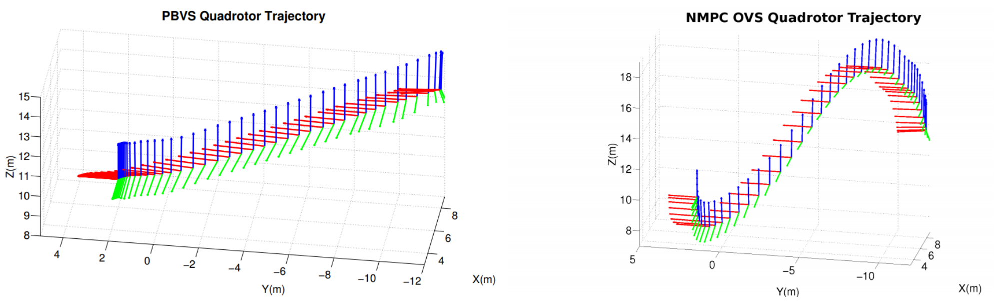


**Figure 5.2.** Example of trajectories obtained using PBVS(a) and NMPC OVS(b) with $s_{nom} = 3.10\frac{m}{s}$: the latter constantly takes into account the target pose during the flight.

The proposed framework is firstly tested in a simulated environment by using the RotorS simulator [53] and a simplified multirotor model with a front-facing camera. The mapping between the high-level control input and the propellers velocities is done by a low-level PD controller that aims to resemble the low-level controller that runs on the real multirotor. From the higher controller level point of view, I implemented a receding horizon NMPC [110], where the optimization problem is solved by means of the efficient ACADO solver [71]. To demonstrate the effectiveness of the proposed method, I fix the number of segments $N$ and then flew the virtual vehicle to the desired goal pose by using the approach described in section 5.2.2 and a standard PBVS technique. The latter adopts a linear interpolation technique between the starting and the goal poses obtaining a vector of $N$ intermediate poses. Such interpolated poses are then sent to the same NMPC that computes the trajectory to track them. Once the final time $t_f$ has been fixed, by tuning $N$ it is possible to act on the flight behavior: increasing the number of segments will involve smoother trajectories and control inputs since the delta-pose between two adjacent desired states segments is smaller. On the other hand, increasing $N$ also brings to higher computational costs when performing the optimal trajectory computation. I used $N = 55$ as a trade-off between smoothness and computational velocity.

The goal pose is computed for each run employing an April Marker [115] attached on a virtual building. Since the aim is to test the proposed approach with different levels of aggressive maneuvers, I act on the $S_{nom}$ parameter (i.e. changing the final time $t_f$). In all the experiments the initial state is set to $x = \{8, -12, 14, 0, 0, 1.918, 0, 0, 0\}$. Since the target is always kept in the same location inside the virtual environment, the computed goal state is $x = \{6 + w_x, 2 + w_y, 9.4 + w_z, 0, 0, 1.57 + w_{yaw}, 0, 0, 0\}$, where $w \in \mathbb{R}^4$ is a small white noise random component due to the target detection errors. The relative transformation between the initial and the final pose forces the multirotor to retrieve an optimal trajectory along the 4 principal motion directions of the vehicle.

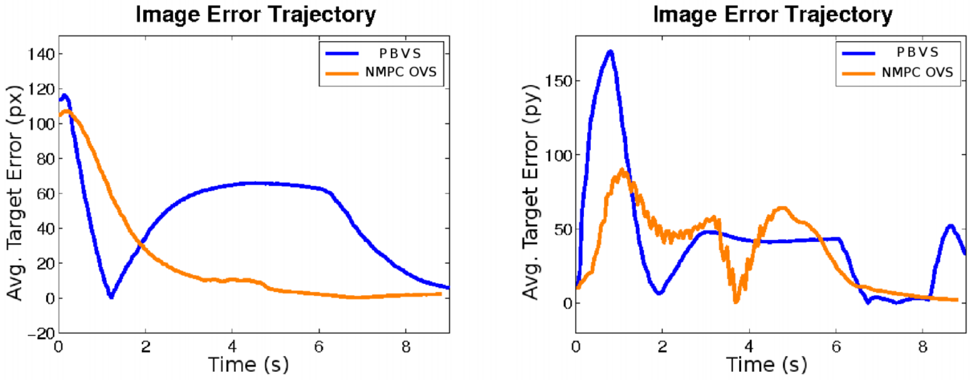


**Figure 5.3.** Comparison of simulated PBVS and NMPC OVS pixel error trajectories for $s_{nom} = 3.10\frac{m}{s}$. Respectively error on the x-axis of the image plane in the left image, while in the right one the pixel error on the y-axis.

## Results

Quantitative image error trajectories for the OVS and PBVS methods for various values of $t_f$ are reported in in Table 5.1. The reported results are obtained averaging the performance of PBVS and OVS given the same goal pose and starting from the same initial state for multiple trials. As a preliminary assessment, the experiments also report some results from the experiments in Sheckells *et al.* [147], showing that the proposed method can provide results comparable with this state-of-the-art approach. It is important to highlight that, given this data, a direct comparison with [147] is not possible, since the pixel error statistics are strictly correlated with the simulation setup which has not been released by the authors.

Remarkably, the target error trajectory along both image axes is almost always lower than both the other approaches. In spite of this, from a qualitative point of view (see Fig. 5.3), the PBVS trajectory seems to behave better in terms of pixel errors at the same points. The explanation for such behavior comes from the different shapes of the two trajectories. In this case, the vehicle is steered to avoid the target to leave the center of the camera field of view, preferring a constant and possibly small error. In the PBVS case, the trajectory is straightforward, involving bigger errors in the acceleration and deceleration phases, worst average and maximum errors, but sporadically smaller error compared with the OVS approach.

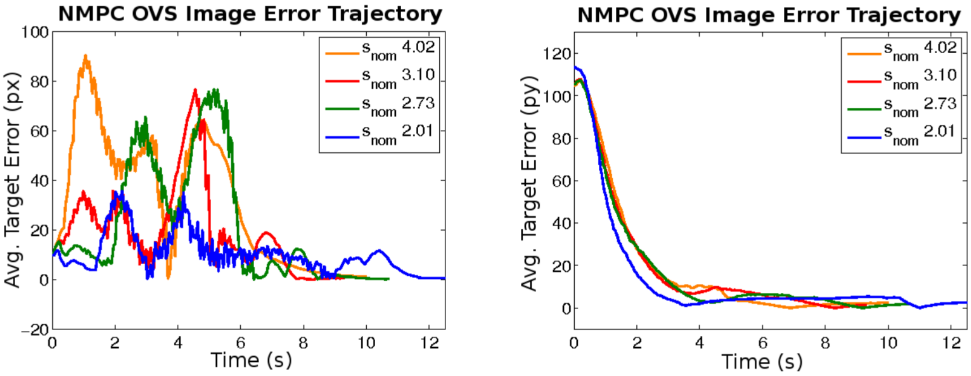


**Figure 5.4.** Comparison of simulated NMPC OVS pixel error trajectories for different values of $s_{nom}$. Respectively error on the x-axis of the image plane in the left image, while in the right one the pixel error on the y-axis.

From the control inputs point of view, the reduced pixel error comes with an energy effort trade-off: as reported in Table 5.2, the RMS thrust of each OVS trajectory is larger with respect to the corresponding PBVS trajectory. A similar conclusion can be drawn from the attitude points of view since maintaining the target in the center of the camera involves larger angles and yaw rate commands. The choice of the correct behavior depends on the task requirements and, by acting on the optimization parameters $Q_k$, $R_k$ and $H_k$, it is possible to obtain the desired trade-off between control effort and image error. Two samples of the trajectories generated by the approaches are depicted in Fig. 5.2. Often the bigger pixel error terms in a VS scenario occur in the initial and in the final phase, due to the attitude components required to accelerate and decelerate the vehicle. As qualitatively reported in Fig. 5.2, the OVS trajectory takes into account these two error sources by a small ascending phase at the same time as the forward pitch command. Similarly, the trajectory dips softly at the end of the flight so that the target remains in the center of the image plane when the multirotor has to pitch backward in order to decelerate. From the PBVS point of view, the trajectory is more or less a straight line. The vehicle starts suddenly to pitch and to decrease its altitude, involving a bigger pixel error.

[3] I emphasize that the statistics from [147] have been obtained with a different simulation setup, so they represent an indicative performance measure.

### 5.2.6. Real Experiments

The proposed framework is tested on an Asctec NEO hexacopter Fig. 5.5 equipped with an Intel NUC i7, where I implemented the proposed algorithm in ROS (Robotic Operating system), running on Ubuntu 14.04. The overall weight of the vehicle is 2.8 Kg. For the state estimation, I make use of a forward-looking VI-Sensor [112] and the ROVIO (Robust Visual Inertial Odometry) framework [20]. The ROVIO output is then fused together with the vehicle Inertial Measurement Unit (IMU) by using an Extended Kalman Filter (EKF) as described in Lynen *et al.* [98]. The control inputs obtained at each time step by the proposed approach are then sent to the low-level onboard controller by using the UART connection.

Several experiments have been carried out by acting on the $s_{nom}$ parameter. In each run, the multicopter starts from the same initial state before starting to look for the fixed target. The distance between the vehicle and the target is approximatively 8 m, and the environment is an indoor closed building.

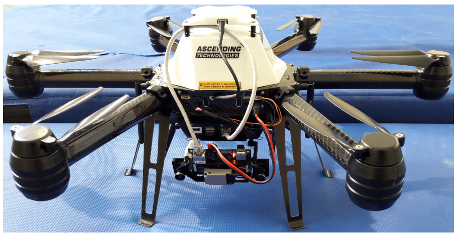


**Figure 5.5.** The Asctec NEO hexacopter used for the real experiments.

| | | Avg. Pixel Error | |
|---|---|---|---|
| $t_f$ | $S_{nom}$ | NMPC OVS | PBVS |
| 6 | 1.79 | **62.89** | 77.2 |
| 5 | 2.11 | **88.03** | 112.4 |
| 4 | 2.73 | **104.17** | 146.7 |
| 3 | 3.21 | **123.8** | 179.2 |

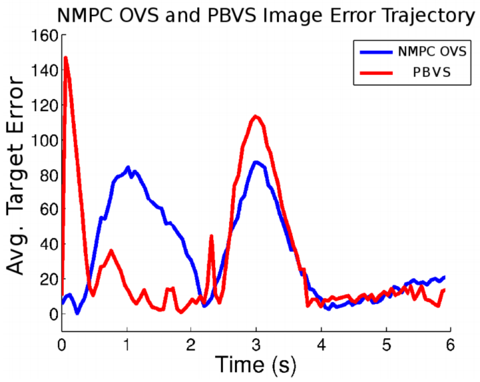
**Table 5.3.** Comparison of image error for each method across different nominal speeds.

#### Results

Qualitative results for trajectories with different values of $s_{nom}$ are reported in Fig. 5.6, while table 5.3 reports the average error statistics between OVS and PBVS. Apart from the same error behavior that also appears in the simulation experiments, it is possible to note how the Mean Squared Error (MSE) pixel error for the OVS approach is lower than the PBVS approach. The difference in terms of pixel MSE is bigger in the initial and final phases where the PBVS, in order to accelerate and slow down, is subject to the greatest kinematic movements in terms of roll and pitch angles. It is also useful to highlights how both OVS and PBVS use a noisy target detection approach. In practice, the multirotor is not able to obtain an accurate 3D position of the target object in the environment. This accounts for a constant non-zero pixel MSE, even when the vehicle reaches the goal state.

## 5.3. Non-Linear Model Predictive Control with Adaptive Time-Mesh Refinement

In this section, I provide an effective solution to NMPC based real-time control problems, by proposing a novel adaptive time-mesh refinement strategy employed



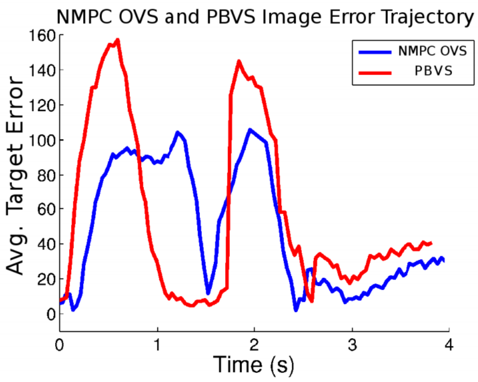


**Figure 5.6.** Comparison of real PBVS and NMPC OVS Mean Squared Error (MSE) pixel trajectories for different values of $s_{nom}$. Respectively the MSE for $s_{nom} = 2.07$ in the left image, while in the right the MSE for $s_{nom} = 3.1$.

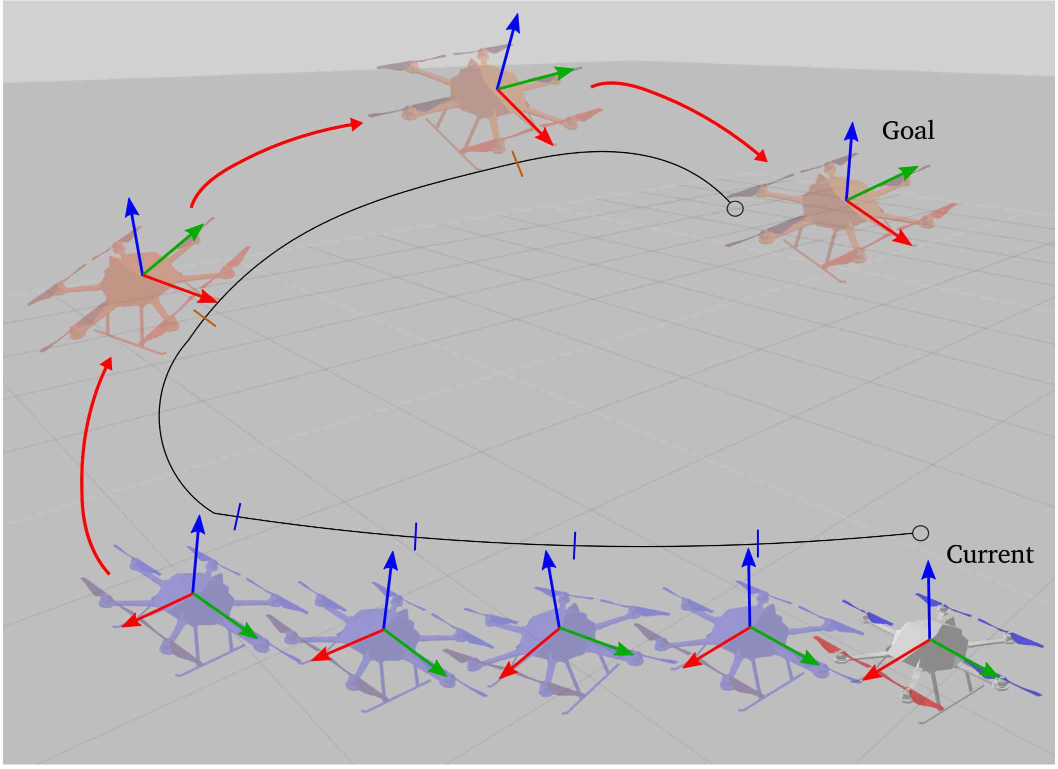


**Figure 5.7.** Illustration of the optimal trajectory computed at each time instant by the proposed approach. UAV frames represent trajectory points along the prediction horizon $T$: the blue frames represent the refined portion of the lattice, while the red ones correspond to the coarser rest of trajectory.

for solving the OCP implemented, and by releasing an open-source implementation of the proposed NMPC strategy.

The proposed approach formulates the task of steering a robot to a goal position, or along a desired trajectory, as a least square problem, where a cost over the dynamics and trajectory constraints is minimized. The optimization is performed over a discrete-time parametrization in terms of the flat outputs [48] of the dynamical system, thus decreasing the dimensionality of the OCP while reducing the computational effort required to find the solution.

In standard OCPs, an approximate solution to the continuous problem is found by discretizing the underlying control problem over a uniform time lattice. The number of discretization points is one of the major factors influencing the accuracy of the solution and the computational time: I propose to mitigate this problem by focusing also on the *points location*, so to condense more time samples where the discretization error is high.

I propose an algorithm that iteratively finds a suitable points distribution within the time-mesh, that satisfies a discretization error criterion. To keep the computational burden as low as possible, the proposed approach increases the local mesh resolution only in the initial parts of the receding horizon, thus providing a higher accuracy in the section of trajectory that is more relevant to the NMPC.

The implementation allows testing the proposed approach on different robotic platforms, such as Unmanned Aerial Vehicles (UAVs), and ground robots, with both holonomic and non-holonomic constraints.
I exploit this implementation in the reported experiments, where the proposed approach is evaluated inside a simulated environment using an Asctec Firefly UAV. I prove the effectiveness of the proposed method with performance comparison against a standard NMPC solution.

### 5.3.1. Adaptive Finite Horizon Optimal Control

#### Problem Statement

The goal of this work is to generate an optimal trajectory along with the set of control inputs to enable a robot to closely track it. This non-linear OCP is formulated in an NMPC fashion over the finite time horizon $T$. A general non-linear time-step rule of the dynamics is assumed as

$$x_{k+1} = f(x_k, u_k) \tag{5.2}$$

where $x_k$ and $u_k$ denote the state vector and the control input vector at the time $t_k$, respectively. $f(x_k, u_k)$ represents the non-linear dynamic model. It is assumed to be differentiable with respect to the state $x_k$ and control input $u_k$, and it maps the state and the input vector in the subsequent time $t_{k+1}$. The time-step of the dynamics is given by $T/N$, so that each $x_k$ occurs at time $t_i = it_f/N$, where $N$ is the number of trajectory points used for finding the optimal solution and $t_f$ indicates the desired final time. The goal is to find an optimal time-varying feedback and feedforward control law of the form:

$$u(x_k, k) = ufb(x_k) + uff(x_k) \tag{5.3}$$

where $uff(x_k)$ is the feedforward term while $ufb(x_k)$ represents the feedback control term. The optimal time-varying control law is found by iteratively finding an optimal solution for minimizing a cost function $\mathcal{C}$.

### Overview of the Control Framework

The cost function $\mathcal{C}$ is composed by a set of constraints over $N$ time steps of the state vector $x_{0:N} = \{x_0, x_1, \ldots, x_N\}$ along with the corresponding input controls $u_{0:N-1} = \{u_0, u_1, \ldots, u_{N-1}\}$. Defining $x_f$ as the desired goal state, the cost function can be formulated in a standard optimal control form as:

$$\hat{\mathcal{C}} = \frac{1}{2} \sum_{k=0}^{N-1} \left( \|x_k - x_f\|_Q^2 + \|u_k\|_R^2 \right) \tag{5.4}$$

where $\| \cdot \|_\Omega^2$ stands for the $\Omega$-norm, $Q \geq 0$ is the matrix that penalizes the distance to the goal state, and $R > 0$ is the matrix weighting the control inputs. Moreover, in order to enable the solver to take advantage of the vehicle dynamic model, the proposed architecture adds a continuity constraint between subsequent states as:

$$\mathcal{C} = \hat{\mathcal{C}} + \frac{1}{2} \sum_{k=1}^{N-1} \left( \|x_{k+1} - f(x_k, u_k)\|_{A_l}^2 \right) \tag{5.5}$$

where the $A_l$ matrix weights the single state component continuity, with $A_l > 0$. Intuitively, temporally adjacent states are forced to attain the system dynamics.

At this points, the proposed architecture adopts a simplifying working assumption in order to reduce the computational cost of the optimization procedure employed for minimizing the cost function stated in (5.5) and, therefore to handle non-linear constraints in an on-line implementation: the assumption consists of restricting the attention to non-linear differential flat systems [48]. As well-known, for such kind of systems it can be find a set of outputs $\zeta \in \mathbb{R}^m$, named *flat*, of the form

$$\zeta = h(x, u, \dot{u}, \cdots, u^{(r)}) \tag{5.6}$$

such that there exist two functions $\psi_k$ and $\phi_k$ for which the state and the input can be expressed in terms of flat states and a finite number of their derivatives

$$x_k = \psi_k(\zeta, \dot{\zeta}, \cdots, \zeta^{(c)}) \tag{5.7}$$

$$u_k = \phi_k(\zeta, \dot{\zeta}, \cdots, \zeta^{(c)}) \tag{5.8}$$

The formulation of the OCP in terms of *flat* states allows for a substantial dimensionality reduction of the problem, and consequently to a saving in terms of computational cost. The cost function may consequently be rewritten using the *flat* outputs as

$$\hat{\mathcal{C}}(\zeta) = \frac{1}{2} \sum_{k=0}^{N-1} \left( \| \underbrace{\psi_k(\zeta) - x_f}_{\nu_k} \|_Q^2 + \|\phi_k(\zeta)\|_R^2 \right) \tag{5.9}$$

and, by adding also the continuity constraints, the whole cost function is calculated as

$$\mathcal{C}(\zeta) = \hat{\mathcal{C}}_0(\zeta) + \frac{1}{2}\sum_{k=1}^{N-1}\Bigg(\|\nu_k(\zeta)\|_Q^2 + \|\phi_k(\zeta)\|_R^2 + \\ + \|\underbrace{\psi_{k+1}(\zeta) - f(\psi_k(\zeta), \phi_k(\zeta))}_{\gamma_k}\|_{A_l}^2\Bigg) \quad (5.10)$$

To find the solution for the cost function in (5.10), the proposed method adopts a well-established numerical method for solving optimal control problems, namely direct multiple shooting [14]. In direct multiple shooting, the whole trajectory is parametrized by finite number of *flat* outputs $\zeta \in \mathbb{R}^{Nm}$. Hence, by stacking all the error components that compose the cost function (5.10), it is possible obtain an error function $e(\zeta)$ as

$$e(\zeta) = \begin{bmatrix} \nu_0(\zeta) \\ \phi_0(\zeta) \\ \gamma_0(\zeta) \\ \vdots \\ \nu_{N-1}(\zeta) \\ \phi_{N-1}(\zeta) \\ \gamma_{N-1}(\zeta) \end{bmatrix} \quad (5.11)$$

We minimize the error function in (5.11) by adopting a least-square iterative procedure, where the trajectory is iteratively updated as

$$\zeta \leftarrow \zeta \boxplus \delta\zeta \quad (5.12)$$

where the $\boxplus$ operator performs the variable update, while taking into account the specific composition of the *flat* state [149]. The update vector $\delta\zeta$ is found by solving a linear system of the form $\mathbf{H}\delta\zeta = \mathbf{b}$ with the terms $\mathbf{H}$ and $\mathbf{b}$ given by

$$\mathbf{H} = J(\zeta)^\top \mathbf{\Omega} J(\zeta) \quad (5.13)$$

$$\mathbf{b} = J(\zeta)^\top \mathbf{\Omega} e(\zeta) \quad (5.14)$$

where $J(\zeta) = \partial e(\zeta)/\partial\zeta$. To limit the magnitude of the perturbation between iterations and thus, enforce a smoother convergence, we solve a damped linear system of the form

$$(\mathbf{H} + \lambda\mathbf{I})\,\delta\zeta = \mathbf{b}. \quad (5.15)$$

### 5.3.2. Adaptive Time-Mesh Refinement

In non-linear OCPs, the choice of the number of trajectories points $N$ is a major factor affecting the computational cost that is required to get a solution and also the accuracy of the solution itself. Hence, the goal is to effectively arrange the trajectory points along the time horizon $T$.

In the proposed method the trajectory points displacement is addressed by means of an adaptive time-mesh refinement strategy. The proposed method starts finding an initial solution by solving the OCP, as described in Sec. 5.3.1, with a coarse lattice. It looks for portions of the horizon $T$ that need to be refined: it then samples here new keypoints with a fine granularity.

In particular, since the NMPC employs a receding horizon strategy – where only the first trajectory point is actually used for the system actuation – the proposed method focuses on the resampling procedure on the initial part of the horizon.

This allows us to achieve adequate tracking performance even with a minimum amount of trajectory points $N$. As a consequence, solving more complex OCPs with longer time horizons $T$ or additional constraints (*e.g.* obstacles to avoid, objects to track, *etc.*) can be handled in real-time.

More formally, once the initial problem has been solved by employing the procedure described in Sec. 5.3.1 over a uniform time-mesh, the solution is iteratively processed. As reported in Alg. 3, at each iteration, the time-mesh refinement performs the following steps: (i) it performs a discretization error check ($\ell$. 2 and $\ell$. 12), which allows detecting where new trajectory points are required; (ii) it then adds new points by interpolating them between the adjacent ones ($\ell$. 7-8); (iii) finally, it transcribes the sub-problem into an OCP and solves it ($\ell$. 10-11). In the following, these steps will be discussed in more detail.

### 5.3.3. Discretization Error Check

In order to proceed with the time-mesh refinement strategy, refinement and stopping criteria have to be defined. The proposed method considers as main refinement criterion of the discretization error between the *flat* variables. The discretization error, at each lattice point, is computed as the difference between the current state and a higher-order approximation of the solution trough the non-linear time-step rule of the dynamics of (5.2). More specifically, starting from the $\zeta_{k-1}$ node, the proposed method performs a finer integration of the dynamics with respect to the one employed during the OCP solution, obtaining $\hat{\zeta}_k$. Hence, the discretization error is computed as

$$\epsilon_{\zeta_k} = \|\hat{\zeta}_k - \zeta_k\| \tag{5.16}$$

where $\|\cdot\|$ stands as the squared norm of two vectors. As a stopping criterion, the proposed method considers a threshold on the discretization error $\epsilon_{\zeta_k}$.

#### Trajectory Points Interpolation

Once the discretization error has been obtained, the time-mesh refinement proceeds by adding new trajectory nodes where required. In order to obtain a smooth interpolation that preserves the *flat* states differentiability, the proposed architecture uses the cubic Hermite interpolation. More formally, let $\zeta_i$ and $\zeta_j$ be two adjacent *flat* states in the lattice, the interpolated *flat* state in the unit time interval $[0, 1]$ is computed as:

$$\zeta_k = \zeta_i(2t^3 - 3t^2 + 1) + \zeta_i^{'}(t^3 - 2t^2 + t) + \zeta_j(3t^2 - 2t^3) + \zeta_j^{'}(t^3 - t^2) \tag{5.17}$$

where $t \in [0, 1]$ is the interpolation point, and $\zeta^{'}$ denotes the first-order *flat* variable derivative.

### Local Optimization Procedure

The refinement procedure adds more node points to the initial portion of the trajectory where the discretization error is higher, so to obtain an improved solution.

As a consequence, the computational time increases. To avoid this the issue, when progressively going from a coarse mesh to a refined one, the error function in (5.11) is scaled only to the initial part of the horizon. More specifically, let $N_{tm} = N_i + N_{add}$ be the number of initial trajectory points that are going to be refined, where $N_i < N$ is a user-defined parameter representing the number of points that belong to an initial section of the trajectory, and $N_{add}$ is the number of points added at each iteration of the refinement algorithm. At each iteration, the refinement procedure transcribes these points in an OCP and re-optimizes only the mesh sub-intervals that belong to $\zeta_{0:N_{tm}} = \{\zeta_0, \cdots, \zeta_{N_{tm}}\}$, while keeping unchanged the remaining part of the trajectory.

```
Data: Cost function $\mathcal{C}$, dynamics $f(x(k), u(k))$, OCP solution $\zeta_{0:N} = \{\zeta_0, \cdots, \zeta_N\}$,
      trajectory points to refine $N_{tm}$.
Result: Refined trajectory $\zeta_{0:N+N_{add}}$
scale the OCP problem: $\zeta_{0:N_{tm}} = \{\zeta_0, \cdots, \zeta_{N_{tm}}\}$;
discretization error computation for each lattice point $\epsilon_{\zeta_{1:N_{tm}}}$;
$iter = 0$;
while $iter < max_{iter}$ do
    foreach $\zeta_i \in \zeta_{1:N_{tm}}$ do
        if $\epsilon_\zeta > err_{trs}$ then
            $\zeta_k = Interpolate(\zeta_i, \zeta_{i+1})$;
            Add $\zeta_k$ to $\zeta_{0:N_{tm}}$;
            $N_{tm} \leftarrow N_{tm} + 1$;
        transcribe the scaled OCP;
        apply the Least-Square solver;
        estimate the discretization error $\epsilon_{\zeta_{1:N_{tm}}}$;
        $iter \leftarrow iter + 1$;
    end
end
```

**Algorithm 3:** Time-Mesh Refinement Algorithm.

**Table 5.4.** Errors and Control Inputs Statistics for the Pose Regulation Experiment

| $N$ | $TMref$ | $Runtime$ $[ms]$ | $err_{trans}$ $[m]$ | $err_{rot}$ $[rad]$ | $Roll_{ref}$ $[rad]$ | $Pitch_{ref}$ $[rad]$ | $YawRate$ $[rad/s]$ | $Thrust$ $[Nm]$ |
|---|---|---|---|---|---|---|---|---|
| 100 | | 42 | 0.0537 | 0.0435 | 0.0196 | 0.0196 | 0.3305 | 15.2450 |
| 50 | | 10 | 0.0613 | 0.0543 | 0.0184 | 0.0210 | 0.2906 | 15.1695 |
| 50 | ✓ | 10.2 | 0.0608 | 0.0521 | 0.0191 | 0.0213 | 0.2863 | 15.2237 |
| 20 | | 1 | 0.0724 | 0.0598 | 0.0195 | 0.0259 | 0.3146 | 15.1554 |
| 20 | ✓ | 1.2 | 0.0703 | 0.0553 | 0.0193 | 0.0224 | 0.2929 | 15.2491 |
| 10 | | 0.5 | 0.1096 | 0.0735 | 0.0207 | 0.0257 | 0.2964 | 15.1352 |
| 10 | ✓ | 0.7 | 0.0823 | 0.0642 | 0.0197 | 0.0243 | 0.3321 | 15.1934 |
| 5 | | 0.2 | fail | fail | fail | fail | fail | fail |
| 5 | ✓ | 0.4 | 0.1032 | 0.0667 | 0.0219 | 0.0234 | 0.3219 | 15.2154 |

### 5.3.4. Experimental Evaluation

The proposed approach is tested in a simulated environment by using the RotorS simulator [53] and a multirotor model. The mapping between the high-level control input and the propeller velocities are done by a low-level PD controller that aims to resemble the low-level controller that runs on a real multirotor.

The evaluation presented here is designed to support the claims made in the introduction. I performed two kind of experiments, namely *pose regulation* and *trajectory tracking.*

I provide a direct comparison between a standard NMPC implementation and the one proposed in this section, *i.e.* by using a time-mesh refinement strategy.

The OCP is formulated by composing each flat state of the UAV simulated model as follows:

$$\zeta = (p_1, p_2, p_3, \gamma) \tag{5.18}$$

where $p_i$ is the translation in the world coordinate reference system along the $i^{th}$ axis, and $\gamma$ represents the *yaw* angle. For more detail about the dynamical model and the flat model, please refer to Appendix A.

#### Pose Regulation

The following pose regulation experiment is designed to prove the accuracy and the robustness of the proposed approach. In all the regulation experiments the desired state is set to:

$$t_f = 2, \quad \zeta = (2, 2, 1, 1.57), \quad \dot{\zeta} = (0, 0, 0, 0).$$

and the time-mesh refinement parameters to be $err_{trs} = 10^{-5}, \quad max_{iter} = 2.$.

The pose regulation tasks are performed while varying the bins number $N$. The goal is to show how the control accuracy deteriorates while going from a finer lattice to a coarser one. To this end, it starts with $N = 200$, which is used as a reference for the other bin setups. Then, the proposed method decreases the bins amount up to $N = 5$ measuring the translational and rotational RMSE with respect to the reference trajectory, for both the standard NMPC case and the proposed approach. Tab. 5.4 reports the results of this comparison, along with the computational time required for solving the OCP and the control inputs average.

The advantages of using a time-mesh refinement strategy are twofold. From one side, when using a coarse lattice, as in the case of $N = 20$ and $N = 10$, the the refined solution provides lower errors, with a negligible increment of computational time. On the other hand, it intrinsically increases the robustness by adaptively adding bins in the trajectory where needed, thus avoiding failures such as the one registered with $N = 5$ in the standard NMPC formulation. Fig. 5.8 directly compares the convergence with the reference trajectory ($N = 200$) and the ones recorded while using $N = 20$ and $N = 5$ with the time-mesh refinement.

#### Trajectory Tracking

To prove the effectiveness of the proposed time-mesh refinement in a tracking scenario, the UAV platform is commanded to track a challenging Lemniscate trajectory, defined pb_control/imgs/gnomic_pose_reg_new.png

**Figure 5.8.** Pose Error

by the following expression:

$$
\begin{aligned}
p_1(t) &= 2sin(t/2)\\
p_2(t) &= 2sin(t/2)cos(t/2)\\
p_3(t) &= sin(t+5)/3\\
\gamma(t) &= sin(t/8)
\end{aligned}
$$

$t_f$ is set to $0.5s$ and $N$ is set to 5 for both the standard and the refined setup, while $err_{trs}$ and $max_{iter}$ are set with the same values used in the pose regulation section 5.3.4.Fig. 5.9 and Fig. 5.10 report the results of the tracking experiments. As expected, the use of a time-meshrefinement strategy allows for a more accurate tracking since the OCP problem is solved over an adapted lattice.

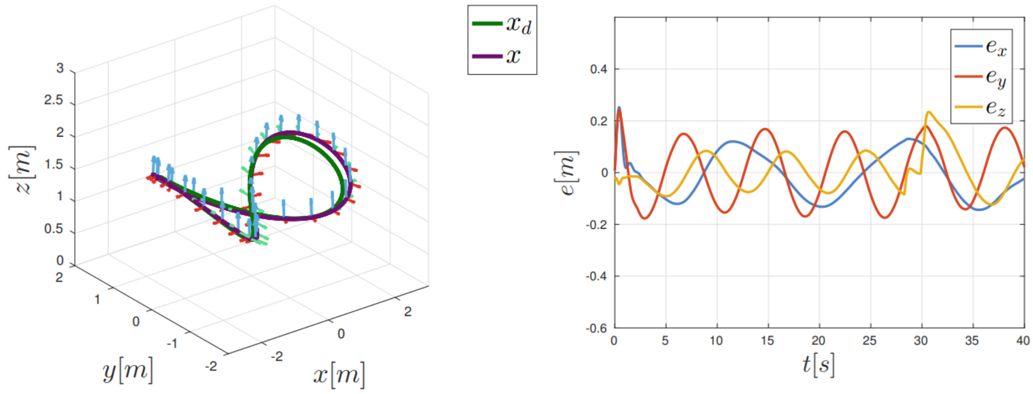


**Figure 5.9.** Lemniscate trajectory tracked with standard approach.

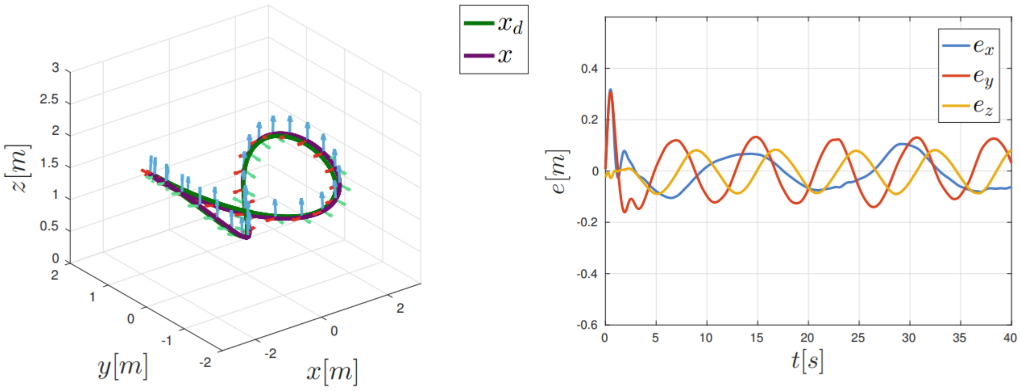


**Figure 5.10.** Lemniscate trajectory tracked with time-mesh refinement.

### Runtime

We recorded the time needed to solve the NMPC problem, as described in Sec. 5.3.1. We performed all the presented experiments on a laptop computer equipped with an i7-5700HQ CPU with 2.70 GHz. Our software runs on a single core and in a single thread.

To reduce the noise in the measurements, we collect the computational times for time horizons between $500ms$ and $4000ms$, using bins of $200ms$. For each configuration of the solver, we compute the average computational time over 4000 planned trajectories with a UAV, and report these values in Fig. 5.11.

As shown in Fig. 5.11, while increasing the time horizon, and consequently the number of bins, the computational time grows almost linearly. The computational overhead of the time-mesh refinement, in this sense, does not affect such a cost, by adding a constant time to the total computation.

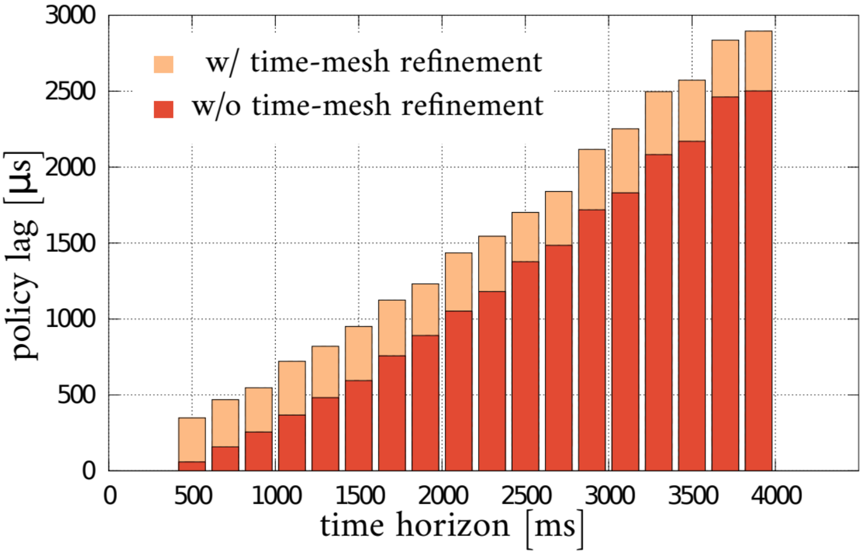


**Figure 5.11.** Average runtime over 4000 planned trajectories with horizon up to $4s$, with an UAV model.

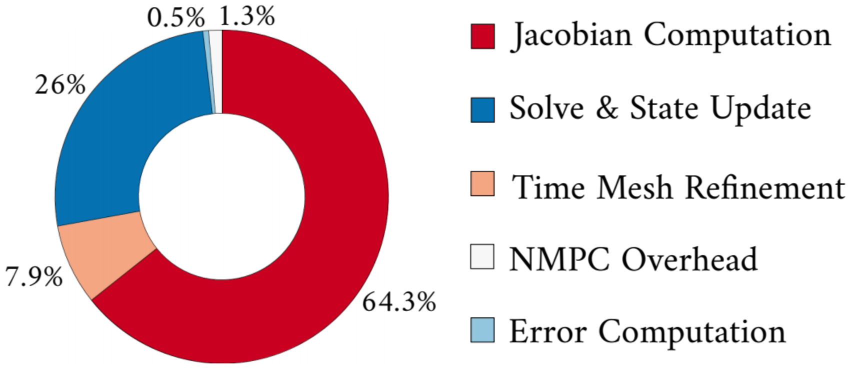


**Figure 5.12.** Runtime percentage of the NMPC algorithm with time-mesh refinement in case of $t_f = 2s$ with bins each $200ms$.

Fig. 5.12 shows the runtime percentage in case of $t_f = 2s$ with bins each $200ms$, where the time-mesh refinement slice includes the computational time involved in all the different mesh refinement steps. Note that the time-mesh refinement portion has constant time consumption. Thus its percentage value is indicative of this particular setup only. It is also noteworthy to highlight how most of the runtime is absorbed by the Jacobian calculation, being computed in a fully numerical manner.

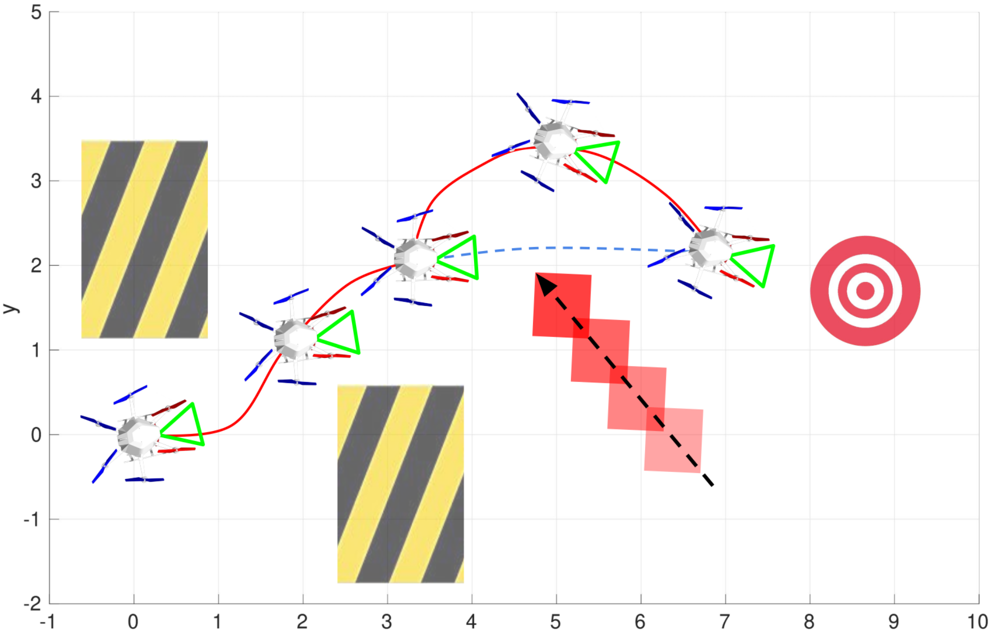


**Figure 5.13.** An example of an application for the proposed system: a UAV is asked to reach the desired state while constantly framing a specific target (the red circular target in the picture). The environment is populated with static obstacles (black and yellow striped objects in the picture) that should be avoided. Dynamic obstacles (e.g., other agents, depicted with the red moving box) may suddenly appear and block the planned trajectory (i.e., the blue dashed line). With the proposed method, the UAV reacts to the detected object by steering along a new safe trajectory (i.e., the red line).

## 5.4. Joint Vision-Based Navigation, Control and Obstacle Avoidance for UAVs in Dynamic Environments

In this section, I take a small step forward in this direction by proposing an optimal controller that takes into account in a joint manner the perception, the dynamic, and the avoidance constraints (Fig. 5.13). The proposed system models vehicle dynamics, perception targets and obstacles in terms of Non-Linear Model Predictive Controller (NMPC) constraints: dynamics are accounted by providing a non-linear dynamic model of the vehicle, while targets are modeled by targets visibility constraints in the camera image plane and obstacles by repulsive ellipsoidal areas, respectively. The proposed system also allows incorporating estimation uncertainties and obstacle velocities in the ellipsoids, allowing it to deal also with dynamic obstacles.

The entire problem is then transcribed into an Optimal Control Problem (OCP) and solved in a receding horizon fashion: at each control loop, the NMPC provides a feasible solution to the OCP and only the first input of the provided optimal trajectory is actually applied to control the robot. By leveraging state-of-the-art numerical optimization, the OCP is solved in a few milliseconds making it possible to control the vehicle in real-time and to guarantee enough reactivity to re-plan the trajectory when new obstacles are detected.

Moreover, the proposed approach does not depend on a specific application and can potentially provide benefits to a large variety of applications, such as vision-based navigation, target tracking, and visual servoing.

### 5.4.1. Problem Formulation

The goal of the proposed approach is to generate an optimal trajectory that takes into account perception and action constraints of a small UAV and, at the same time, allows to safely fly through the environment by avoiding all the obstacles that can lie along the planned trajectory.

The need for coupling action and perception constraints derives from different factors. On the one hand, there are the vision-based navigation limits where to guarantee an accurate and robust state estimation, it is necessary to extract meaningful information from the image. On the other hand, in some specific cases (e.g. Visual Servoing) the feedback information used to control the vehicle is extracted from a vision sensor, thus the vision target should be kept in the camera image plane. Similarly, taking into account the surrounding obstacles is important to ensure a safe flight in cluttered environments.

Considering all those factors together allows to fully leverage the agility of UAVs and to have a fully autonomous flight. Therefore, it is essential to jointly consider all these constraints.

Let $l$ be the state vector of the target object (e.g., the 3D point representing the center of mass of a target object), while let $x$ and $u$ be the state and the input vectors of a robot, respectively. Furthermore, let $o$ the state vector of the obstacles to avoid. Let assume the robot's dynamic can be modeled by a general, non linear, differential equations system $\dot{x} = f(x, u)$. Finally, given some flight objectives, an action cost can be defined as $c_a(x_t, u_t)$, a perception cost $c_p(x_t, l_t, u_t)$, a navigation cost $c_n(x_t, u_t)$, and an avoidance cost $c_o(x_t, o_t, u_t)$

Thus, the coupling of action, perception, and avoidance can be formulated as an optimization problem with cost function:

$$\begin{aligned} J = c_f(x_{tf}) + \int_{t_0}^{t_f} c_a(x_t, u_t) + c_p(x_t, l_t, u_t) + c_n(x_t, u_t) + c_o(x_t, o_t, u_t) dt \\ \textit{subject to} : \dot{x} = f(x, u) \\ h(x_t, l_t, o_t, u_t) \leq 0 \end{aligned} \tag{5.19}$$

where $h(x_t, l_t, o_t, u_t)$ stands for the set of inequality constraints to satisfy along the trajectory, $c_f(x_{tf})$ stands for the cost on the final state, and $t_f - t_0$ represents the time horizon in which one wants to find the solution. In the following, a description of all the cost function components is reported.

#### Quadrotor Dynamics

In the proposed approach, five reference frames will be considered: (i) the world reference frame $W$; (ii) the body reference frame $B$ of the UAV; (iii) the camera reference frame $C$; (iv) the ith obstacle reference frame $Oi$ and the target reference

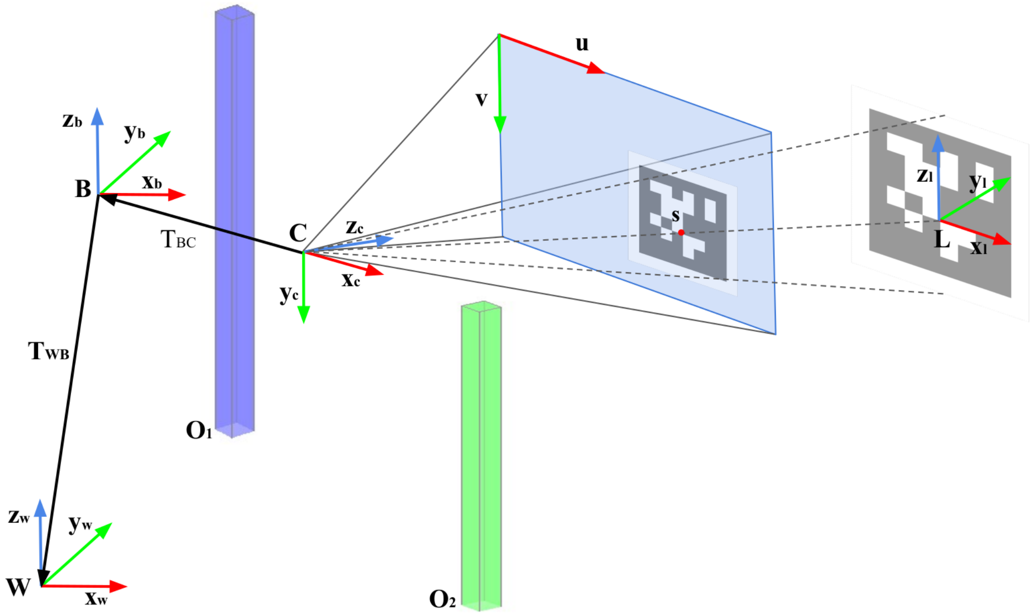


**Figure 5.14.** Overview of the reference systems used in this work: the world frame $W$, the body frame $B$, the camera frame $C$, the landmark and obstacles frames $L$ and $O_i$. $T_{WC}$ and $T_{BC}$ represent the body pose in the world frame $W$ and the transformation between the body and the camera frames $C$, respectively. Finally, $s$ is the landmark reprojection onto the camera image plane.

frame $L$. An overview of the reference systems is illustrated in Fig. 5.14.
To represent a vector or a transformation matrix, the proposed formulation makes use of a prefix that indicates the reference frames in which the quantity is expressed. For example, $x_{WB}$ denotes the position vector of the body $B$ frame with respect to the world frame $W$, expressed in the world frame.
According to this representation, let $p_{WB} = (p_x, p_y, p_z)^T$ and $r_{WB} = (\phi, \theta, \psi)^T$ be the position and the orientation of the body frame with respect to the world frame, expressed in the world frame, respectively. Additionally, let $V_{WB} = (v_x, v_y, v_z)^T$ be the velocity of the body, expressed in the world frame. Finally, let $u = (T, \phi_{cmd}, \theta_{cmd}, \dot{\psi}_{cmd})^T$ be the input vector, where $T = (0, 0, t)^T$ is the thrust vector normalized by the mass of the vehicle, and $\phi_{cmd}$, $\theta_{cmd}$, $\dot{\psi}_{cmd}$ are the roll, pitch, and yaw rate commands, respectively. Thus, the quadrotor dynamic model $f(x, u)$ can be expressed as:

$$\begin{aligned} v_{WB} &= \dot{p}_{WB} \\ \dot{v}_{WB} &= g_W + R_{WB} T \\ \dot{\phi} &= \frac{1}{\tau_\phi}(k_\phi \phi_{cmd} - \phi) \\ \dot{\theta} &= \frac{1}{\tau_\theta}(k_\theta \theta_{cmd} - \theta) \\ \dot{\psi} &= \dot{\psi}_{cmd} \end{aligned} \tag{5.20}$$

where $R_{WB}$ is the rotation matrix that maps the mass-normalized thrust vector $T$ in the world frame, and $g_W = (0, 0, -g)^T$ is the gravity vector. For the attitude dynamics, the proposed formulation makes use of a low-level controller that maps the high-level attitude control inputs into propellers' velocity. The $\tau_i$ and $k_i$ parameters are obtained through an identification procedure [80].

### Perception Objectives

Let $p_{WL} = (l_x, l_y, l_z)^T$ the 3D position of the target of interest in the world frame $W$. It is assumed that the UAV to be equipped with a camera having extrinsic parameters described by a constant rigid body transformation $T_{BC} = (p_{BC}, R_{BC})$, where $p_{BC}$ and $R_{BC}$ are the position and the orientation, expressed as a rotation matrix, of the camera frame $C$ with respect to $B$. The target 3D position in the camera frame $C$ is given by:

$$p_{CL} = (R_{WB} R_{BC})^T (p_{WL} - (R_{WB} p_{BC} + p_{WB})) \tag{5.21}$$

The 3D point $p_{CL}$ is then projected onto the image plane coordinates $s = (u, v)^T$ according to the standard pinhole model:

$$u = f_x \frac{p_{CLx}}{p_{CLz}}, \qquad v = f_y \frac{p_{CLy}}{p_{CLz}} \tag{5.22}$$

where $f_x$ and $f_y$ stand for the focal lenghts of the camera. It is noteworthy to highlight that the optical centers parameters $c_x$ and $c_y$ are not being used in projecting the target, since it is convenient to refer it with respect to the center of the image plane. To ensure a robust perception, the projection $s$ of a target of interest should be kept as close as possible to the center of the camera image plane. Therefore the formulation of the perception cost $c_p(x_t, l_t, u_t)$ will be:

$$c_p(x_t, l_t, u_t) = sHs^T, \qquad H = h \begin{bmatrix} \frac{1}{f_c} & 0 \\ 0 & \frac{1}{f_r} \end{bmatrix} \tag{5.23}$$

where $f_c$ and $f_r$ represent the number of columns and rows in the camera image, while $h$ is a weighting factor. With this choice, the reprojection error of $s$ is more penalized in the shorter image axis. For instance, if the camera streams a 16:9 image, the optimal solution will care more to keep $s$ closer to the center of the image along the $v$-axis.

### Avoidance and Navigation Objectives

Let $o_{WO_i} = (o_{x_i}, o_{y_i}, o_{z_i})^T$ be the 3D position of the *i-th* obstacle in the world frame $W$. To enable the UAV to safely flight, the trajectory has to constantly keep the aerial vehicle at a safe distance from all the surrounding obstacles. Moreover, the cost function(5.19) has to take into account objects with different shapes and sizes. Thus, the avoidance cost $c_o(x_t, o_t, u_t)$ is formulated as

$$\begin{gathered} \sum_{i=1}^{N_o} \frac{1}{d_i W_i d_i^T}, \qquad d_i = P_{WB} - o_{WO_i} \\ W_i = diag(w_{xi}, w_{yi}, w_{zi}), \;\; w_i = f(\gamma_i, \epsilon_i, v_i) \end{gathered} \tag{5.24}$$

where $N_o$ is the number of the obstacles and $W_i$ is the i-th weighting matrix. The latter weighs the distances along the 3 main axes, creating an ellipsoidal bounding box. More specifically, each component $w_i$ embeds the obstacle's size $\gamma$, velocity $v$, and estimation uncertainties $\epsilon$ (see Fig. 5.15). Among the others, this formulation allows to set more conservative bounding boxes according to the obstacle detection accuracy. Moreover, to guarantee a robust collision avoidance, the minimum acceptable distance is formulated as an additional inequality $h(x_t, o_t, u_t)$ constraint:

$$\sum_{i=1}^{N_o} d_i W_i d_i^T >= d_{min,i} \tag{5.25}$$

where $d_{min,i}$ represents the minimum acceptable distance for the i-th obstacle.

### Action Objectives

The action objectives act to penalize the amount of control inputs used to steer the vehicle. Therefore, the action cost $c_a(x_t, u_t)$ is formulated as:

$$c_a(x_t, u_t) = \bar{u} R \bar{u}^T, \qquad \bar{u} = u - u_{ref} \tag{5.26}$$

where $R$ is a weighting matrix, and $u_{ref}$ represents the reference control input vector (e.g. the control commands to keep the aerial vehicle in hovering). Moreover, to constrain the control commands to be bounded inside the input range allowed by the real system, the proposed method also makes use of an additional inequality constraint $h(x_t, u_t)$:

$$u_{lb} <= u <= u_{ub} \tag{5.27}$$

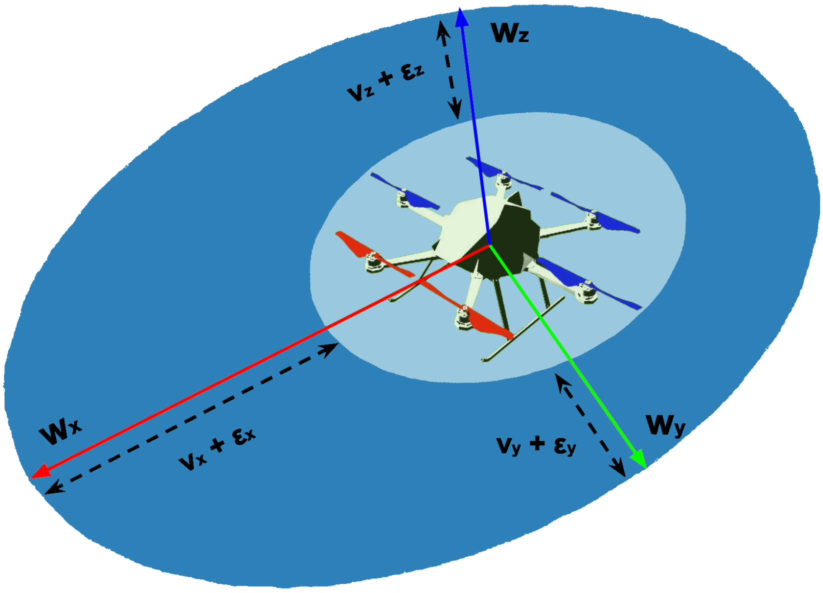


**Figure 5.15.** Ellipsoidal bounding box concept overview: the light blue area bounds the obstacle physical dimensions, while the blue area embeds the uncertainties $\epsilon$ in the obstacle pose estimation and velocity $v$. The blue area is stretched along the x axis direction to take into account the object estimated velocity.

The remaining costs $c_n(x_t, u_t)$ and $c_f(x_{tf})$ penalize the distance from the goal pose, and are formulated as:

$$\begin{aligned} c_n(x_t, u_t) &= \bar{x} Q \bar{x}^T, \quad & \bar{x} &= x - x_{ref} \\ c_f(x_{tf}) &= \bar{x}_{tf} Q_N \bar{x}_{tf}^T, \quad & \bar{x}_{tf} &= x_{tf} - x_{ref} \end{aligned} \tag{5.28}$$

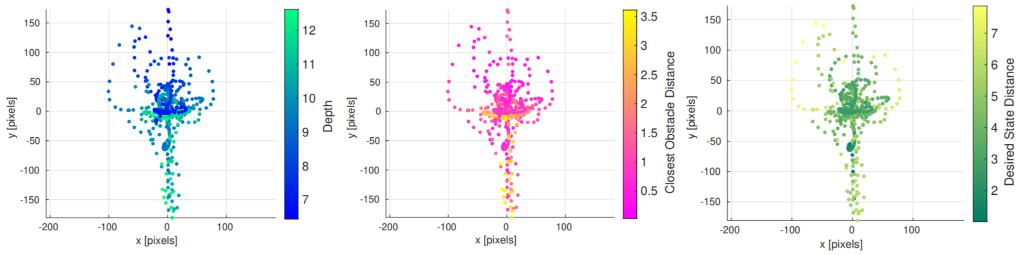


**Figure 5.16.** Target reprojection error for 10 hover-to-hover flights. The three different color-maps represent the depth of the point of interest, the distance from the closest static obstacle, and the distance between the current pose and the desired pose, respectively.

### 5.4.2. Non-Linear Model Predictive Control

The cost function given in(5.19) results in a non-linear optimal control problem. To find a time-varying control law that minimizes it, the proposed method makes use of a Non-Linear Model Predictive Controller, where the cost function(5.19) is firstly approximated by a Sequential Quadratic Program (SQP), and then iteratively solved by a standard Quadratic Programming (QP) solver.

The whole system works in a receding horizon fashion, meaning that at each new measurement, the NMPC provides a feasible solution and only the first control input of the provided trajectory is actually applied to control the robot.

To achieve that, the system dynamics is discretized with a fixed time step $dt$ over a time horizon $T_H$ into a set of state vectors $x_{0:N} = \{x_0, x_1, \ldots, x_N\}$ and a set of inputs controls $u_{0:N} = \{u_0, u_1, \ldots, u_{N-1}\}$, where $N = T_H/dt$. Additionally,the state, the final state, and input cost matrices are defined as $Q$, $Q_N$, and $R$, respectively. The final cost function will be:

$$J = \bar{x}_{tf} Q_N \bar{x}_{tf}^T + \sum_{i=0}^{N-1} (c_n + c_a + c_p + c_o) \tag{5.29}$$

where $\bar{x}$ represents the difference with respect to the state reference values, while $c_n$, $c_a$, $c_p$ and $c_o$ refer to the navigation, action, perception and avoidance objectives introduced in the previous section.

For NMPC to be effective, the optimization should be performed in real-time. In this regard, an approximation of each optimal solution is computed by executing only a few iterations at each control loop. Moreover, the previous approximated solution is kept and used as the initialization for the next optimization.

### 5.4.3. Experiments

The evaluation presented here is designed to support the claims made in the introduction. Two kinds of experiments are performed, namely *hover-to-hover flight with static obstacles* and *hover-to-hover flight with dynamic obstacles.* To demonstrate the real-time capabilities of the proposed approach a computational time analysis will also be presented. In all the reported results, the multirotor is asked to fly multiple times by randomly changing the obstacles setup and the goal state.

### 5.4.4. Simulation Setup

The proposed approach is tested in a simulated environment by using the RotorS UAV simulator [53] and an AscTec Firefly multirotor. The non-linear control problem is set up with the ACADO toolbox and used the qpOASES solver [45]. By using the ACADO code generation tool, the problem is then exported in a highly efficient C code that is integrated within a ROS (Robot Operating System) node. The discretization step to be $dt$ is set as 0.2 $s$ with a time horizon $T_H = 2$ $s$. To guarantee enough agility to the vehicle, the control loop is run at 100 $Hz$. The mapping between the optimal control inputs and the propeller velocities is done by a low-level PD controller that aims to resemble the low-level controller that runs on a real multirotor. To make the simulation more realistic, a further white noise with

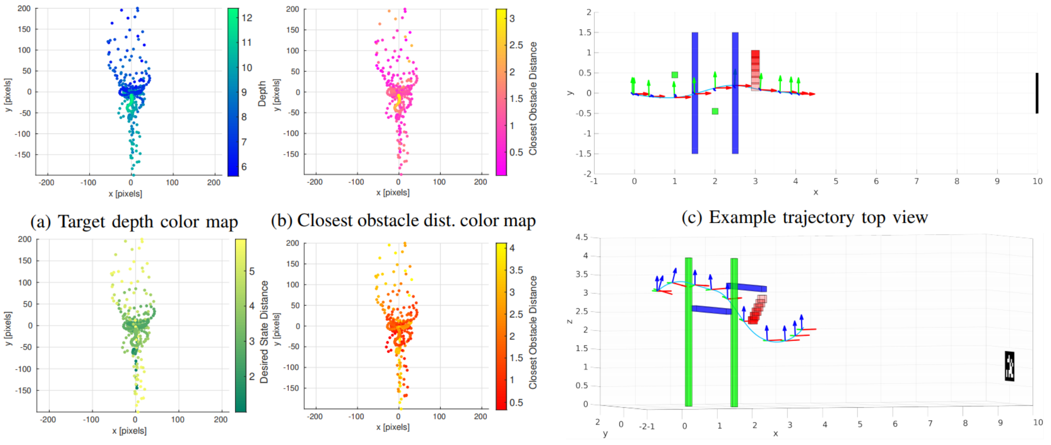
| Dynamic Obstacle | delay [s] | failure rate [%] | Avg. pixel error | Max pixel error | Torque [N] | $T_{cmd}[g]$ | $\sigma[cm]$ |
|---|---|---|---|---|---|---|---|
| | | 0.2 | 53 | 98 | 0.029 | 1.35 | 2 |
| ✓ | .2 | 0.4 | 79 | 129 | 0.030 | 1.39 | 2 |
| ✓ | .4 | 2.0 | 84 | 142 | 0.031 | 1.41 | 2 |
| ✓ | .6 | 10.4 | 90 | 151 | 0.040 | 1.47 | 2 |
| ✓ | .8 | 18.9 | 92 | 155 | 0.040 | 1.50 | 2 |
| ✓ | 1 | 24.8 | 98 | 157 | 0.041 | 1.52 | 2 |

**Table 5.5.** Trajectory statistics comparison across different simulation setups.

| Dyn. obst. delay | Dyn. obst. velocity [s] | failure rate [%] | Avg. pix. error | Max pix. error | Torque [N] | $T_{cmd}[g]$ | $\sigma[cm]$ |
|---|---|---|---|---|---|---|---|
| .2 | .2 | 1.9 | 81 | 101 | 0.031 | 1.40 | 2 |
| .2 | .4 | 3.5 | 89 | 111 | 0.031 | 1.41 | 2 |
| .2 | .6 | 4.8 | 90 | 113 | 0.034 | 1.44 | 2 |
| .4 | .2 | 3.9 | 93 | 140 | 0.032 | 1.45 | 2 |
| .4 | .4 | 7.4 | 94 | 147 | 0.034 | 1.44 | 2 |
| .4 | .6 | 11.9 | 107 | 151 | 0.035 | 1.49 | 2 |

**Table 5.6.** Trajectory statistics comparison across different dynamic obstacle spawning setups.

standard deviation $\sigma$ is also added on the 3D positions of the detected obstacles. The code developed in this work is publicly available as open-source software.



(a) Target depth color map (b) Closest obstacle dist. color map (c) Example trajectory top view

(d) Desired pose dist. color map (e) Dyn. obstacle dist. color map (f) Example trajectory lateral view

**Figure 5.17.** Reprojection error in the hover-to-hover flight with dynamic obstacles (left-side), and an example of the planned trajectory (right-side). The moving, dynamic, obstacle is represented by the red cube where the varying color intensity represents its motion over time.

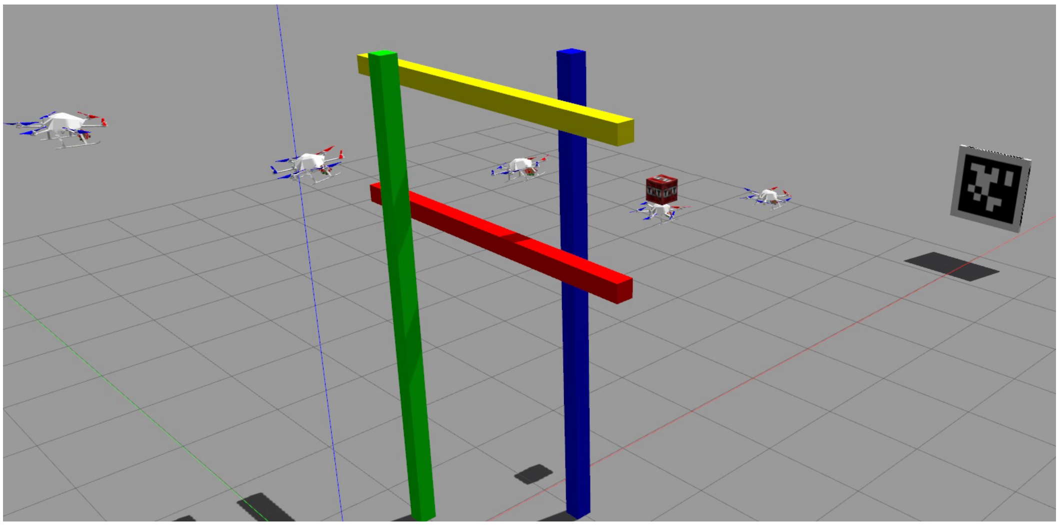


**Figure 5.18.** Example of a trajectory generated in the simulated scenario. The depicted UAVs represent different poses assumed by the aerial vehicle across the time horizon. The colored objects represent the static obstacle, while the red box the dynamic one.

### Hover-To-Hover Flight with Static Obstacles

The presented experiment shows the capabilities in *hover-to-hover* flight maneuvers with static obstacles. More specifically, the UAV is commanded to reach a set of $M$ randomly generated desired states $X_{des} = (x_{ref,0}, x_{ref,1}, \dots, x_{ref,M})$. Unlike standard controllers, the proposed approach will generate, at each time step, control inputs that will steer the vehicle towards the goal state while avoiding obstacles and keeping the target in the image plane. Fig. 5.16 depicts the reprojection error of the point of interest and its correlation with (i) the depth of the point of interest, (ii) the distance from the closest obstacle, and (iii) the distance from the desired state. The largest reprojection errors occur when the UAV is farther from the desired state, or when the UAV has to fly closer to the obstacles. In these cases, the reprojection error is slightly higher since the UAV has to perform more aggressive maneuvers. However, as reported in Tab. 5.5, the UAV keeps a success rate of almost 100% while keeping a low usage of control inputs.

### Hover-To-Hover Flight with Dynamic Obstacles

This experiment shows the capabilities to handle more challenging flight situations, such as the flight in the presence of dynamic, unmodeled obstacles (see Fig. 5.18 for an example). To demonstrate the performance in such a scenario, a dynamic obstacle is randomly spawned along the planned trajectory. Thus, to successfully reach the desired goal, the UAV has to quickly re-plan a safe trajectory (see Fig. 5.17(c) and Fig. 5.17(f) for an example). Moreover, to make experiments with an increasing level of difficulty, the dynamic obstacle is spawned with a random delay and with a random non-zero velocity. The random delay simulates the delay in detecting the

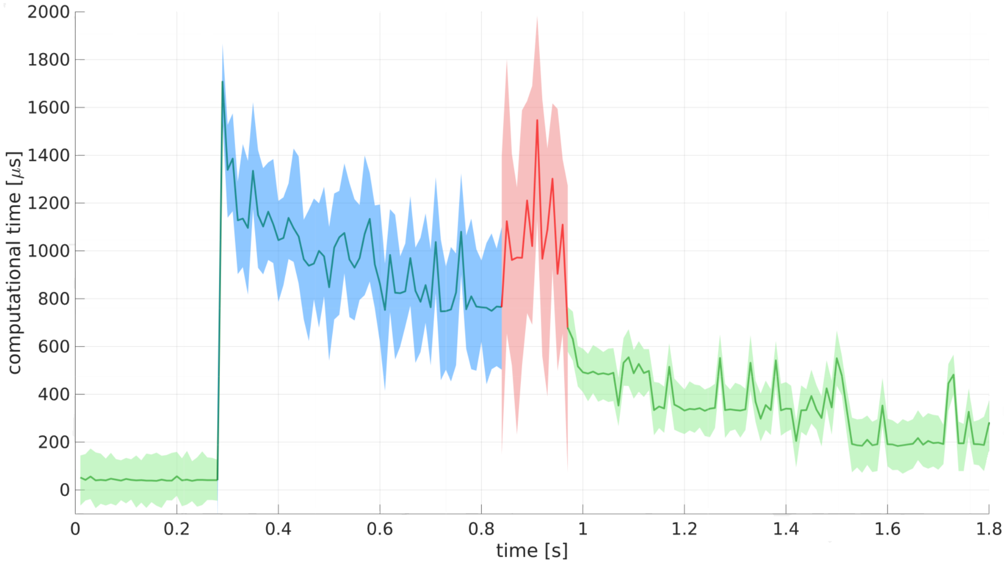


**Figure 5.19.** Average computational time plot across the planning phases: the (i) planning phase in blue, (ii) the steady-planning phase in green, and (iii) the emergency re-planning phase in red. The shaded areas represent the variance of the average computational time.

obstacle or the possibility that the obstacle appears after the vehicle has already planned the trajectory.

The reprojection error follows a similar trend as the previous set of experiments (see Fig. 5.17), being higher when the vehicle is closer to static obstacles of farther from the desired state. Fig. 5.17(d) reports the evolution of the reprojection error colored according to the distance from the dynamic obstacle. Since the latter is spawned close to the planned trajectory, the UAV has to perform an aggressive maneuver to keep a safe distance from it. This usually leads tin having a smaller reprojection error when the dynamic obstacle is close (i.e. the object is spawned while the UAV was on the optimal trajectory), and a bigger error when the obstacle is farther (i.e. the drone reacted with an aggressive maneuver to avoid it). Tab. 5.5 and Tab. 5.6 report some trajectory statistics. The proposed method keeps a success rate above the 75% in almost all the conditions, even in the presence of large delays. It is also noteworthy to highlight how the delay turns out to be more critical than the dynamic obstacle's velocity. The latter, indeed, makes the re-planning more challenging only in specific circumstances (e.g. when the object moves towards the UAV). Finally, the capability to avoid obstacles comes with a performance trade-off. The greater the difficulty, the greater the use of control inputs. This is especially true when the UAV has to avoid dynamic obstacles with a large delay since it involves making expensive control maneuvers.

### Computational Time

To meet the control loop real-time constraints, the NMPC computational cost should be as low as possible. Moreover, the computational cost is not constant and might vary according to the similarity between the initial trajectory and the optimal one. In this regards, the presented computational time evaluation distinguishes among three main flight phases:

1. the planning phase, which occurs when the UAV has to plan a trajectory to reach $x_{des}$;
2. the steady-planning phase, which occurs when the UAV is already moving toward $x_{des}$;
3. the emergency re-planning, which occurs when a dynamic obstacle suddenly appears along the optimal trajectory.

Fig. 5.19 reports the average computational costs for all those flight phases. The average computational cost is constantly lower than  0.01 seconds, meeting the control loop frequency constraints. The steady-planning phase turns out to be the cheapest one. Indeed, since the control loop runs 100 times per second, the neighbor trajectories are quite similar. Conversely, the emergency re-planning phase is the most expensive and variable, since the trajectory to re-plan is often quite different from the previous one, depending on where the dynamic obstacle is spawned.

## 5.5. Summary of Results and Lessons Learned

This chapter focuses on the problem of controlling a robot by explicitly considering some perception and avoidance constraints. The overall study proposes three different perspectives to address this task, namely "*using a cascade of optimal controllers*", "*leveraging a least-square solver to exploit the differential flatness property*" and "*incorporating avoidance and perception constraints in a single optimal controller*". Besides their generality, the described approaches may provide several benefits to monitoring and controlling operations in farming activities, particularly where the repetitiveness and the perceptual aliasing might mislead the onboard navigation system of the robot. The proposed control architectures start from the basic concept of visual servoing, which has been widely investigated in the literature and enforce it by solving the whole problem in a minimization manner.

More specifically, in the first proposed solution, the perception constraint is formulated as the error between the target object projection onto the camera image plane and the center of the camera view. Thus, such a constraint is incorporated into a Non-Linear Model Predictive Control that returns an optimal trajectory to follow. Despite the low running frequency, the output trajectory satisfies both the perception and physical constraints of the vehicle, leading to better performance. To reduce the computational burden, a least-square solver and the differential flatness property of underactuated vehicles are leveraged. More specifically, I formulate a multi-resolution online smoother to solve, in real-time on a dynamically complex platform, a non-linear optimal control problem. With this approach, I provide a more refined solution in the first portion of trajectory, *i.e.* the part that is actually used to control the platform, at a reduced computational cost. Finally, in the last of the proposed contributions, I also considered obstacles in the control stack. The obstacles are formulated as ellipsoids where their dimensions depend on their physical size, velocity, and detection uncertainty. Those avoidance constraints are incorporated in a real-time Non-Linear Model Predictive Control implementation together with the perception constraint. Through the analysis of the performed trajectories, I show how the system allows for agile and effective maneuvers while meeting all the constraint requirements. The same approaches can be also used in more simple contexts, such as planning.

# Chapter 6

# Conclusion and Discussion

The goal of this thesis is to assess to what extent an enhanced perception and an *ad-hoc* environmental modeling can be exploited to improve the level of autonomy and the effectiveness of robotic platforms in farming activities. This thesis is motivated by the growing adoption of sensing technologies and automation for monitoring and on-field intervention in agriculture: on the one hand, as described in chapter 1, robotics is commonly perceived as an enabling technology to further increase the window of opportunity for intervention and farm field monitoring. On the other hand, at the current status, robotic platforms do not show adequate flexibility and successfully handle only a few specific tasks. The main reasons for that are the set of specific challenges that farming scenarios propose, ranging from the perceptual aliasing, the lack of geometric structures and landmarks, and the large variability in both the ground conditions and crop types. Therefore, the lack of operating flexibility leads to low profitability of farming robotic platforms and poses a high risk of no return on investment for farmers.

Hence, I focus on enhancing the robot's capabilities in perceiving and modeling the surrounding environment. In particular, I faced the following problems: crop/weed classification, localization and mapping, collaborative mapping, and perception-based control. In the following, a brief recap of each contribution is provided.

## 6.1. Chapter 2: Crop/Weed Segmentation

This chapter focused on the crop/weed segmentation problem, and on reducing the human annotation effort while training effective machine learning classifiers. In this specific context, robots are expected to navigate the environment while detecting weeds, with a high level of accuracy, to carry out a weed control process. To this end, I propose two different methods that reduce the annotation effort required to create adequate training datasets for data-driven classification methods, such as Convolutional Neural Networks. In particular, I initially propose a dataset summarization technique to select from big datasets, a small set of real images to label. The proposed method allows keeping an adequate level of informativeness for the network training procedure. I also address the problem from a different perspective, namely by generating a huge number of realistic images by exploiting a modern graphics engine. Both the proposed approaches show promising results: they allow training effective visual classifiers while involving less effort during the labeling procedure. In conclusion, the overall study provides some simple methods that may help training data-driven approaches for farming robots with smaller annotation efforts.

**Open Problems and Future Directions.** Despite the promising results, there are still some open problems. An open challenge in robotic vision, and in particular for machine learning perception in robotic agriculture, is the adaptation of those methods to long-term operations. Most of the existing solutions, indeed, solely perform initial training before the actual deployment of the considered robotic solution. However, the great extent of farmlands and their fast appearance variability lead the trained model to not be effective in all the possible working conditions.

Another open problem, this time more related to the proposed solution, concerns the the dataset summarization method. Particularly, the informativeness of each subset is evaluated by deep features extracted from the last layers of a neural network. However, this study still relate on a partial annotation of the dataset: an unsupervised approach would be desirable. Other interesting open problems regard the synthetic plant generation algorithm. More specifically: (i) More specifically, part of the saved labeling effort is invested in setting up the generation engine: is there an alternative method that enables to drastically reduce this overhead? (ii) Although segmentation methods exploit the NIR channel, the generation procedure generates annotated RGB images. Is there a way to realistically simulate NIR data? Both the above-mentioned issues might be valuable subjects for the research on design an effective segmentation framework for farming robots. An interesting and viable solution may be in adopting Generative Adversarial Networks (GANs), i.e. training a network to learn how to generate realistic images.

## 6.2. Chapter 3: Mapping and Localization

In this chapter, I introduced a robust approach for improving the robot localization and mapping capabilities within a farm environment. In fact, exploiting the specificity of the agricultural domain, the proposed approach fuses several sensors modalities and exploit domain-specific constraints. In particular, the problem is formulated as a pose-graph minimization in 3D: each sensor reading is fed into the pose-graph with an associated measurement matrix that encodes the noise and the accuracy with respect to each degree of freedom to estimate. The pose-graph is then optimized by minimizing a cost function in a sliding window fashion. The proposed method is flexible and can be used with different sensor modalities or just with a few of them. Indeed, the activation or deactivation of a specific sensor cue act on the accuracy of a specific portion of the robot's state. Experimental results show that, in spite of the simplicity of the proposed technique, it allows the robot to reach sufficient accuracy guaranteeing the robot to not harm the crop. I additionally assessed and benchmark different sensors in the agriculture context, highlighting both their weaknesses and strengths.

**Open Problems and Future Directions.** Despite the flexibility of the proposed method, there are still some open challenges. First of all, the map should properly scale with the farmland extent while highlighting the major characteristics of the target field. Most of the existing work, including the method proposed in this thesis, formulate the map as a sparse set of keypoints, namely a set of salient points, leading the proposed architectures to not be directly applicable to large-scale farming environments.

Other interesting future work directions include: (i) introducing a more accurate and dense Digital Elevation Model. In this regard, a viable solution may be in using DEMs provided by UAVs; (ii) adding additional cues coming from the crop-rows detection or, alternatively, by using semantic data as additional localization landmarks (i.e. crop and weeds). On the other hand, there are still challenging open questions. In particular, how to adapt this approach to different agriculture contexts (i.e. orchards)? How to remove the dependency from the GPS? They both represent research issues on design an effective and reliable localization and mapping framework for farming robots.

## 6.3. Chapter 4: Collaborative Mapping

This chapter presents the AgriColMap framework, namely a method that allows heterogeneous robots such as a UGV and a UAV to build a map in a collaborative manner. This allows the proposed method to produce a highly 3D detailed map of the farm and to collaboratively keep the map up to date. The AgriColMap framework is composed of several steps, each of which focuses a specific sub-problem of the whole process. The key components are in the environment representation and in the data association procedure. The former cast the problem in 2D, enhancing the key characteristics of a farm field while filtering out all the redundant data. The latter, namely a LDOF, allows retrieving an accurate data association by handling the strong perceptual aliasing. Experimental evaluations prove the effectiveness of the proposed solution, providing outstanding accuracy against translational and rotational errors and scale discrepancies as well. Moreover, the outcomes prove how the method well generalize over different crop types.

**Open Problems and Future Directions.** The key drawback of the proposed method is the lower robustness against noisy initial guesses in the map orientations with respect to noise in the translation and scale. This topic represents an interesting future direction to develop a robust collaborative map framework. Additional effort is also required to keep improving the process, starting from a reduction of the computational burden, and including additional features when creating the environmental model used in the registration process. Future research will also focus on the extension of the proposed methodology, for example by considering maps gathered over several weeks. Moreover, an additional interesting point may be in adding semantic information in the built map, such as crop/weed density information. In conclusion, the proposed solution can support further and more challenging research topics in the context of heterogeneous robot collaboration, such as build a temporal map of the target field to monitor the growth and health indexes of the crop.

## 6.4. Chapter 5: Perception Based Control

This chapter addresses the problem of controlling a robot by explicitly considering perception and avoidance constraints. This is a key feature that can provide benefits to farming robots. In the farming context, indeed, the strong perceptual aliasing might mislead the on-board navigation system when performing specific agronomic interventions. The approaches proposed in this thesis formulate the problem as a cost function minimization over a finite time horizon, where the perception and avoidance requirements are fed into the optimization as constraints. The experimental evaluations show the effectiveness of the proposed solution, providing empirical proof of the benefits introduced by these augmented control policies.

**Open Problems and Future Directions.** As remarked in chapter 5, the proposed control architectures have only been tested in indoor and simulated environments. They represent a general assessment of their effectiveness when controlling aerial platforms, and can, therefore, be considered as a preliminary assessment in the direction of their possible application in precision agriculture. Thus, further effort should be put in testing the proposed architectures within a farming environment with real robots, either aerial or terrestrial.

## 6.5. Thesis Statement and Final Remarks

I believe that the proposed thesis represent a small but concrete step forward in the state-of-the-art of robotic systems applied to precision agriculture, with solutions that are easily applicable to a wide range of robots, farm management activities, and crop types. In particular, this thesis explores the role of the perception and environmental mapping in precision farming applications, arguing that:

1. enhancing the perception capabilities of robotic platforms in farming scenarios, such as exploiting the specific features that it proposes, improves the autonomous navigation capabilities;

2. an *ad-hoc* environment representation aids the localization and mapping capabilities and the collaboration between heterogeneous robotic platforms;

3. the development of effective crop/weed segmentation systems can be carried out with a lower annotation effort while keeping a sufficient segmentation accuracy;

4. a control system that takes into account perception and avoidance constraints leads to better performance and higher reliability in on-field operations.

In fact, I proved that, by taking into account the specific features and constraints provided by farming scenarios, it is possible to improve the farming robot capabilities under several aspects. Such a claim is supported by experiments carried out on different components an autonomous robot is commonly equipped with: mapping, localization, collaborative mapping, control, and environment semantic segmentation. Though this thesis is just a starting point for a future research line, it proved the benefits brought by a proper use of the environment specificity in the precision farming context.

# Appendix A

# Differential Flatness for UAV

Let $F_I$ be the right-hand inertial reference frame with unit vectors along the axes denoted by $\{\vec{\mathbf{i}}_x, \vec{\mathbf{i}}_y, \vec{\mathbf{i}}_z\}$. The vector $p = (p_1, p_2, p_3) \in F_I$ denotes the position of the center of mass of the vehicle.

Let $F_B$ be the right-hand body reference frame with unit vectors $\{\vec{\mathbf{b}}_x, \vec{\mathbf{b}}_y, \vec{\mathbf{b}}_z\}$, where these vectors are the axes of frame $F_B$ with respect to frame $F_I$. The orientation of the rigid body is given by the rotation ${}^I R_B = R = \begin{bmatrix} \vec{\mathbf{b}}_x & \vec{\mathbf{b}}_y & \vec{\mathbf{b}}_z \end{bmatrix} \in SO(3)$.

Let $v \in F_I$ express the linear velocity of the body, expressed in the inertial reference frame $F_I$. Let $\omega \in F_B$ be the angular velocity of the body with respect to $F_I$. Let $m$ denote the mass of the rigid body, and $\mathbf{I} \in \mathbb{R}^{3x3}$ the constant inertia matrix expressed in body frame, the rigid body equations can be expressed as

$$\dot{\xi} = v, \tag{A.1a}$$

$$m\dot{v} = mg\vec{\mathbf{i}}_z + RF_t \tag{A.1b}$$

$$\dot{R} = R\,\omega_\times \tag{A.1c}$$

$$\mathbf{I}\dot{\omega} = -\omega \times \mathbf{I}\omega + \tau \tag{A.1d}$$

where the notation $\omega_\times$ denotes for the skew-symmetric matrix formed from $\omega$. The system inputs $F_t, \tau \in F_B$ act respectively as thrust force and body torques. The system reported in (A.1) can be represented exploiting the *differential flatness* property. For an UAV underactuated vehicle, the flat outputs are given as $\zeta = (p_1, p_2, p_3, \gamma) \in \mathbb{R}^4$, where $\gamma$ represents the yaw angle. Hence, by denoting $p_t = (p_1, p_2, p_3)$, it is possible to recover the full state and controls by using the following relations: $p = p_t$, $\dot{p} = \dot{p_t}$, $F_t = \|m(\ddot{p_t} - g)\|$ and the three columns of the rotation matrix $R$ as:

$$R_z = m(\ddot{p_t} - g)/F_t$$

$$R_y = R_z \times \begin{pmatrix} cos\ \gamma_t \\ sin\ \gamma_t \\ 0 \end{pmatrix} / \left\| R_z \times \begin{pmatrix} cos\ \gamma_t \\ sin\ \gamma_t \\ 0 \end{pmatrix} \right\|$$

$$R_x = R_y \times R_z.$$

The angular velocity is recovered as

$$
\begin{gathered}
\omega_x = -R_y \; \dddot{p_t}/F_t \\
\omega_y = R_x \; \dddot{p_t}/F_t \\
\omega_z = \dot{\gamma}_t \; (\vec{\mathbf{i}}_z \; R_z)
\end{gathered}
$$

where $\vec{\mathbf{i}}_z$ is the standard unit vector along the z-axis. To recover $\tau$, $\dot{\omega}$ is firtly recovered. Note that from the dynamics

$$
m \; p_t^{(4)} = (R\hat{\omega}^2 + R\hat{\dot{\omega}})F_t \; \vec{\mathbf{i}}_z + 2 \; R \; \hat{\omega}\dot{F}_t\vec{\mathbf{i}}_z + R \; \ddot{F}_t \; \vec{\mathbf{i}}_z
$$

Solving this for $\dot{\omega}$ gives

$$
\begin{gathered}
\omega_x = (-mR_y \; p_t^{(4)} - \omega_y\omega_z F_t + 2\omega_x\dot{F}_t)/F_t \\
\omega_y = (mR_y \; p_t^{(4)} - \omega_x\omega_z F_t - 2\omega_y\dot{F}_t)/F_t \\
\omega_z = \ddot{\gamma}_t\vec{\mathbf{i}}_z \; R_z + \dot{\gamma}_t\vec{\mathbf{i}}_z^T R\hat{\omega}\vec{\mathbf{i}}_z
\end{gathered}
$$

Then, it is possible to use the dynamics $\tau = \mathbf{I}\dot{\omega} - \mathbf{I}\omega \times \omega$ and $\tau$ is recovered.

# Bibliography

[1] Sapos. URL https://www.sapos.de/.

[2] Nexus robotics. URL http://nexusrobotics.ca/.

[3] Pix4Dmapper by Pix4D. URL http://www.pix4d.com/.

[4] Bosch deepfield robotics for weed control, 2015. URL https://spectrum.ieee.org/automaton/robotics/industrial-robots/bosch-deepfield-robotics-weed-control.

[5] Australian center for field robotics, 2017. URL https://confluence.acfr.usyd.edu.au/display/AGPub/Our+Robots.

[6] Martín Abadi and *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/.

[7] M.J. Aitkenhead, I.A. Dalgetty, C.E. Mullins, A.J.S. McDonald, and N.J.C. Strachan. Weed and crop discrimination using image analysis and artificial intelligence methods. *Computers and Electronics in Agriculture*, 39(3):157–171, 2003.

[8] D. Alexander. 9 robots that are invading the agriculture industry, 2018. URL https://interestingengineering.com/9-robots-that-are-invading-the-agriculture-industry.

[9] E. Altug, J. P. Ostrowski, and R. Mahony. Control of a quadrotor helicopter using visual feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 2002.

[10] C. Anderson. Agricultural drones, 2014. URL https://www.technologyreview.com/s/526491/agricultural-drones/.

[11] T. Baca, P. Stepan, V. Spurny, D. Hert, R. Penicka, M. Saska, J. Thomas, G. Loianno, and V. Kumar. Autonomous landing on a moving vehicle with an unmanned aerial vehicle. *Journal of Field Robotics*. doi: 10.1002/rob.21858.

[12] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, Dec 2017.

[13] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[14] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance control and dynamics*, 21(2):193–207, 1998.

[15] A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, 94(7):1384–1397, 2006.

[16] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[17] S. Blackmore. New concepts in agricultural automation. pages 127–137, London, UK, 2009. HGCA.

[18] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

[19] José L Blanco, Javier González, and Juan-Antonio Fernández-Madrigal. A robust, multi-hypothesis approach to matching occupancy grid maps. *Robotica*, 31:687–701, 2013.

[20] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[21] T. M. Bonanni, B. Della Corte, and G. Grisetti. 3-D map merging on pose graphs. *IEEE Robotics and Automation Letters (RA-L)*, 2(2):1031–1038, 2017.

[22] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[23] T. F. Burks, S. A. Shearer, R. S. Gates, and K. D. Donohue. Backpropagation neural network design and evaluation for classifying weed species using color image texture. *Transactions of the ASAE*, 43(4):1029–1037, 2000.

[24] M. Burri, J. Nikolic, C. Hürzeler, G. Caprari, and R. Siegwart. Aerial service robots for visual inspection of thermal power plant boiler systems. In *Proc. of the 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.

[25] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool. Efficient volumetric fusion of airborne and street-side data for urban reconstruction. In *Proc. of the International Conference on Pattern Recognition (ICPR)*, pages 3204–3209, 2016.

[26] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Europ. Conf. on Computer Vision (ECCV)*, pages 778–792, 2010.

[27] N.A. Che Hussin, N. Jamil, S. Nordin, and K. Awang. Plant species identification by using scale invariant feature transform (SIFT) and grid based colour moment (GBCM). In *Proc. of the IEEE Conference on Open Systems (ICOS)*, pages 226–230, 2013.

[28] N. Chebrolu, T. Läbe, and C. Stachniss. Robust long-term registration of UAV images of crop fields for precision agriculture. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):3097–3104, 2018.

[29] F. Auat Cheein, G. Steiner, G. Perez Paina, and R. Carelli. Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection. *Computers and Electronics in Agriculture*, 78:195 – 207, 2011.

[30] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, 89(2-3), 2003.

[31] William C. Cohen. Optimal control theory: an introduction, donald e. kirk, prentice hall, inc., new york (1971), 452 pages. *AIChE Journal*, 17(4):1018–1018, 1971.

[32] D. Cousins. Self-driving ibex robot sprayer helps farmers safely tackle hills, 2016. URL `https://www.fwi.co.uk/livestock/grassland-management/self-driving-ibex-robot-sprayer-helps-farmers-safely-tackle-hills`.

[33] A. Craft. Making it rain: Drones could be the future for cloud seeding, 2017. URL `https://www.foxnews.com/tech/making-it-rain-drones-could-be-the-future-for-cloud-seeding`.

[34] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar. Devices, systems, and methods for automated monitoring enabling precision agriculture. In *Proc. of the IEEE International Conference on Automation Science and Engineering (CASE)*, pages 462–469, 2015.

[35] J. Deere. Want a really hard machine learning problem? try agriculture, 2019. URL `https://spectrum.ieee.org/view-from-the-valley/robotics/artificial-intelligence/want-a-really-hard-machine-learning-problem-try-agriculture-say-john-deere-labs-lea`

[36] M. Di Cicco, C. Potena, G. Grisetti, and A. Pretto. Automatic model based dataset generation for fast and accurate crop and weeds detection. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5188–5195, 2017.

[37] J. Dong, J. G. Burnham, B. Boots, G. Rains, and F. Dellaert. 4D crop monitoring: Spatio-temporal reconstruction for agriculture. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[38] V. Drevelle and P. Bonnifait. A set-membership approach for high integrity height-aided satellite positioning. *GPS Solutions*, 15(4):357–368, October 2011.

[39] Tom Duckett, Simon Pearson, Simon Blackmore, Bruce Grieve, and Melvyn Smith. White paper - agricultural robotics: The future of robotic agriculture, 2018.

[40] A. English, P. Ross, D. Ball, and P. Corke. Vision based guidance for robot navigation in agriculture. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[41] A. English, P. Ross, D. Ball, and P. Corke. Vision based guidance for robot navigation in agriculture. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1693–1698, May 2014.

[42] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5774–5781, May 2017. doi: 10.1109/ICRA.2017.7989679.

[43] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, Oct 2018. doi: 10.1109/IROS.2018.8593739.

[44] Jay Farrell. *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill, Inc., 2008.

[45] Joachim Ferreau. *An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control*. PhD thesis, 01 2006.

[46] F. Feyaerts and L. Van Gool. Multi-spectral vision system for weed detection. *Pattern Recognition Letters*, 22(6-7):667–674, 2001.

[47] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. In *British Machine Vision Conference*, pages 662–670, 2001.

[48] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International journal of control*, 61(6):1327–1361, 1995.

[49] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Air-ground localization and map augmentation using monocular dense reconstruction. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[50] Clément Fouque and Philippe Bonnifait. On the use of 2D navigable maps for enhancing ground vehicle localization. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[51] C. Frueh and A. Zakhor. Constructing 3D city models by merging ground-based and airborne views. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[52] C. Fruh and A. Zakhor. Constructing 3D city models by merging aerial and ground views. *IEEE Computer Graphics and Applications*, 23(6):52–61, 2003.

[53] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. 2016.

[54] S. Nigel G. Stefan, P. Jeremy and L. Geoff. Real-time procedural generation of ‘pseudo infinite’ cities. In *Proc. of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, 2003.

[55] G. Garimella, M. Sheckells, and M. Kobilarov. Robust Obstacle Avoidance for Aerial Platforms Using Adaptive Model Predictive Control. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[56] A. Gawel, T. Cieslewski, R. Dubé, M. Bosse, R. Siegwart, and J. Nieto. Structure-based vision-laser matching. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[57] Abel Gawel, Renaud Dubé, Hartmut Surmann, Juan Nieto, Roland Siegwart, and Cesar Cadena. 3d registration of aerial and ground robots for disaster response: An evaluation of features, descriptors, and transformation estimation. In *Proc. of the IEEE SSRR*, 2017.

[58] Arturo Gil, Óscar Reinoso, Mónica Ballesta, and Miguel Juliá. Multi-robot visual SLAM using a rao-blackwellized particle filter. *Robotics and Autonomous Systems*, 58(1):68 – 80, 2010.

[59] M. Greeff and A. P. Schoellig. Flatness-based model predictive control for quadrotor trajectory tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6740–6745, Oct 2018. doi: 10.1109/IROS.2018.8594012.

[60] G. Grisetti, R. Kuemmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4): 31–43, 2010. doi: 10.1109/MITS.2010.939925.

[61] Nicolas Guenard, Tarek Hamel, and Robert E. Mahony. A practical visual servo control for a unmanned aerial vehicle. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[62] A. H. A. Hafez, E. Cervera, and C. V. Jawahar. Hybrid visual servoing by boosting ibvs and pbvs. In *Proc. of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, 2008.

[63] Tarek Hamel and Robert Mahony. Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach. *IEEE Transactions on Robotics and Automation*, 18(2):187–198.

[64] Hironori Hattori, Yasodekshna Vishnu Naresh Boddeti, Kris M Kitani, and Takeo Kanade. Learning scene-specific pedestrian detectors without real data. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[65] S. Haug, P. Biber, A. Michaels, and J. Ostermann. Plant stem detection and position estimation using machine vision. In *13th Intl. Conf. on Intelligent Autonomous Systems (Workshops)*, 2014.

[66] S. Haug, A. Michaels, P. Biber, and J. Ostermann. Plant classification system for crop/weed discrimination without segmentation. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1142–1149, March 2014.

[67] J. Hemming and T. Rath. PA-precision agriculture: Computer-vision-based weed identification under field conditions using controlled lighting. *Journal of Agricultural Engineering Research*, 78(3):233–243, 2001.

[68] Timo Hinzmann, Thomas Stastny, Gianpaolo Conte, Patrick Doherty, Piotr Rudol, Mariusz Wzorek, Enric Galceran, Roland Siegwart, and Igor Gilitschenski. Collaborative 3D reconstruction using heterogeneous UAVs: System and experiments. In *Proc. of the Intl. Sym. on Experimental Robotics (ISER)*, pages 43–56.

[69] Christian Hirt. *Digital Terrain Models*, pages 1–6. Springer International Publishing, 2014.

[70] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[71] B. Houska, HJ Ferreau, and M. Diehl. ACADO toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011. ISSN 1099-1514. doi: 10.1002/oca.939. URL http://dx.doi.org/10.1002/oca.939.

[72] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *Intl. Journal of Robotics Research (IJRR)*, 25(12):1243–1256, 2006.

[73] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine patch match for large displacement optical flow. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[74] M. Imperoli, C. Potena, D. Nardi, G. Grisetti, and A. Pretto. An effective multi-cue positioning system for agricultural robotics. *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[75] J. Jessup, S. N. Givigi, and A. Beaulieu. Robust and efficient multi-robot 3D mapping with octree based occupancy grids. In *Proc. of the IEEE Intl. Conf. on Systems, Man, and Cybernetics (SMC)*, 2014.

[76] Jin Jian and Tang Lie. Corn plant sensing using real-time stereo vision. *Journal of Field Robotics*, (6-7):591–608.

[77] Dongwon Jung and Panagiotis Tsiotras. On-line path generation for unmanned aerial vehicles using b-spline path templates. *Journal of Guidance Control and Dynamics*, 36, 11 2013. doi: 10.2514/1.60780.

[78] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *IEEE Conference on Control Applications (CCA)*, 2015. doi: 10.1109/CCA.2015.7320769.

[79] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto. Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243, Sep. 2017. doi: 10.1109/IROS.2017.8202163.

[80] Mina Kamel, Michael Burri, and Roland Siegwart. Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. *arXiv:1611.09240*, 2016.

[81] Roman Käslin, Péter Fankhauser, Elena Stumm, Zachary Taylor, Elias Mueggler, Jeffrey Delmerico, Davide Scaramuzza, Roland Siegwart, and Marco Hutter. Collaborative localization of aerial and ground robots through elevation maps. In *Proc. of the IEEE SSRR*, 2016.

[82] Raghav Khanna, Martin Möller, Johannes Pfeifer, Frank Liebisch, Achim Walter, and Roland Siegwart. Beyond point clouds-3d mapping and field parameter measurements using uavs. In *IEEE ETFA*, 2015.

[83] M. Kise, Q. Zhang, and F. Rovira Más. A stereovision-based crop row detection method for tractor-automated guidance. *Biosystems Engineering*, 90(4):357 – 367, 2005.

[84] Florian Kraemer, Alexander Schaefer, Andreas Eitel, Johan Vertens, and Wolfram Burgard. From plants to landmarks: Time-invariant plant localization that uses deep pose regression in agricultural fields. *In arXiv:1709.04751.*

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of a Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012.

[86] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. Lopez, and J. V. B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *The 12th European Conference on Computer Vision (ECCV)*, pages 502–516, 2012.

[87] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. A navigation system for robots operating in crowded urban environments. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[88] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[89] Keerthy Kusumam, Tomáš Krajník, Simon Pearson, Tom Duckett, and Grzegorz Cielniak. 3d-vision based detection, localization, and sizing of broccoli heads in the field. *Journal of Field Robotics*, 34(8):1505–1518, 2017. doi: 10.1002/rob.21726. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21726`.

[90] D. Lee, H. Lim, H Jin Kim, Y. Kim, and K. Seong. Adaptive image-based visual servoing for an underactuated quadrotor system. *Journal of Guidance, Control, and Dynamics*, 35:1335–1353, 07 2012. doi: 10.2514/1.52169.

[91] Sue Han Lee, Chee Seng Chan, P. Wilkin, and P. Remagnino. Deep-plant: Plant identification with convolutional neural networks. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, pages 452–456, 2015.

[92] W.S. Lee, V. Alchanatis, C. Yang, M. Hirafuji, D. Moshou, and C. Li. Sensing technologies for precision specialty crop production. *Computers and Electronics in Agriculture*, 74(1):2 – 33, 2010. ISSN 0168-1699.

[93] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 37, pages 510–520, 2011.

[94] Roland van der Linden, Ricardo Lopes, and Rafael Bidarra. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):78 – 89, mar 2014.

[95] P. Lottes and C. Stachniss. Semi-supervised online visual crop and weed classification in precision farming exploiting plant arrangement. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5155–5161, Sep. 2017.

[96] P. Lottes, M. Hoeferlin, S. Sander, M. Müter, P. Schulze, and L. C. Stachniss. An effective classification system for separating sugar beets and weeds for precision farming applications. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5157–5163, May 2016.

[97] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss. Uav-based crop and weed classification for smart farming. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3024–3031, May 2017.

[98] S Lynen, M Achtelik, S Weiss, M Chli, and R Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[99] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32, 2012.

[100] Michele Mancini, Gabriele Costante, Paolo Valigi, and Thomas A. Ciarfuglia. Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 4296–4303, 2016.

[101] M. Mazur. Six ways drones are revolutionizing agriculture, 2016. URL https://www.technologyreview.com/s/601935/six-ways-drones-are-revolutionizing-agriculture/.

[102] R. Mebarki, V. Lippiello, and B. Siciliano. Autonomous landing of rotary-wing aerial vehicles by image-based visual servoing in gps-denied environments. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, Oct 2015. doi: 10.1109/SSRR.2015.7443009.

[103] U. Meier. Growth stages of mono-and dicotyledonous plants. Technical report, Federal Biological Research Centre for Agriculture and Forestry, 2001.

[104] Luis Mejias, Srikanth Saripalli, Pascual Campoy, and Gaurav S. Sukhatme. Visual servoing of an autonomous helicopter in urban areas using feature tracking. *Journal of Field Robotics*, 23(3-4):185–199, 2006.

[105] Henrik S. Midtiby, Thomas M. Giselsson, and Rasmus N. Jørgensen. Estimating the plant stem emerging points (PSEPs) of sugar beets at early growth stages. *Biosystems Engineering*, 111:83 – 90, 2012.

[106] M. W. Mueller, M. Hehn, and R. D'Andrea. A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3480–3486, Nov 2013.

[107] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[108] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

[109] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

[110] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1398–1404, 2016.

[111] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time SLAM.

[112] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[113] Michael Nørremark, Hans W Griepentrog, Jon Nielsen, and Henning Tangen Søgaard. The development and assessment of the accuracy of an autonomous GPS-based system for intra-row mechanical weed control in row crops. *Biosystems Engineering*, 101(4):396–410, 2008.

[114] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran. Continuous-time trajectory optimization for online uav replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339, Oct 2016. doi: 10.1109/IROS.2016.7759784.

[115] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[116] S. Omari, M. D. Hua, G. Ducard, and T. Hamel. Nonlinear control of VTOL UAVs incorporating flapping dynamics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[117] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[118] A. A. Paranjape, K. C. Meier, X. Shi, S. Chung, and S. Hutchinson. Motion primitives and 3-d path planning for fast flight through a forest. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2940–2947, Nov 2013. doi: 10.1109/IROS.2013.6696773.

[119] O. M. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Cats and dogs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3498–3505, 2012.

[120] I. Parra, M. Ángel Sotelo, D. F. Llorca, C. Fernández, A. Llamazares, N. Hernández, and I. García. Visual odometry and map fusion for GPS navigation assistance. In *IEEE International Symposium on Industrial Electronics (ISIE)*, 2011.

[121] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 27(8):1226–1238, 2005.

[122] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3): 287–296, 1985.

[123] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy. Computer vision based general object following for gps-denied multirotor unmanned vehicles. In *2014 American Control Conference*, pages 1886–1891, June 2014. doi: 10.1109/ACC.2014.6858831.

[124] Jesús Pestana, Jose Luis Sanchez-Lopez, Srikanth Saripalli, and Pascual Campoy Cervera. Computer vision based general object following for gps-denied multirotor unmanned vehicles. In *Proc. of the American Control Conference (ACC)*, 2014.

[125] Thomas Petersen. A comparison of 2D-3D pose estimation methods. *Aalborg University, Aalborgb*, 2008.

[126] Matt Pharr and Greg Humphreys. *Physically based rendering: From theory to implementation.* Morgan Kaufmann, 2004.

[127] Ciro Potena, Daniele Nardi, and Alberto Pretto. Fast and accurate crop and weed identification with summarized train sets for precision agriculture. In *Intelligent Autonomous Systems 14*, pages 105–121. Springer International Publishing, 2017.

[128] S. M. Prakhya, L. Bingbing, Y. Rui, and W. Lin. A closed-form estimate of 3D ICP covariance. In *IAPR International Conference on Machine Vision Applications (MVA)*, 2015.

[129] A. Pretto, E. Menegatti, and E. Pagello. Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[130] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl. Autogenerating microsecond solvers for nonlinear mpc: A tutorial using acado integrators. *Optimal Control Applications and Methods*, 36(5), 2015. ISSN 1099-1514. doi: 10.1002/oca.2152. URL http://dx.doi.org/10.1002/oca.2152.

[131] Joern Rehder, Kamal Gupta, Stephen T. Nuske, and Sanjiv Singh. Global pose estimation with limited GPS and long range visual odometry. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[132] A. K. Reyes, J. C. Caicedo, and J. E. Camargo. Fine-tuning deep convolutional networks for plant recognition. In *Working Notes of Conference and Labs of the Evaluation forum (CLEF)*, 2015.

[133] Charles Richter, Adam Bry, and Nicholas Roy. *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*, pages 649–666. Springer International Publishing, 2016. ISBN 978-3-319-28872-7. doi: 10.1007/978-3-319-28872-7_37.

[134] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, 2016.

[135] J. W. Rouse, Jr., R. H. Haas, J. A. Schell, and D. W. Deering. Monitoring vegetation systems in the great plains with ERTS. In *Proc. of the 3rd Earth Resource Technology Satellite (ERTS) Symposium*, volume 1, 1974.

[136] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 2564–2571, 2011. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126544.

[137] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[138] I. Sa, Z. Chen, M. Popović, R. Khanna, F. Liebisch, J. Nieto, and R. Siegwart. weednet: Dense semantic weed classification using multispectral images and mav for smart farming. *IEEE Robotics and Automation Letters*, 3(1):588–595, Jan 2018.

[139] S. Saeedi, L. Paull, M. Trentini, and H. Li. Multiple robot simultaneous localization and mapping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[140] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics (JFR)*, 33(1):3–46. doi: 10.1002/rob.21620.

[141] David Schleicher, Luis M. Bergasa, Manuel Ocana, Rafael Barea, and Maria E. Lopez. Real-time hierarchical outdoor SLAM based on stereovision and GPS fusion. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):440 – 452, 2009.

[142] Markus Schreiber, Hendrik Königshof, André-Marcel Hellmund, and Christoph Stiller. Vehicle localization with tightly coupled GNSS and visual odometry. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2016.

[143] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[144] A. Shafei, James J. Little, and M. Schmidt. Play and learn: Using video games totrain computer vision models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[145] Q. Shan, C. Wu, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz. Accurate geo-registration by ground-to-aerial image matching. In *2nd Int. Conf. on 3D Vision*, 2014.

[146] S. A. Shearer and R. G. Holmes. Plant identification using color co-occurrence matrices. *Transactions of the ASAE*, 33(6):2037–2044, 1990.

[147] Matthew Sheckells, Gowtham Garimella, and Marin Kobilarov. Optimal visual servoing for differentially flat underactuated systems. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[148] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2005.

[149] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[150] Ancha Srinivasan. *Handbook of precision agriculture: principles and applications.* CRC press, 2006.

[151] Albert Stoll and Heinz Dieter Kutzbach. Guidance of a forage harvester with GPS. *Precision Agriculture*, 2(3):281–291, 2000.

[152] Nathan Michael *et al.* Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics (JFR)*, 29(5): 832–841, Sept 2012.

[153] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar. Visual servoing of quadrotors for perching by hanging from cylindrical objects. *IEEE Robotics and Automation Letters*, 1(1):57–64, Jan 2016.

[154] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005.

[155] B Thuilot, C Cariou, L Cordesses, and P Martinet. Automatic guidance of a farm tractor along curved paths, using a unique CP-DGPS. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[156] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010. doi: 10.1109/TPAMI.2009.77.

[157] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers. Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 215–222, Sep. 2017. doi: 10.1109/IROS.2017.8202160.

[158] L. T. Vinh, S. Lee, Y. Park, and B. J. d'Auriol. A novel feature selection method based on normalized mutual information. *Applied Intelligence*, 37(1): 100–120, 2011.

[159] K. Vishal, C. V. Jawahar, and V. Chari. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[160] C. Wang, K. Wilson, and N. Snavely. Accurate georegistration of point clouds using geographic data. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 33–40, 2013.

[161] Rui Wang, Martin Schwörer, and Daniel Cremers. Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, 2017.

[162] Ulrich Weiss and Peter Biber. Plant detection and mapping for agricultural robots using a 3D lidar sensor. *Robotics and autonomous systems*, 59(5): 265–273, 2011.

[163] W. Winterhalter, F. V. Fleckenstein, C. Dornhege, and W. Burgard. Crop row detection on tiny plants with the pattern hough transform. *IEEE Robotics and Automation Letters*, 3(4):3394–3401, Oct 2018.

[164] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3D to 2D label transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[165] Sajjad Yaghoubi, Negar Ali Akbarzadeh, Shadi Sadeghi Bazargani, Sama Sadeghi Bazargani, Marjan Bamizan, and Maryam Irani Asl. Autonomous robots for agricultural tasks and farm assignment and future trends in agro robots, 2015.

[166] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2016.

[167] B Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1577–1584, 2011.

[168] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3320–3328. 2014.